OS/VS DB/DC Data Dictionary
Administration and Customization Guide

Licensed Program
Program Number 5740-XXF
Release 6
Second Edition (December 1986)

This is a major revision of, and makes obsolete, SH20-9174-1.

This edition applies to Release 6 of OS/VS DB/DC Data Dictionary, Licensed Program 5740-XXF, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Changes" following the preface. Because the technical changes in this edition are extensive and difficult to localize, they are not marked by vertical bars in the left margin.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370, 30xx, and 4300 Processors Bibliography, GC29-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's program may be used. Any functionally equivalent program may be used instead.

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. If you request publications from the address given below, your order will be delayed because publications are not stocked there.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California, U.S.A. 95150. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1979, 1984, 1986
Preface

This publication is addressed to Dictionary administrators and other administrative personnel who are responsible for establishing the OS/VS DB/DC Data Dictionary at an installation. These administrators exercise control over the content of the definitions in OS/VS DB/DC Data Dictionary data bases. These overall responsibilities can be collectively called Dictionary administration.

The publication is also for system programmers and those who code user programs and validation routines in response to specifications defined by end users, systems analysis personnel, or data base administrators.

The content of this publication applies to OS/VS DB/DC Data Dictionary, program number 5740-XXF. For brevity, this publication refers to the set of programs and data bases that make up the licensed program as the Dictionary.

How to Use This Publication

This publication is a guide to the ongoing administration of the Dictionary, and to providing data services for groups of end users. Information on planning for installation and operation is consolidated in OS/VS DB/DC Data Dictionary Installation Guide.

Uses of this publication are:

- To help you plan for and control access to data stored in your Dictionary data bases
- As a reference manual for Dictionary extensions to describe data resources not supported by the Dictionary standard categories
- As programming reference material that specifies how COBOL, PL/I, and Assembler programs can be coded to access Dictionary data
- As a guide for the transfer of data from one Dictionary to another
- As a guide on how to prepare for and use the Language Preprocessor
Prerequisite Knowledge

Before using this publication, you should understand the basic concepts of the Dictionary system, regarding its approach to documenting the programming resources in a DB/DC environment. This can be found in OS/VS DB/DC Data Dictionary General Information Manual. It is assumed that you understand IMS/VS concepts and facilities or CICS/OS/VS concepts and facilities with DL/I applications, and the access methods used by DL/I. The system programmer using this publication should also have programming language skills.

In this publication, it is assumed that the reader is familiar with the operation of the Dictionary, both online and batch, and understands the use and syntax of Dictionary commands. This information is in OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference.

How This Publication Is Organized

- **Chapter 1** contains an overview of general administration and planning topics and summarizes the tasks involved.

- **Chapter 2** includes both planning and implementation information for operation of the Dictionary with authorization checking active, and describes how to define user access profiles with commands or display forms.

- **Chapter 3** gives the conceptual information necessary for the design of extensions and customization for the Dictionary. Installation-defined categories and types of relationships are described and the definition process is explained. The chapter also outlines the potential for user-written programs to retrieve data from the Dictionary data bases.

- **Chapter 4** specifies how to define attributes and associate them with new categories and types of relationships.

- **Chapter 5** specifies how to define new subject categories with associated attributes. It also describes what reports are available to assist with the documentation of such categories.

- **Chapter 6** specifies how to define new types of relationships with associated relationship data. It also describes what reports are available to document such relationships.

- **Chapter 7** specifies how the definitions of new subject categories and relationship-types are made operational for the end user.

- **Chapter 8** describes how you can modify installation-defined categories and relationship-types.

- **Chapter 9** describes the use of display forms to define and report on Dictionary category and relationship extensions.

- **Chapter 10** discusses how to code and use validation routines.
• Chapter 11 specifies how groups of relationship-types can be defined for specialized processing of Dictionary data, and describes how to implement structure-type definitions.

• Chapter 12 specifies the Dictionary support for user programs that retrieve data from the Dictionary data bases. It gives detailed reference information to enable the system programmer to code those user programs.

• Chapter 13 describes all aspects of preparing to make the Dictionary Language Preprocessor available to users.

• Chapter 14 describes the facilities available for transferring data from one Dictionary to another.

• Appendix A gives a summary discussion of the desirability of standardization of Dictionary definitions and gives some suggestions for their maintenance.

• Appendix B contains reference material useful for writing programs that use the Program Access facility.

• Appendix C gives specifications for output record formats produced by the EXPORT command.

Prerequisite Publications

Before using this publication, you should read OS/VS DB/DC Data Dictionary General Information Manual, GH20-9104, as an introduction to the basic application of the Dictionary, the modes of operation, and the terminology associated with the Dictionary.

Corequisite Publications

• OS/VS DB/DC Data Dictionary Terminal User’s Guide and Command Reference, SH20-9189, contains details of how to operate the batch and online facilities, together with an explanation of the command conventions and syntax.


• OS/VS DB/DC Data Dictionary Messages and Codes, SH20-9211, conveniently packages the Dictionary messages so that they can be used by a cross-section of all users, including Language Preprocessor users.

• OS/VS DB/DC Data Dictionary Applications Guide, SH20-9190, identifies how user tasks are performed with the Dictionary, and explains the type of support offered by the product.
Related Publications


Several of the steps described in the Installation Guide support the implementation of services described in this publication.

- *IMS/VS General Information Manual*, GC26-4180, contains a general introduction to IMS/VS, the facilities it provides, and machine and program requirements.

For installations with CICS/OS/VS:


Notational Conventions

In this publication, a uniform system of notation describes the format of Dictionary commands. This notation is not part of the language; it merely provides a basis for describing the structure of the commands.

The command-format illustrations in this publication use the same conventions as those in *OS/VS DB/DC Data Dictionary Terminal User’s Guide and Command Reference*:

- Required operands appear above the bold line.
- Optional operands appear below the bold line.
- Optional operands indicate a choice of entries; unless a default is indicated, you must choose one of the entries.
- Items separated by a vertical bar represent alternative items. No more than one of the items may be selected.
- An ellipsis (...) indicates that multiple entries of the type immediately preceding the ellipsis are allowed.
- Other punctuation (parentheses, commas, spaces, and so forth) must be entered as shown.
- Uppercase type indicates the exact characters to be entered. Such items must be entered exactly as shown.
- Lowercase type indicates values to be supplied by the user.
- Bold branch lines indicate a default option. If the operand is omitted, the value inside the branch line is assumed.
- Entries in a vertical list represent a choice of entries.
Summary of Changes

Release 6

Release 6 of the OS/VS DB/DC Data Dictionary operates as an application program with:

- IMS/VS Version 1 Release 3
- IMS/VS Version 2 Release 1
- CICS/OS/VS Version 1 Release 7 with IMS/VS Version 1 Release 3 DB
- CICS/OS/VS Version 1 Release 7 with IMS/VS Version 2 Release 1 DB

and any subsequent versions or releases until otherwise specified.

The Dictionary's Interactive System Productivity Facility (ISPF) dialog operates on any MVS system with TSO and ISPF Version 2.2 (and any subsequent releases).

Dictionary Processing with ISPF

Time Sharing Option (TSO) users can have access to and update the Dictionary data bases online, using the Interactive System Productivity Facility (ISPF).

- The Dictionary's ISPF dialog includes a set of interactive display panels that allows a user to access and update the Dictionary data bases online from the user's terminal. Dialog messages and help panels are also provided.

- The Dictionary's ISPF interface allows user-written dialogs to initiate execution of the Dictionary, and to submit commands or Program Access facility requests.

Dictionary access control and use of the Double Byte Character Set (DBCS) are also supported in the ISPF environment.
Double Byte Character Set Support

A Double Byte Character Set (DBCS) is a character set in which each symbol is represented by 2 bytes. The Dictionary supports DBCS for languages having character sets that exceed the 256 characters allowed in a single-byte code, and that are supported by the appropriate software and hardware.

Users may facilitate communication by entering, displaying online, and generating output with DBCS strings. Users are provided the option of displaying EXPLANATION forms in either English or Japanese. In addition, the EXPLANATION forms for installation-defined definitions can be entered and displayed with DBCS characters.

Users who process Dictionary data from the ISPF environment can also enter DBCS strings. In addition, the predefined ISPF panels and messages are available in both English and Japanese.

The Report and File Generator (RFG) feature also supports DBCS.

Fast Path Support

Dictionary commands and the Interactive Display Forms facility allow the required IMS/VS descriptors to be specified to support the Fast Path facility of IMS/VS Version 1, Release 3. Within a given application program, a user can make use of Fast Path to improve performance for simply structured data bases, while using the full function of IMS/VS for more complex data bases.

Dictionary Release 6 provides support for two Fast Path data bases: Main Storage Data Base (MSDB) and Data Entry Data Base (DEDB). The MSDB stores and provides access to an installation's most frequently used data. The DEDB provides a high level of availability for, and efficient access to, large volumes of detailed data.

Fast Path support is reflected by a change in the Program Access facility instance number (PAFINST) from a two-digit to a three-digit instance number.

Other Changes

Enhanced Trace Facility

The Dictionary Trace facility is enhanced to allow users more control in invoking and executing the tracing of Dictionary execution, in either batch or online modes. Users can also select the subset of modules for which output is produced. The user can choose whether the module entry/return trace output is generated or not, and also whether any additional type of trace output is generated.
Publication Additions

The OS/VS DB/DC Data Dictionary Interactive System Productivity Facility User's Guide provides information on how to access and update the Dictionary data bases from the ISPF environment. Examples of the content and interaction of each of the Dictionary's ISPF display panels are provided.

The OS/VS DB/DC Data Dictionary Interactive Display Forms Facility User's Guide provides descriptions of the most frequently used data entry procedures. Examples of the content and interaction of most of the available display forms are illustrated.

Compatibility (Release 6 with Release 5)

The OS/VS DB/DC Data Dictionary, Release 6, includes all the functions of the OS/VS DB/DC Data Dictionary Release 5. The format of the Dictionary data bases is modified in Release 6 with the addition of new fields added to existing DL/I segments.

Upward compatibility from Release 5 to Release 6 is provided for Extensibility validation routines.

Release 6 has the following inconsistencies with Release 5:

- New PACA and PADAFT formats are used to define PACAs and PADAFTs for Release 6 to allow the retrieval of all 240 possible DATASET groups. The format code for the new binary format is 'B'. Any Program Access facility program compiled or recompiled after update to Release 6 must use binary PAFINST numbers rather than character PAFINST numbers.

- If you are operating under Release 6 of the Dictionary (or any subsequent releases), you can selectively retrieve a list of subject names, based on their status, by specifying a valid status code (A-T, 0-9, *) for the PACA field PASTATUS. Use of the PASTATUS field is valid only when PACAFM=B.

- For the updated RFG to be compatible with Dictionary Release 5 and Release 6 (or any subsequent releases), a new RFG default parameter has been added. The format of the default parameter is:

  PAFORMT = A | B

PAFORMT=A indicates that the base Dictionary is Dictionary Release 5 and PAFORMT=B indicates that the base Dictionary is Release 6 (or any subsequent releases).
Report and File Generator Feature

The Report and File Generator (RFG) is a separate feature of the Dictionary. It is an application program that uses the Dictionary Program Access facility. It provides full access to Dictionary data and functions for retrieving and processing that data. Data retrieved from the Dictionary can be formatted into reports to be printed and records to be stored in an external file.

The RFG provides a language that can be used to define and generate reports or other output that can be customized to contain the desired information in a format of the user's choosing. Users can, optionally, define and generate reports that can be used in lieu of reports currently being produced by the Dictionary. A set of output definitions for the complete reports shipped with the RFG are similar in function and format to some of the current Dictionary reports, but are not meant to be the same as the current reports.

The RFG can be run with either the OS/VS DB/DC Data Dictionary Release 5 or Release 6 (or any subsequent releases).

Publication Changes

OS/VS DB/DC Data Dictionary Report and File Generator User's Guide and Reference is a new publication that describes how to develop output definitions and execute the RFG. OS/VS DB/DC Data Dictionary Installation Guide consolidates all planning information needed for installation of the Dictionary and the RFG. Information on establishing defaults and execution JCL for both the Dictionary and the RFG are included.

Release 5

Language Preprocessor

An additional Dictionary program to be used as a preprocessor within a source language compilation procedure is available. As an alternative to COPY library processing, source language data structures can be extracted directly from the Dictionary and embedded in the input to a compiler. Support is for COBOL, PL/I, and Assembler.

Dictionary Structure Processing

A structure-type definition can be established which determines the categories to be included and how the relationship-types existing between those categories are to be interpreted. Structure-type definitions are recorded as STRTYPE subjects, which can be installed and later referred to by other Dictionary commands.

Dictionary structure-types composed of standard and/or installation-defined categories and relationship-types can be used as the basis of various Dictionary
processing functions such as: COPY, DELETE__STRUCTURE, EXPORT, and STRUCTURE__REPORT.

Individual Control of Dictionary Output

Output Routing: Individual users of the Dictionary, both in online and batch usage, can route their Dictionary output from commands to an OS/VS sequential or partitioned data set.

IMS/VS online users can specify the output destination as a logical terminal name.

Facilities are also provided for embedding Dictionary output within header and trailer data. One application of this function, under OS/VS2 MVS, is that Dictionary output can be embedded in JCL and be routed directly to an MVS reader.

Output Page Size: The number of lines on a report page can be dynamically changed so that the output is suited to the chosen output device.

Command Processing Flexibility

Stack Processing: Provision is made for the repetitive execution of a single command for a group of Dictionary subjects. Individual subject names are placed in a “stack” in a set of User Data or in an OS/VS sequential or partitioned data set. One way to build a stack is with output from the SCAN command. The SETSTACK command identifies the location of the stack. The RUNSTACK command causes a specified command to be repetitively executed for each member of the stack; for example, a series of STRUCTURES__OUT commands.

CLIST Processing: Dictionary commands themselves, not just subject names, can be grouped within a list kept in a chosen set of User Data. The choice of the subject whose User Data contains the CLIST is specified with the SETCLLIST command. The RUNCLLIST command causes all the commands stored within the specified set of User Data to be executed in turn. This capability enables groups of commands, such as a set of reports, to be executed with only two commands.

Reports Based on Structure Processing: The STRUCTURE__REPORT command provides a report containing all subjects occurring in a specified structure, with various options that control the amount of data included.

Other Dictionary Command Changes: The action of the ADD__RELATIONSHIP command is dependent on a default chosen for the installation as regards access of one subject in the relationship to which the user does not have update authority. The action of the COPY and DELETE__STRUCTURE commands is changed to prevent copying or deleting of subjects to which the user is not authorized.

Both the DELETE__DATA and DELETE__RELATIONSHIP__DATA commands can have all associated data segments specified for deletion. In addition, a range of associated text data segments may be specified for the DELETE__DATA command.

When text data is being entered, a code of “L” causes the input line to be entered as the last line.
The PUNCH command can punch Description as well as User Data lines.

Interactive Display Form Changes

*Display Forms Usability:* Access between the various common subordinate forms has been improved for processing a subject on a display form. Also, most forms that show a relationship to other Dictionary subjects allow a user to move directly to the form for the related subject without a return to the Header form. The COMMAND form can be accessed from most other display forms.

*Relationship Forms:* The two SUBJECT-RELATIONSHIP display forms are now used generally for processing a relationship. The RELATED-ENTITIES form is deleted.

A MODE option of "old" has been made consistent for all relationship forms.

*File Specification:* The appropriate display forms are modified to display and edit the definition for "file" (DBTYPE=F) data bases. This capability is equivalent to that previously available only via the commands.

Customizing the Dictionary

*Modification of Customized Entities:* Installation-defined definitions for categories and relationship-types can be modified. Within certain restrictions, this allows changes to the definition of attribute data previously defined for subjects in the category or instances of the relationship-type.

Data already existing under the old definition is automatically converted and migrated to the new definition constraints.

*Subject Name Validation Routines:* The installation can now add subject name validation routines for standard categories. When a subject is added to the Dictionary, the installation can check that the user name conforms to established naming conventions. These routines can be added at any time, and normal operations do not have to be interrupted.

Additional Dictionary Administration Capabilities

*Portability of Dictionary Data:* Provision is made with the EXPORT command to extract Dictionary data and create output consisting of Dictionary update commands and batch forms input that can become input to another Dictionary.

The "exported" output can then be reviewed for its effect on the receiving Dictionary (without updating the Dictionary data bases) with the PREVIEW command. When any potential conflicts have been resolved, the exported stream can be entered into the receiving Dictionary as a normal batch job.

*Control of Command Usage through the Access Profile:* The access profiles established for individual Dictionary users are expanded to specify one or more groups of commands that are authorized.
Dictionary Subject Name Defaults: The Dictionary subject name defaults can be obtained from the access profile. In online mode, the defaults are displayed on the Header form on completion of sign-on.

Dictionary defaults can now be changed by assembling and link-editing a single module.

Diagnostic Tracing

A facility for starting and stopping a trace of internal events has been added. Tracing greatly helps in documenting the flow of control when diagnosing a problem. In addition to the TRACE command, a dump of Dictionary control information can be obtained dynamically, using the SNAP command.

Other Changes

CICS/OS/VS Storage Utilization: Changes to the CICS/OS/VS transaction achieve efficiencies in the use of storage when multiple screens of data have to be presented.

IMS/VS System Definition: Support for the DATABASE and TRANSACT macros now includes all parameters present for the IMS/VS V1 R2 level.
Contents

Chapter 1. Overview of Dictionary Administration ........................................ 1
    How Administration Relates to Other Dictionary Tasks .......................... 1
        Planning for Dictionary Installation ....................................... 1
        Installing the Dictionary .................................................. 2
        Administering the Dictionary ............................................ 2
        Using the Dictionary ...................................................... 2
        Diagnosis ................................................................. 3
    Dictionary Administration Tasks ................................................. 3
    Characterizing Groups of Dictionary Users ....................................... 4
        Supporting Dictionary Administration .................................. 5
        Supporting Dictionary Language Preprocessor Users ................. 5
        Supporting Occasional Users ............................................ 6
    Double Byte Character Set Support ............................................. 6
        The Shift Characters ..................................................... 6
        Character Strings ....................................................... 7
        DBCS and Field Length .................................................. 8
    Naming Conventions .................................................................... 8
    Dictionary Extensions and Administration ....................................... 8
    Controlling Access to Dictionary Data ........................................... 10
    Transferring Data between Dictionaries ......................................... 10

Chapter 2. Controlling Access to Dictionary Definitions ............................. 13
    Defining Groups of Users ......................................................... 14
    Dictionary User Access Control .................................................. 14
        Defining Status Code Access ............................................. 14
        Defining Subject Category Access ..................................... 15
        Category Number and Name Pairs ..................................... 15
        Determining Category Authorization Requirements ............... 16
    User Access Rules .................................................................... 16
        Specifying Access Authorization for Relationships ............... 17
        Command Considerations .................................................. 18
        Display Form Considerations ............................................. 20
    Allowing Selective Use of Dictionary Commands .............................. 21
        Internally Generated Dictionary Commands .......................... 22
    Specifying Subject Name Default Qualifiers .................................. 23
    Establishing Dictionary User Access Control ................................. 24
    Establishing Command Groups .................................................... 25
        How Command Groups Are Defined .................................... 25
        Abbreviations Used in Command Groups ............................... 26
        Assigning Commands to Command Groups ............................ 27
    Defining Dictionary Users with Commands ..................................... 29
        Subject Names for Dictionary Users .................................... 29
        Using the ADD Command .................................................. 30
Example of Creating a User Profile with the ADD Command .......... 31
Access Profile Keywords .................................................. 31
Access Profile Syntax ....................................................... 35
Syntax for Category Records ................................................. 36
Examples of User Access Profiles .......................................... 38
  Security Administration Access Profile .................................. 38
  Developer Access Profile .................................................. 38
  Command User Access Profile ............................................ 39
  Language Preprocessor Profile ........................................... 39
Defining Dictionary Users with Display Forms .......................... 39
  Using the DDUSER Form .................................................... 39
    Accessing a DDUSER Form ............................................... 40
    Content ................................................................ 40
    Working with the Form .................................................... 41
  Using the USER DATA Form for Access Profiles ...................... 41
    Working with the Form .................................................... 42
    Use of a Model Access Profile ........................................... 43
  Customizing Header Forms ................................................ 44
Authorizing and Verifying Access Profiles .............................. 45
  Verifying Sign-On in Batch ............................................... 46
  Verifying Sign-On Online ................................................. 46
  Verifying Category-Number Pairs ........................................ 46
  Maintaining Access Profiles ............................................. 46
    Reports for DDUSER Subjects ............................................ 47
  Allowing Users to Define Their Own Access Profiles ................ 47

Chapter 3. Planning for Dictionary Extensions and Customization .......... 49
Installation-Defined Dictionary Categories and Relationship-Types ........ 50
  Subjects and Subject Categories ....................................... 50
  Relationships and Relationship-Types ................................ 51
  Attributes of a Subject or Relationship ................................ 54
  How Customized Entities Are Defined .................................. 55
    The Installation Process ................................................ 57
    Restrictions for Deleting Installed Entities ....................... 57
    Modifying Installed Entities .......................................... 58
Validation Routines ......................................................... 58
  Planning for Name Validation Routines ................................ 59
  Planning for Attribute Value Validation Routines ................... 60
General Requirements for Validation Routines .......................... 61
Understanding Dictionary Structures ..................................... 61
  Representing a Dictionary Structure ................................... 62
  Understanding Structure-Type Usage .................................... 63
  Defining Superior and Subordinate Categories ........................ 64
  Understanding Processing Action ....................................... 64
  Using Dictionary Structure-Types ...................................... 66
  The Default Structure-Type .............................................. 68
Accessing the Dictionary Data with User Programs ..................... 70
  Retrieving Attribute Data from the Dictionary ....................... 71
  Retrieving Other Dictionary Data ...................................... 71
  User Program Output ..................................................... 72
  Communicating with a User Program ................................... 72
User Program Characteristics .............................................. 72
Dictionary-Supplied Programming Aids .................................. 73
Example of a Relationship-Type GUIDE Report .......................... 126
FORMAT Reports .................................................................... 127
How Extensibility Data Is Stored ........................................... 127
Storage for Attribute Values .................................................. 128
Storage of Validation Values ................................................. 129
CATEGORY and RELTYPE FORMAT Reports ......................... 130
Example of a Subject Category FORMAT Report ...................... 131
Example of a Relationship-Type FORMAT Report ..................... 132
Optimizing Attribute Data Storage ....................................... 133

Chapter 8. Modifying Installed Category and Relationship-Type Definitions ............................................................... 135
Planning for Different Types of Changes ................................. 136
Changing a Category's Definition Operands ............................. 136
Changing Attributes Specified for Subjects in the Category ........ 137
Changing a Relationship-Type's Definition Operands ................ 138
Changing Attributes Specified for Instances of the Relationship-Type 140
Procedure for Reinstalling a Category or Relationship-Type ........ 140
Creating a Replacement Definition ....................................... 141
Installing the Replacement Definition .................................... 142
Rules and Restrictions .......................................................... 142
Changing a Category's Alias Names ....................................... 142
Changing Name Validation Operands for a Category ................. 143
Changing a Relationship-Type's Alias Names ......................... 143
Changing Categories Participating in a Relationship-Type ........... 143
Changing the Directed Attribute for a Relationship-Type .......... 143
Changing Formatting Requirements for an Attribute ............... 144
Changing Validation Specifications for an Attribute ................ 144
The Reinstallation Process .................................................... 145
Performance Considerations .................................................. 146
Data Conversion ................................................................... 146
DATATYPE Operand ............................................................. 147
DATAMAX Operand ............................................................. 147
REPEAT Operand .................................................................. 148
Data Not Converted to Replacement Definition ....................... 149
Summary of Reinstallation Steps ............................................ 149
Example of a Reinstalled Category Definition ......................... 151
Installation Processing Report for a Reinstalled Category Definition 155
GUIDE Report for a Reinstalled Category Definition ................ 158
Example of a Reinstalled Relationship-Type Definition ............... 159
Installation Processing Report for a Reinstalled Relationship-Type ..... 162
GUIDE Report for a Reinstalled Relationship-Type Definition ........ 164

Chapter 9. Display Forms for Customized Entities ....................... 165
Extensibility Control Information Categories and the Header Form 166
Using Display Forms for CATEGORY Subjects .......................... 168
The EXTENSIBILITY SUBJECT Form for a CATEGORY Subject ........ 168
The CATEGORY-ATTRIBUTES Display Form .......................... 169
The CATEGORY-RELTYPES Display Form ............................... 170
Using Display Forms for RELTYPE Subjects ............................ 171
The EXTENSIBILITY SUBJECT Form for a RELTYPE Subject ........ 171
The RELTYPE-ATTRIBUTES Display Form .............................. 172
Using Display Forms for ATTRTYPE Subjects .......................... 173
The EXTENSIBILITY SUBJECT Form for an ATTRTYPE Subject . . . . 173
Considerations for SUBJECT-RELATIONSHIPS Forms ................. 174

Chapter 10. Validation Routines ............................................. 175
Validation Routine Processing ............................................. 176
Requirements for Validation Routines ..................................... 176
Communication Area for Attribute Value Validation ..................... 176
Communication Area for Name Validation .................................. 177
Program Entry Points ....................................................... 178
Validation Return Codes and Messages ..................................... 179
Installing Validation Routines ............................................... 180
Link-Editing Requirements .................................................. 180
Link-Editing Validation Routine for COBOL ............................... 180
Link-Editing Validation Routines for PL/I ................................. 181
Link-Editing Validation Routines for Assembler ......................... 181
Managing Validation Libraries ............................................. 182

Chapter 11. Defining Structure-Types ...................................... 183
Defining a STRTYPE Subject Name ......................................... 183
STRTYPE Definition Operands .............................................. 184
Using Commands with Structure-Type Definitions ....................... 185
Adding and Updating a Definition ....................................... 185
Using the COPY Command .................................................. 185
Deleting a Structure-Type Definition ...................................... 186
Relating Relationship-Types to a Structure-Type ......................... 186
Using Relationship Commands with Structure-Type Subjects ........... 186
Defining Processing Options for Relationship Types in a Structure Type . 188
    Designating a Superior Category ....................................... 188
    Processing Direction Options ....................................... 189
    Required Operand Values .......................................... 189
Summary of Structure-Type Characteristics ................................ 190
An Example of Structure-Type Definition ................................ 190
Reporting Structure-Type Development ................................... 192
Structure-Type Definition Reports ....................................... 192
Example of a Subject-Specific STRTYPE Report ......................... 192
GUIDE Report for Structure Types ...................................... 194
Example of a Structure-Type GUIDE Report ................................ 195
Installing Structure-Type Definitions ................................... 195
    The INSTALL STRTYPE Command ................................... 197
    Validations Performed by the INSTALL Command for a Structure-Type . 198
    Report for Installing a Structure-Type ............................ 199
Procedure for Changing an Installed Structure-Type ..................... 200
Display Forms for Defining Structure-Types .............................. 200
    The Extensibility Subject Form for a STRTYPE Subject .......... 201
    The STRTYPE-RELTYPES Display Form ............................. 202
The Default Structure-Type ............................................. 203
    Modifying the DEFAULT Structure-Type ............................ 206
    Procedure for Changing the DEFAULT Structure-Type .......... 207
    Alterations to the DEFAULT STRTYPE Subject .................... 207
Structure-Type Considerations ........................................... 207

Chapter 12. Writing Programs to Access Dictionary Data .............. 209
How a User-Written Program Operates ................................... 209
How to Invoke a User-Written Program .................................. 212

Contents xix
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to Write a Program to Access Dictionary Data</td>
<td>212</td>
</tr>
<tr>
<td>The Program-with-Dictionary Interface</td>
<td>213</td>
</tr>
<tr>
<td>Requesting Dictionary Access</td>
<td>214</td>
</tr>
<tr>
<td>Program Completion</td>
<td>214</td>
</tr>
<tr>
<td>Access Authorization Checking</td>
<td>214</td>
</tr>
<tr>
<td>User-Program Restrictions</td>
<td>214</td>
</tr>
<tr>
<td>Examples of Programs That Access the Dictionary</td>
<td>215</td>
</tr>
<tr>
<td>Retrieving Dictionary Data</td>
<td>215</td>
</tr>
<tr>
<td>Sending Output Messages via the Dictionary</td>
<td>216</td>
</tr>
<tr>
<td>Data Areas for Dictionary-User Program Communication</td>
<td>216</td>
</tr>
<tr>
<td>Using the Control Area (PACA)</td>
<td>217</td>
</tr>
<tr>
<td>Using the Data Area (PADA)</td>
<td>218</td>
</tr>
<tr>
<td>Using the Format Table (PADAFT)</td>
<td>218</td>
</tr>
<tr>
<td>Coding Options for the PADAFT</td>
<td>218</td>
</tr>
<tr>
<td>Verify and Execute Option: Coding Your Own PADAFT</td>
<td>219</td>
</tr>
<tr>
<td>Build and Execute Option: Using a Master PADAFT with Retrieval Requests</td>
<td>219</td>
</tr>
<tr>
<td>Prebuild Option: Modifying a Master PADAFT before Retrieval</td>
<td>220</td>
</tr>
<tr>
<td>An Example of Program Access Control Block Usage</td>
<td>221</td>
</tr>
<tr>
<td>Retrieving Data</td>
<td>224</td>
</tr>
<tr>
<td>Retrieving Attributes of Subjects and Relationships</td>
<td>224</td>
</tr>
<tr>
<td>Retrieving Lists of Data</td>
<td>225</td>
</tr>
<tr>
<td>The Format Table (PADAFT) for a List-Type Request</td>
<td>226</td>
</tr>
<tr>
<td>The Control Area (PACA) for a List-Type Request</td>
<td>226</td>
</tr>
<tr>
<td>Positioning within a List</td>
<td>227</td>
</tr>
<tr>
<td>Locating Instances of the List in the Data Area (PADA)</td>
<td>228</td>
</tr>
<tr>
<td>Retrieval Requests</td>
<td>228</td>
</tr>
<tr>
<td>Retrieving Information about Fast Path Subjects</td>
<td>229</td>
</tr>
<tr>
<td>Combining Several Types of Retrieval</td>
<td>230</td>
</tr>
<tr>
<td>How Each Retrieval Request Is Described in This Section</td>
<td>231</td>
</tr>
<tr>
<td>Retrieving a Subject’s Attribute Data (RSA Call)</td>
<td>232</td>
</tr>
<tr>
<td>Retrieving a Relationship’s Attribute Data (RRA Call)</td>
<td>235</td>
</tr>
<tr>
<td>Retrieving Text Data (RDSC, RUDn, RPLI Calls)</td>
<td>238</td>
</tr>
<tr>
<td>Description Data</td>
<td>239</td>
</tr>
<tr>
<td>User Data Text</td>
<td>240</td>
</tr>
<tr>
<td>PL/I Data Text</td>
<td>240</td>
</tr>
<tr>
<td>Retrieving a Subject’s Aliases (RA Call)</td>
<td>243</td>
</tr>
<tr>
<td>Retrieving Subject Names in a Category (RSN and RSAN Calls)</td>
<td>246</td>
</tr>
<tr>
<td>Retrieving Relationships for a Subject (RSR, RSRC, RSRK Calls)</td>
<td>250</td>
</tr>
<tr>
<td>Retrieving a List of Relationship-Types Allowed for a Category (RCR Call)</td>
<td>262</td>
</tr>
<tr>
<td>Format Table (PADAFT) for Relationship-Type Retrieval Requests</td>
<td>263</td>
</tr>
<tr>
<td>Retrieving a List of Relationship-Types Defined for an Installed Structure-Type (RSTR Call)</td>
<td>266</td>
</tr>
<tr>
<td>Format Table (PADAFT) for Structure-Type Retrieval Requests</td>
<td>267</td>
</tr>
<tr>
<td>User-Program Output</td>
<td>271</td>
</tr>
<tr>
<td>Coding a COBOL Program to Access Dictionary Data</td>
<td>274</td>
</tr>
<tr>
<td>Coding an Entry Point</td>
<td>274</td>
</tr>
<tr>
<td>Specifying a Parameter String Area</td>
<td>274</td>
</tr>
<tr>
<td>Specifying the Program Access Data Area and Control Blocks</td>
<td>275</td>
</tr>
<tr>
<td>Coding a CALL for Data Retrieval or Output</td>
<td>275</td>
</tr>
<tr>
<td>Returning to the Dictionary</td>
<td>276</td>
</tr>
<tr>
<td>Link-Editing the COBOL Program</td>
<td>277</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>SYSDEF WITH TRANSACTION</td>
<td>453</td>
</tr>
<tr>
<td>TRANSACTION WITH PSB</td>
<td>453</td>
</tr>
<tr>
<td>Extensibility Control Information Categories</td>
<td>454</td>
</tr>
<tr>
<td>ATTRTYPE Category</td>
<td>454</td>
</tr>
<tr>
<td>CATEGORY Category</td>
<td>454</td>
</tr>
<tr>
<td>RELTYPE Category</td>
<td>455</td>
</tr>
<tr>
<td>STRTYPE Category</td>
<td>455</td>
</tr>
<tr>
<td>Extensibility Control Information Relationship-Types</td>
<td>456</td>
</tr>
<tr>
<td>CATEGORY HAS ATTRTYPE</td>
<td>456</td>
</tr>
<tr>
<td>RELTYPE HAS ATTRTYPE</td>
<td>456</td>
</tr>
<tr>
<td>STRTYPE CONTAINS RELTYPE</td>
<td>457</td>
</tr>
<tr>
<td>Appendix B.7. Program Access Macro Support</td>
<td>458</td>
</tr>
<tr>
<td>Appendix B.8. Special Topics for Attribute Retrieval</td>
<td>462</td>
</tr>
<tr>
<td>Types of Attributes</td>
<td>462</td>
</tr>
<tr>
<td>Sequence Values</td>
<td>464</td>
</tr>
<tr>
<td>Repeating Attributes</td>
<td>464</td>
</tr>
<tr>
<td>Repeating Groups of Attributes</td>
<td>465</td>
</tr>
<tr>
<td>Appendix C. Format Specifications for EXPORT Command Output Streams</td>
<td>467</td>
</tr>
<tr>
<td>Command Formats</td>
<td>467</td>
</tr>
<tr>
<td>ADD, CHANGE_IN, and DELETE_DATA Command Formats</td>
<td>468</td>
</tr>
<tr>
<td>ADD_RELATIONSHIP and CHANGE_RELATIONSHIP_DATA Command Formats</td>
<td>470</td>
</tr>
<tr>
<td>INSTALL Command Format</td>
<td>471</td>
</tr>
<tr>
<td>Batch Forms Formats</td>
<td>472</td>
</tr>
<tr>
<td>Date and Time-Stamp Record Format</td>
<td>473</td>
</tr>
<tr>
<td>Abbreviations</td>
<td>475</td>
</tr>
<tr>
<td>Glossary</td>
<td>477</td>
</tr>
<tr>
<td>Index</td>
<td>483</td>
</tr>
</tbody>
</table>
Figures

1. Category Authorization Requirements for Certain Dictionary Commands 16
2. SECRET Specifications and Relationship Authorization Combinations 18
3. Dictionary Assigned Abbreviations with the DBDGSGRP Macro 26
4. Dictionary-Supplied Command Groups 27
5. Category Record Format 36
6. DDUSER Form 40
7. USER DATA Form with Access Profile 42
8. USER DATA Form with Model Access Profile 44
9. Customized Header Form after Sign-On 45
10. Example of Installation-Defined Categories and Relationships to CATEGORY, RELTYPE, and ATTRTYPE Categories 56
11. Summary of Commands or Functions for Name Validation Processing 59
12. An Example of a Structure 61
13. A Partial Structure-Type of Standard Categories 62
14. An Installation-Defined Structure-Type 63
15. MAIN1 Structure-Type 67
16. TASK Structure-Type 67
17. Structures Mapped by the TASK and MAIN1 Structures 68
18. Hierarchy of Categories Defined in the DEFAULT Structure-Type 69
19. Structure-Type for a Non-DL/I Application 70
20. Example of ATTRTYPE Subject-Specific Report 83
21. Summary of Definition Operands for an Installation-Defined Category 94
22. Example of a Subject-Specific CATEGORY Subject Report 97
23. Summary of Definition Operands for an Installation-Defined Relationship-Type 106
24. Example of a Relationship-Type Subject-Specific Report 109
25. Example of a Subject Category Installation Report 119
26. Example of a Relationship-Type Installation Report 120
27. Example of a Subject Category GUIDE Report 124
28. Example of a Relationship-Type GUIDE Report 127
29. Example of a Subject Category FORMAT Report 132
30. Example of a Relationship-Type FORMAT Report 133
31. Example of FORMAT Report with Optimized Storage 134
32. Summary of Definition Operands for an Installation-Defined Category 137
33. Summary of Definition Operands for an Installation-Defined Relationship-Type 139
34. Allowable Data Type Conversions 144
35. Sample Installation Processing Report for a Reinstalled Category 155
36. Example of a Category GUIDE Report after a Reinstall Process 158
37. Sample Installation Processing Report for a Reinstalled Relationship-Type 162
38. Example of a Relationship-Type GUIDE Report after a Reinstallation Process 164
39. Hierarchy of Display Forms for Extensibility Subjects .......... 167
40. EXTENSIBILITY SUBJECT Form for a CATEGORY Subject ....... 168
41. The CATEGORY-ATTRIBUTES Display Form .................. 169
42. The CATEGORY-RELTYPES Display Form ...................... 170
43. EXTENSIBILITY SUBJECT Form for a RELTYPE Subject ......... 171
44. The RELTYPE-ATTRIBUTES Display Form ...................... 172
45. EXTENSIBILITY SUBJECT Form for an ATTRTYPE Subject ...... 173
46. Format of Communication Area for Attribute Value Validation .. 177
47. Format of Communication Area for Name Validation .......... 178
48. Example of a Subject-Specific STRTYPE Subject Report ....... 193
49. Example of a Structure-Type GUIDE Report .................... 195
50. Example of a Structure-Type Installation Report ............... 199
51. Extensibility Form for a STRTYPE Subject ................. 201
52. The STRTYPE-RELTYPES Display Form ......................... 202
53. Dictionary-Supplied Relationships .............................. 204
54. GUIDE Report for Dictionary-Supplied Default Structure-Type ... 205
55. How a User-Written Program Operates in the Dictionary Environment 210
56. How Program Access Data Areas Are Related .................. 217
57. Example of PACA before and after a Retrieval Request .......... 221
58. Example of PADAFT before and after a Retrieval Request ........ 222
59. Required PACA Fields for an RSA Request ..................... 232
60. PACA Fields Set by the Dictionary on Return from an RSA Request . 233
61. Summary of Return Codes for an RSA Retrieval Request ........ 234
62. Required PACA Fields for an RRA Request ..................... 235
63. PACA Fields Set by the Dictionary on Return from an RRA Request . 236
64. Summary of Return Codes for an RRA Retrieval Request ........ 237
65. Required PACA Fields for a Text Data Request ................. 238
66. Optional Field to Control Last Text Line Retrieved .......... 240
67. PACA Fields Set by the Dictionary on Return from a Text Data Retrieval .................. 241
68. Summary of Return Codes for a Text Retrieval Request ......... 242
69. Required PACA Fields for an RA Request ...................... 243
70. Optional Field to Control Last Alias Name Retrieved .......... 244
71. PACA Fields Set by the Dictionary on Return from an RA Request . 244
72. Summary of Return Codes for an RA Retrieval Request ........ 245
73. Required PACA Fields for an RSN or RSAN Request ............ 246
74. Optional Field to Control Last Subject Name Retrieved ........ 248
75. PACA Fields Set by the Dictionary on Return from an RSN or RSAN Request .................. 248
76. Summary of Return Codes for an RSN or RSAN Retrieval Request . 249
77. Required PACA Fields for an RSR Request ..................... 251
78. Optional Fields to Control Starting Position for RSR Requests ........ 252
79. PACA Fields Set by the Dictionary on Return from an RSR Request . 253
80. Summary of Return Codes for an RSR Retrieval Request ......... 254
81. Required PACA Fields for an RSRC Request .................... 255
82. Optional Fields to Control Starting Position for RSRC Requests ........ 256
83. PACA Fields Set by the Dictionary on Return from an RSRC Request . 256
84. Summary of Return Codes for an RSRC Retrieval Request ......... 257
85. Required PACA Fields for an RSRK Request ..................... 258
86. Optional Fields to Control Starting Position for RSRK Requests ........ 259
87. PACA Fields Set by the Dictionary on Return from an RSRK Request . 259
88. Summary of Return Codes for an RSRK Retrieval Request ......... 260
89. Required PACA Fields for an RCR Request ..................... 262
90. PACA Fields Set by the Dictionary on Return from an RCR Request . 264
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>91.</td>
<td>Optional Fields to Control Starting Position for RCR Requests</td>
</tr>
<tr>
<td>92.</td>
<td>Summary of Return Codes for an RCR Retrieval Request</td>
</tr>
<tr>
<td>93.</td>
<td>Required PACA Fields for an RSTR Request</td>
</tr>
<tr>
<td>94.</td>
<td>Optional Field to Control Starting Position for RSTR Requests</td>
</tr>
<tr>
<td>95.</td>
<td>Optional Field to Control Name of Last Relationship-Type Retrieved</td>
</tr>
<tr>
<td>96.</td>
<td>PACA Fields Set by the Dictionary on Return from an RSTR Request</td>
</tr>
<tr>
<td>97.</td>
<td>Summary of Return Codes for an RSTR Retrieval Request</td>
</tr>
<tr>
<td>98.</td>
<td>Required PACA Fields for an Output Request</td>
</tr>
<tr>
<td>99.</td>
<td>PACA Field Set by the Dictionary on Return from an Output Request</td>
</tr>
<tr>
<td>100.</td>
<td>Summary of Return Codes for an Output Request</td>
</tr>
<tr>
<td>101.</td>
<td>Parameter Area for Program Access (COBOL)</td>
</tr>
<tr>
<td>102.</td>
<td>Return Codes Available for COBOL Program Access</td>
</tr>
<tr>
<td>103.</td>
<td>Parameter Area for Program Access (PL/I)</td>
</tr>
<tr>
<td>104.</td>
<td>Return Codes Available for PL/I Program Access</td>
</tr>
<tr>
<td>105.</td>
<td>Parameter Area for Program Access (Assembler)</td>
</tr>
<tr>
<td>106.</td>
<td>Return Codes Available for Assembler Program Access</td>
</tr>
<tr>
<td>107.</td>
<td>Components of Dictionary Preprocessing</td>
</tr>
<tr>
<td>108.</td>
<td>Portion of a COBOL Source Program with an Inclusion Statement</td>
</tr>
<tr>
<td></td>
<td>Expanded</td>
</tr>
<tr>
<td>109.</td>
<td>Example of SYSPRINT Output Produced by the Language Preprocessor</td>
</tr>
<tr>
<td>110.</td>
<td>Options Stored in the Default Options Table</td>
</tr>
<tr>
<td>111.</td>
<td>Language Preprocessor User Exit: COBOL Parameter Area</td>
</tr>
<tr>
<td>112.</td>
<td>Language Preprocessor User Exit: PL/I Parameter Area</td>
</tr>
<tr>
<td>113.</td>
<td>Language Preprocessor User Exit: Assembler Parameter Area</td>
</tr>
<tr>
<td>114.</td>
<td>Where SETCLIST Command Obtains Operand Information</td>
</tr>
<tr>
<td>115.</td>
<td>GUIDE Report for Category Used for Command Lists</td>
</tr>
<tr>
<td>116.</td>
<td>Comparison of Equivalent Command and Processor Default Options</td>
</tr>
<tr>
<td>117.</td>
<td>Data Extraction Choices Used with the EXPORT Command</td>
</tr>
<tr>
<td>118.</td>
<td>EXPORT Command Destination in Batch Mode with Output Routing</td>
</tr>
<tr>
<td>119.</td>
<td>EXPORT Command Destination in Online Mode with Output Routing</td>
</tr>
<tr>
<td>120.</td>
<td>EXPORT Command Destination in Batch Mode without Output Routing</td>
</tr>
<tr>
<td>121.</td>
<td>EXPORT Command Destination in Online Mode without Output Routing</td>
</tr>
<tr>
<td>122.</td>
<td>Example of Generated Output Commands from an EXPORT Command</td>
</tr>
<tr>
<td>123.</td>
<td>Example of an EXPORT Processing Report</td>
</tr>
<tr>
<td>124.</td>
<td>Example of PREVIEW Processing Report</td>
</tr>
<tr>
<td>125.</td>
<td>Example of PREVIEW Processing Report with User-Generated Stream</td>
</tr>
<tr>
<td>126.</td>
<td>Return Codes for Retrieval and Output Requests</td>
</tr>
<tr>
<td>127.</td>
<td>Return Codes for PADAFT Entries</td>
</tr>
<tr>
<td>128.</td>
<td>PATYPE: Request-Type Character Codes</td>
</tr>
<tr>
<td>129.</td>
<td>PAACCESS: Character Codes for Positioning in a List</td>
</tr>
<tr>
<td>130.</td>
<td>PADAOP: Character Codes for Blanking Unretrieved Fields</td>
</tr>
<tr>
<td>131.</td>
<td>PAFTU: Character Codes for PADAFT Usage</td>
</tr>
<tr>
<td>132.</td>
<td>PACCCTRL: Codes for Format Control of Output Lines</td>
</tr>
<tr>
<td>133.</td>
<td>PAFDTYP: Character Codes for Attribute Types</td>
</tr>
<tr>
<td>134.</td>
<td>PAFFUNC: Retrieve or Ignore Code</td>
</tr>
<tr>
<td>135.</td>
<td>DATABASE Category</td>
</tr>
<tr>
<td>136.</td>
<td>ELEMENT Category</td>
</tr>
<tr>
<td>137.</td>
<td>PCB Category</td>
</tr>
<tr>
<td>138.</td>
<td>SEGMENT Category</td>
</tr>
<tr>
<td>139.</td>
<td>SYSTEM Category</td>
</tr>
</tbody>
</table>
Chapter 1. Overview of Dictionary Administration

Dictionary administration is the task of administering the content of the Dictionary data bases and planning and establishing procedures for Dictionary use. The major tasks are concerned with understanding what data is to become part of that data base, who is going to use the data, and how the data is to be maintained. Controlling the data effectively and making it available are key to the success of the endeavor.

Some tasks, such as the incorporation of source information from libraries, are done in a specific sequence; others are accomplished in response to the requirements of users.

How Administration Relates to Other Dictionary Tasks

This section describes the major activities associated with installing and using the OS/VS DB/DC Data Dictionary. These activities can be divided as follows:

- Planning for Dictionary installation and operation
- Installing the Dictionary programs and data bases
- Administering the ongoing use and growth of the Dictionary
- Using the Dictionary
- Diagnosing problems

Planning for Dictionary Installation

Planning for installation requires a thorough understanding of the extent of Dictionary usage and the operating environment where the Dictionary is to be installed. Such planning is a general task that entails making any necessary preparations for the entry of the Dictionary into the data processing support in addition to such tasks as:

- Coordinating system generation and terminal usage
- Providing operations procedure and recovery information
- Notifying operations of system device, data set, and JCL use
This information is covered in *OS/VS DB/DC Data Dictionary Installation Guide.*

**Installing the Dictionary**

Installing a Dictionary is the process of preparing it for use and defining necessary resources. This is essentially the implementation of decisions made by the system administrators. Among the subtasks installation comprises are:

- Establishing the hardware and software environment
- Building the Dictionary distribution libraries
- Loading the initial Dictionary data bases
- Coding macros for system generation
- Tailoring and performing system generation
- Running installation verification procedures
- If migrating from an earlier release, running regression tests

All information about initially installing the Dictionary into the online operating environment of CICS/OS/VS, IMS/VS, and the batch Dictionary installation, is given in *OS/VS DB/DC Data Dictionary Installation Guide.*

**Administering the Dictionary**

Administering the ongoing Dictionary operation and obtaining productive use require an awareness of activity in all areas. Administrative tasks are discussed in later sections.

**Using the Dictionary**

The Dictionary can be used by many different types of users with differing skill levels. The Dictionary programs operate in both online and batch mode. The content of the Dictionary data bases is accessible with various reporting and data extraction methods in addition to data entry and manipulation. Some facilities provided by the Dictionary are more appropriate, though not exclusively, for administrative action. There are two major groups of usage.

**General Use**

- Using Dictionary commands, batch forms, and display forms for the basic entry, manipulation, and reporting of Dictionary data
- Using facilities to make the Dictionary efficient: output routing, reports, stacks, command lists, and language data structure recalculation
- Using facilities that perform more complex or extensive data manipulation: input from libraries, copying, or deleting collections of data
Information about general use is contained in *OS/VS DB/DC Data Dictionary Applications Guide*.

**Administrative Use**

Administrative use involves planning and establishing procedures for Dictionary use as well as administering the contents of the Dictionary data bases. See "Dictionary Administration Tasks."

Information about administrative use is given in subsequent chapters of this publication.

As a reference manual for all the above types of usage, users need *OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference*. This publication is meant to be used in combination with the above two guides. Users also need *OS/VS DB/DC Data Dictionary Messages and Codes* to help interpret diagnostics evoked by their Dictionary usage.

**Diagnosis**

The task of diagnosis covers identifying and describing errors in the Dictionary licensed program. Identifying a problem involves:

- Using error messages, trace information, and two manuals, *OS/VS DB/DC Data Dictionary Diagnosis: Guide* and *OS/VS DB/DC Data Dictionary Diagnosis: Reference*, to identify the problem by 1) type and 2) area of failure within the program logic

- Constructing a keyword string that describes the problem, in order to search the software support data base for a description of a similar problem

- Reporting the problem for correction, if necessary

Problem correction is usually the responsibility of IBM Central Programming Group personnel. Administrative personnel need to be aware of any user constraints during the above problem resolution.

**Dictionary Administration Tasks**

The following summarizes the tasks of Dictionary Administration. You can see that there is a need to initially promote the Dictionary and to actively supervise the growth of the data collection. As the data stored in the Dictionary data bases builds in volume and interrelationship, so does the potential value. One important activity is to keep in touch with the groups of users and plan ahead for their support. The tasks described in this publication are:

- Establishing naming conventions for Dictionary subjects and other requirements for documentation standards

- Defining and installing customized entities to extend the contents of the Dictionary data beyond the standard categories and relationships
• Responding to processing control requirements for structure-dependent processing

• Controlling access to Dictionary data

• Understanding how Dictionary can be used as an integral part of application development in the CICS/OS/VS or IMS/VS environments through the Language Preprocessor

• Transferring data between Dictionaries

The Dictionary administration task described in *OS/VS DB/DC Data Dictionary Installation Guide* shows how to set up operational procedures.

Other Dictionary administration tasks you might want to consider are:

• Investigating how the Dictionary can be made part of data administration or how its use can be part of application or data base design

• Understanding how the Dictionary can be used with other licensed programs, such as the Application Development Facility and GIS/VS

• Identifying sources of Dictionary data

• Determining requirements for data collection

• Acting as an “Information Center” for Dictionary services and distributing information about customized entities

• Training personnel in Dictionary use

• Monitoring interactive response time and the growth of the Dictionary data bases

**Characterizing Groups of Dictionary Users**

There are three general groups of Dictionary users:

• **Dictionary administrators**: personnel who are responsible for designing, maintaining, and customizing the content of Dictionary data bases. The Dictionary administrator establishes security practices, and is generally responsible for providing service for the active users.

These personnel often have responsibility for data administration in an installation. As such, they are concerned with data as a resource, documenting an application's use of data, naming conventions, and access to data resources. For example, they may develop routines that validate data entered in the Dictionary.

Often this group includes database administration staff who are supporting database design and require control over such definitions and related application program descriptors.
• **Active users:** the employees who use the Dictionary as a tool while maintaining and developing application programs and their related data bases. These users enter data into the Dictionary data bases, maintain Dictionary entries, extract language data structures, and extract other data (often in the form of reports).

• **General users:** personnel who refer to reports prepared by the data processing installation, and who may be occasional Dictionary users, or who may provide input data for the Dictionary. For example, such a user might access the Dictionary to find information about some specific data resource.

**Supporting Dictionary Administration**

An important aspect of the role that a Dictionary will play in your installation is the degree of control and standardization present. If you have an active data administration function, the Dictionary can be actively used to register new definitions of resources and help organize the categories of data for report purposes.

If your installation has a strong data base administrative structure, the Dictionary can be used for information about data base design and all those data processing resources that relate to the design, segments, and fields. The Dictionary enables you to keep several versions of data base definitions and helps protect the integrity of the production version. You can build the design incrementally and you have several ways to record additional documentation that is associated with that design.

**Supporting Dictionary Language Preprocessor Users**

A special group of users work with host compilers and can use language data structures extracted from the Dictionary. The Dictionary Language Preprocessor is a program that is provided as part of the Dictionary system. Although its use is optional, it can be a valuable service to your installation. It can be both an integral part of the development process and it can make a contribution to programmer productivity. Language data structures that have been stored in the Dictionary data bases can be made available to a variety of end users without the need for Dictionary education. With appropriate preparation, source language data structures stored within the Dictionary can be used by application developers when they are compiling and testing programs.

Obtaining a language data structure from the Dictionary helps ensure that the structure meets standards and is the approved version. In fact, with appropriate preparation of the Dictionary, you can ensure that the current version is automatically used without the need for any alteration of the source program.

Making use of the Language Preprocessor does not preclude the use of any existing source libraries (such as COPY/INCLUDE libraries) and does not require specialized source coding. The Language Preprocessor accepts as input a COBOL, PL/I, or Assembler source program. Source statements, in the host language, are recognized by the Dictionary language preprocessor as it scans the source program. The preprocessor statements are simple and their syntax is familiar. See Chapter 13, “Administering the Dictionary Language Preprocessor” on page 291 for more information on the Language Preprocessor.
Supporting Occasional Users

A Dictionary user is not always required to become an expert in the use of commands and other Dictionary procedures. The formation of an “Information Center” can help make the Dictionary support more responsive to the needs of the occasional user. An “Information Center” can help Dictionary users by providing:

- Information on the Dictionary Language Preprocessor and help with Dictionary message interpretation
- Interpretation of conventions used within Dictionary subject definitions
- Education for batch form use for bulk data collection
- Development of command lists for executing regular report requests
- Distribution of GUIDE reports and other support for customized entities

Double Byte Character Set Support

The Dictionary supports a Double Byte Character Set (DBCS) in which each character is represented in 2 bytes. In comparison, each EBCDIC character is represented in 1 byte.

For DBCS support to be available, a default value must be initialized when the Dictionary is installed. For more information on initializing the DBCS default value, refer to OS/VS DB/DC Data Dictionary Installation Guide. The appropriate hardware is also required for the Dictionary to support DBCS. The hardware requirements are described in OS/VS DB/DC Data Dictionary General Information Manual.

Some terminals and printers support currency symbols other than the $ (dollar sign). For example, the 5550 Multistation family supports the Y (yen symbol). When you enter data in the Dictionary, you must specify the currency symbol having the hexadecimal value of 5B. Generally, this is the currency symbol of your native country.

The Shift Characters

DBCS data is delimited by shift characters to separate it from EBCDIC data. The DBCS string is preceded by a shift-out (‘so’) character and followed by a shift-in (‘si’) character. The Dictionary reads anything inside the pair of ‘so’ and ‘si’ characters as 2-byte DBCS characters, and anything outside as 1-byte EBCDIC characters.

The examples in this publication use special symbols to represent the shift characters. The symbol ‘<’ represents the ‘so’ character. The symbol ‘>’ represents the ‘si’ character. These are only representations used in this text and are not the actual characters. The actual characters used have hexadecimal values of X ‘0E’ for ‘so’ and X ‘0F’ for ‘si’.
Character Strings

The Dictionary defines two types of strings that may include DBCS characters: a uniform string and a mixed string. Each type of string is validated against a specific set of rules.

A uniform string is a character string containing either EBCDIC characters or DBCS characters but not both. If the uniform string consists of DBCS characters, it must be enclosed in the 'so' and 'si' characters.

For a Dictionary entry to be a valid uniform string, you must enter it according to Name Rules. These are:

- Enter either EBCDIC or DBCS characters, but not both, for the value.
- Enclose a DBCS uniform string in 'so' and 'si' characters.
- Do not use a string that consists of only the 'so' and 'si' characters or that contains only DBCS blanks between the 'so' and 'si' characters.
- Do not embed DBCS or EBCDIC blanks in the value or use them as leading characters.
- You may use DBCS and EBCDIC blanks as trailing characters in the value. Note that each trailing DBCS blank in the value is converted to two trailing EBCDIC blanks. The 'si' character is then shifted to the left following the last DBCS character.
- Within the range of X'4241' to X'42FE' in DBCS, you can use only DBCS characters corresponding to the following EBCDIC characters:

  0 through 9

  A through Z (uppercase only)

  @ (at), # (pound sign), $ (dollar sign), . (period), / (slash),
  _ (underscore), - (hyphen)

A mixed string is a character string that may contain all EBCDIC characters, all DBCS characters, or a combination of EBCDIC and DBCS characters. Any portion of the string in DBCS must be enclosed within 'so' and 'si' characters.

For a Dictionary entry to be a valid mixed string, you must follow the Text Rules. These are:

- Use all EBCDIC characters, all DBCS characters, or a combination of EBCDIC and DBCS characters.
- Enclose any DBCS strings within the value in the 'so' and 'si' characters.
- If you embed EBCDIC blanks in the value or use them as leading characters, you must enclose the whole value in quotation marks. If you embed DBCS blanks in the value or use them as leading or trailing characters, you do not have to enclose the value in quotation marks.
DBCS and Field Length

The number of DBCS characters that can be entered for a value is not the same as the number of EBCDIC characters that can be entered for the same value. To calculate the number of DBCS characters that will fit in the same field, you must account for the 2 bytes required for each DBCS character and for the 'so' and 'si' characters.

For example, if a field is 31 bytes long, you can enter 31 EBCDIC characters. To calculate the number of DBCS characters that will fit, first subtract 2 bytes from the length for the 'so' and 'si' characters. In the example, 31 minus 2 leaves 29 bytes. Divide this number by the 2 bytes required for each DBCS character. This gives the number of DBCS characters that can be entered in the field. In the example, dividing 29 by 2 shows that 14 DBCS characters can be entered in a 31-byte field.

Naming Conventions

A major activity that the Dictionary supports is the implementation of naming conventions within the installation. Within the subject name design for Dictionary subjects is an approach to the development process and the need for control of redundancy in naming data processing resources.

You can provide routines to validate and either accept or reject:

- User names specified for subjects in any installation-defined category or any standard category
- Input values specified as attributes of a subject in an installation-defined category, or as relationship data for an instance of an installation-defined relationship-type

For more information on validation routines, see Chapter 10, “Validation Routines” on page 175.

Dictionary Extensions and Administration

A special feature of the Dictionary is the ability to define categories and relationship-types that are tailored to the data collection requirements of your installation. For example, in support of a set of application programs, you can record the current and past maintenance against those programs with such associated data as the number and types of changes, and the names of personnel who implemented the changes. An administrative task is to gather requirements for such types of data and make them operational in a controlled and responsive manner.
The development activities by which the scope of the Dictionary system can be enhanced are:

- The definition of installation-defined subject categories, and the further definition of relationship-types to exist between subjects in these categories, or between subjects in these categories and subjects in standard categories
- The definition of structure-types to control the processing performed when entire collections of interrelated data are being manipulated or reported
- The preparation of installation-written programs to retrieve Dictionary data and perform subsequent reporting or other processing

Support for Customized Entities

The routines and other parts of the Dictionary that allow you to create customized subject categories and relationship-types (that is, to “extend” the Dictionary) are collectively called the Extensibility facility.

Information on the Extensibility facility is divided into a progression of tasks:

- Chapter 3, “Planning for Dictionary Extensions and Customization” gives conceptual and planning information about customized entities. This will assist you in responding to requirements for such Dictionary extensions.
- Chapter 4, “Defining Attribute-Types” through Chapter 6, “Defining Customized Relationship-Types” and Chapter 9, “Display Forms for Customized Entities” explain how to define these customized entities.
- Chapter 7, “Installing Categories and Relationship-Types” and Chapter 8, “Modifying Installed Category and Relationship-Type Definitions” explain how extensibility definitions are installed and give guidance should you have to modify and reinstall any of those definitions.
- Chapter 11, “Defining Structure-Types” describes how to define and install structure-types to support processing requirements when working with collections of interrelated data.

Support for Installation-Written Programs

The content of the Dictionary data bases can be examined using the REPORT, STRUCTURE_REP, and SCAN commands, or with display forms. This data can also be retrieved by programs that are developed by your installation. Such programs can customize the content and format of individual reports. Alternatively, they could perform other processing on the retrieved data. For example, a program could validate a set of definitions for completeness.

The ability for installation-written programs to extract data and produce output using printer, punch, or terminal devices assigned to the Dictionary is provided by the Program Access facility.

Chapter 12, “Writing Programs to Access Dictionary Data” describes the programming requirements and data extraction facilities available with Program Access.
Controlling Access to Dictionary Data

As the Dictionary increases the scope of the data it contains, there is an increased demand for access to that data and, consequently, a need to control user access.

The Dictionary provides a way to establish authorization to access Dictionary data but tailors that authority to the needs of groups of users or individuals.

You partition groups of users so that they may access only a portion of the data contained in the Dictionary data bases. You specify an "access profile," using either Dictionary commands or display forms. Procedures for making Dictionary user access control active, and for verifying the user access profiles, are not complex.

Information on user access control support can be found in Chapter 2, "Controlling Access to Dictionary Definitions" on page 13. This chapter tells you about:

- Planning for the access requirements of groups of users or individuals and the procedure for making access control active
- Defining access profiles with commands and display forms
- Validating those profiles

Transferring Data between Dictionaries

With the growth of Dictionary use and the need to distribute information to other installations besides the central installation, there can be a need for independently operated Dictionaries.

Facilities are available to extract data from a central or source Dictionary and transfer that data to a receiving Dictionary. The extracted data is in a form that can be readily understood, edited, and previewed for its effect on the receiving Dictionary.

This support for the task of transferring data is collectively called the Portability facility. It allows you to extend the availability of Dictionary definitions beyond a single data base:

- A receiving Dictionary can be initialized with data extracted from a centrally maintained Dictionary.
- Definitions that are independently developed within an installed Dictionary can be incorporated into a central or receiving Dictionary.
Information on the Portability facility can be found in Chapter 14, "Transferring Dictionary Data" on page 349. This chapter tells you:

- How the **EXPORT** command can extract one or more subject definitions, or even whole collections of interrelated subjects, and generate a Dictionary output stream containing data in a Dictionary command format

- How the **PREVIEW** command allows you to examine the generated command stream's impact on the receiving Dictionary without updating it

- How the data transfer is completed by using the generated stream as batch input to the receiving Dictionary
Chapter 2. Controlling Access to Dictionary Definitions

This chapter discusses how you partition groups of users so that they may access only a portion of the data contained in the Dictionary data bases. It specifies how to define an “access profile,” using either Dictionary commands or display forms. Procedures for activating Dictionary user access control and verifying the user access profiles are included.

Access control consists of:

- **USE** authorization, which pertains only to categories. If a user has “use” authorization to a category, the user may view or update subjects in that category. For additional information, see the discussion in “Defining Status Code Access” on page 14.

- **VIEW** and **UPDATE** authorization, which pertains to status codes. If a user has view authorization to a status code, the user may view subjects with that status code. If a user has update authorization to a status code, the user may update subjects with that status code. Update authorization also implies view authorization.

- **Command** authorization, which pertains to Dictionary commands. A user can be authorized to use only certain commands in online and batch modes.

Dictionary access control is established for a user as follows:

- Identify the individual user with a subject in the DDUSER category.

- Define an access profile for the user, added as subject User Data, specifying:
  - A list of categories the user can access
  - A list of status codes for which the user can update subjects in those categories
  - A list of status codes, in addition to the update codes, for which the user can view subjects in those categories
  - A list of commands available to the user
  - A list of subject name defaults specific to the user
Defining Groups of Users

Generally, installations will have different groups of personnel requiring access to the information contained within the Dictionary. Only selected users need to be able to update the information. If Dictionary authorization checking is inactive, users are allowed unlimited access to data and unlimited ability to modify the data. Usually, some information is more confidential than others, and therefore should not be accessible by all users. Similarly, "update" capability usually should be limited to a smaller group of personnel. Additionally, most users do not need to use all categories of data stored in the Dictionary.

The status code portion of the Dictionary subject name can be used to associate the subject with other subjects in the same category, or in different categories. The status code, thus, partitions the Dictionary subjects in such a manner that each group of users can work with its own subjects. The Dictionary status codes considered to be test levels are: '0' through '9' and 'A' through 'T', with the exception of 'P', which is reserved for production level. It is expected that the definition of subjects in "test" status will be made by several groups of users, but only a limited number of users will control subjects in production status.

Dictionary access control enables you to specify whether a user can update subject definitions with given status codes, and, as a separate specification, whether the user can view (read only) subject definitions with given status codes. Consequently, a user is said to have update authorization or view authorization for a subject and its associated data.

Dictionary User Access Control

The Dictionary enables you to define individual access profiles. Each profile stipulates the categories and the chosen status codes that may be accessed for that particular user identification. The profile also specifies whether the access allows update of the Dictionary data or only allows the data to be viewed. The user can also be authorized to use all or a set of Dictionary commands. To have these kinds of control for individual users, you must have Dictionary authorization checking active. The access profiles must, of course, be defined and maintained.

Defining Status Code Access

In an access profile for an individual user, you must declare both the status codes with update authorization and the status codes with view authorization. A user with update authorization automatically has view authorization to that same set of status codes. In the online use of display forms, the list of authorized status codes for view and update appears on the Sign-On Header form. The list of status codes is chosen from 'A' through 'T', '0' through '9', and '***'.

The status code '***' can occur for subjects in extensibility control information categories (the ATTRTYPE, CATEGORY, RELTYPE, and STRTYPE) after they have been installed. Those personnel who will be using the INSTALL command must have update authorization for subjects in extensibility control information categories with 'P' and '***' status.
One status code must be used for the category (DDUSER) that contains all the names and access profiles of Dictionary users. This status is assigned as one of the defaults for installation, and must appear in the access profiles of security administration. In addition, this status code can be used by other subject categories.

**Defining Subject Category Access**

Dictionary user access control determines whether or not a user has a category available for use. You should ascertain which of the Dictionary standard and installation-defined categories have to be accessed by individual users. To authorize category access, you must declare the required categories in the individual user’s access profile.

Those personnel who will be defining categories, relationship-types, or structure-types for your installation need to access the following categories: ATTRTYPE, CATEGORY, RELTYPE, and STRTYPE.

The personnel who will be establishing or maintaining the access profiles will require access to the special category defined for user access control—DDUSER.

**Category Number and Name Pairs**

The Dictionary associates a unique identifying number with every category. Dictionary-supplied categories have preassigned numbers in the range of 1 through 55. Installation-defined categories have a number in the range of 56 through 255, specified as part of their definition. The number and category name form a pair that is entered in the access profile. The list of available pairs is as follows:

<table>
<thead>
<tr>
<th>Category Number</th>
<th>Category Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>DATABASE</td>
</tr>
<tr>
<td>002</td>
<td>SEGMENT</td>
</tr>
<tr>
<td>003</td>
<td>FIELD</td>
</tr>
<tr>
<td>004</td>
<td>PCB</td>
</tr>
<tr>
<td>005</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>006</td>
<td>PSB</td>
</tr>
<tr>
<td>007</td>
<td>PROGRAM</td>
</tr>
<tr>
<td>008</td>
<td>MODULE</td>
</tr>
<tr>
<td>009</td>
<td>JOB</td>
</tr>
<tr>
<td>010</td>
<td>TRANSACT</td>
</tr>
<tr>
<td>011</td>
<td>SYSDEF</td>
</tr>
<tr>
<td>012</td>
<td>DDUSER</td>
</tr>
<tr>
<td>013-019</td>
<td>Reserved for IBM usage</td>
</tr>
<tr>
<td>020</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>021</td>
<td>RELTYPE</td>
</tr>
<tr>
<td>022</td>
<td>ATTRTYPE</td>
</tr>
<tr>
<td>023</td>
<td>STRTYPE</td>
</tr>
<tr>
<td>024-055</td>
<td>Reserved for IBM usage</td>
</tr>
<tr>
<td>056-255</td>
<td>Installation-defined</td>
</tr>
</tbody>
</table>
Determining Category Authorization Requirements

In order to manipulate a Dictionary relationship, it is necessary for the user to be authorized to access both categories. Satisfying this requirement is straightforward for some commands. However, several Dictionary commands process a predetermined set of categories. Consequently, these requirements are tabulated in Figure 1.

<table>
<thead>
<tr>
<th>Command</th>
<th>DBS</th>
<th>SEG</th>
<th>PCB</th>
<th>DTE</th>
<th>PSB</th>
<th>SDF</th>
<th>TRN</th>
<th>PGM</th>
<th>MOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBD_IN</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSB_IN</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>COBOL_IN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLI_IN</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECALCULATE SEGMENT</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRUCTURES_OUT</td>
<td>*</td>
<td>*</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBD OUT</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDT.OUT</td>
<td>R</td>
<td>R</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSB.OUT</td>
<td></td>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAGE_.1_OUT</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R Required
* Depends on the category used in the command

Figure 1. Category Authorization Requirements for Certain Dictionary Commands

User Access Rules

The following access rules are in effect when Dictionary user access control is active:

1. First, the command is examined to determine if it is authorized for this user.

2. A user can access a specific Dictionary subject and its data if the access profile permits use authorization to the subject category and view or update authorization to specific status codes.

   The access authorization check is made when the Dictionary initially gets the request, which, for commands, occurs after the command itself has been parsed. For display forms, the checking occurs during processing of the Header or reusable forms.

   After completion of access authorization checking, processing that follows relationships to other subjects will allow names of those related subjects to be reported or displayed although the user’s profile may not explicitly authorize the related subject’s category or status code.

3. An access to a specific subject will allow the creation of, addition to, modification of, or deletion of that subject if the access profile specifies update authorization to the status code of the subject and use authorization to the category.
Retrieval of a specific subject by display forms, program access, or commands is allowed if the access profile specifies view authorization to the status code of the subject and use authorization to the category.

4. To establish a relationship between a pair of subjects, your installation has the option of requiring update authorization over both of the status codes involved, and category use authorization. Alternatively, use authorization to both categories, update authorization to the status code for one of the subjects, and view authorization to the status code of the other, can allow the relationship to be established. These requirements are also necessary for the addition to or modification of relationship data.

**Specifying Access Authorization for Relationships**

As indicated in the fourth access rule above, the access authorization required when establishing a relationship between a pair of subjects depends on the current criteria.

To illustrate the need for an authorization rule, suppose a programmer was working with a segment subject in test status 'T' because of the need to add an element, but the remaining elements in the segment are unchanged and identical to those in production status 'P1'. Both SEGMENT and ELEMENT categories are authorized for access. You can proceed as follows:

- The element in test status could first be related to the test segment. Given update authorization to the status code 'T', and use authorization to SEGMENT and ELEMENT categories, the ADD_RELATIONSHIP command could establish the relationship—and add the element subject (and segment) if it did not already exist. This action presents no access problem.

- To allow the elements in production status to be related to the test segment would create a structure with mixed status codes. Given further authorization, update authorization to ELEMENT subjects in 'P1' status, the elements could be related to the test segment.

- However, this implies that the user relating production subjects to test subjects has authorization to update other production elements, which may not be desirable.

The Dictionary allows you to relax the requirement for update authorization for both subjects. As long as the user has update authorization for one subject's status code, and view authorization to the other subject's status code, the mixed status relationship can be established. Also, relationship data can be added or modified. If one of the subjects did not previously exist, it is created, provided that the user has update authorization to the missing subject's status code.

You need to establish a relationship authorization rule for your installation. You use the SECREL operand of the DBDGFLTS macro to declare the rule. See OS/VS DB/DC Data Dictionary Installation Guide for information on how to use the DBDGFLTS macro. Specify a value of BOTH if you decide that, in general, there is a requirement for update authorization for both sides of a relationship. Specify a value of ONE if you want to relax the requirement.
Although an installation default is set, individual access profiles can override the relationship authorization rule. A DEFAULTS keyword in the profile has an operand SECREL=BOTH, which is exactly equivalent to the installation default.

In our previous example, suppose the programmer is working with view authorization to the 'P' production status subjects in the ELEMENT category. The programmer's access profile could override an installation default of BOTH with a specification of SECREL=ONE. In this way, relationships to those 'P' status elements could be made.

The specification for SECREL and a table of relationship authorization combinations are shown in Figure 2.

<table>
<thead>
<tr>
<th>SECREL</th>
<th>Subject A</th>
<th>Subject B</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONE</td>
<td>UPDATE</td>
<td>VIEW</td>
</tr>
<tr>
<td></td>
<td>VIEW</td>
<td>UPDATE</td>
</tr>
<tr>
<td>BOTH</td>
<td>UPDATE</td>
<td>UPDATE</td>
</tr>
</tbody>
</table>

Figure 2. SECREL Specifications and Relationship Authorization Combinations

Command Considerations

There are specific requirements for the following commands when user access control is active:

- The ADD_RELATIONSHIP, CHANGE_RELATIONSHIP_DATA, DELETE_RELATIONSHIP, EXTEND_RELATIONSHIP, and RELOCATE SEGMENT commands follow the current SECREL specification (described in the previous section). Batch forms generate ADD_RELATIONSHIP commands and, therefore, use the SECREL specification.

- Several of the SET commands have implied status code/category use:

  - The SETSTAT status code is checked for view authorization.

  - The SETSTACK and SETCLIST commands (when their operands include a category and subject name) are checked for use authorization to the category and view authorization to the status code.

  - When output routing is used to control the destination of command output, the JOB category might be used on the SETPRINT and SETPUNCH commands. In this case, the user needs JOB category use authorization and view authorization to the status code of the JOB subject.

- In connection with Dictionary stacks, the SCAN and STRUCTURE__REPORT commands can be used to generate the stack. In this case, access control requires the user to have update authorization to the status code of the subject containing the stack as well as use authorization to the stack category.
• In connection with the RUNSTACK command, as the command cited in the RUNSTACK operands is executed for each subject in a Dictionary stack, authorization to that subject is checked. Use authorization to the subject category and either view or update authorization to the subject's status code are required, depending on the command being executed.

• In connection with the RUNCLIST command, as each command in the command list is executed, the subject(s) named in the command is checked for access authorization.

• The REPORT command allows an option, *SYO, specified in place of the category name, for obtaining a glossary report for all subjects in the SYSTEM data base. To use *SYO, access authorization checking requires that the user be authorized to all subject categories in the SYSTEM data base, including the DDUER category.

• The COPY command requires view authorization for the source subject status code, and update authorization for the target subject status code as well as use authorization to the category.

Before relationships are followed and related subjects are copied, category and status authorization is required for all subjects being copied. The current SECREL authorization must be satisfied before COPY will copy any relationship.

• The DELETE__STRUCTURE command requires use authorization to all categories and update authorization to all status codes of subjects involved in the structure being deleted.

• The STRUCTURE__REPORT command requires use authorization to all categories and view authorization to all status codes of subjects in the structure being reported.

• The EXPORT command requires use authorization to all categories and view authorization to all status codes of subjects in the structure being exported.

• The PREVIEW command requires use authorization to all categories and view authorization to all status codes of subjects in the PREVIEW input stream.

• The INSTALL command requires update authorization for the '**' status code, because subjects with that status will be created. The user also needs update authorization over production status, because subjects with 'P' status involved in the installation will be deleted upon successful installation. The user also is required to have use authorization to the extensibility control information category (CATEGORY, RELTYPE, ATTRTYPE, or STRTYPE) referenced on the command.
Display Form Considerations

Access Control on Header Form: To access a subject definition form from the Header form, the user must have entered under the appropriate headings on the CATEGORY NBR/NAME line:

- Use authorization to the category, and
- View authorization to the status code

If these requirements are not met, an error message will be issued and the user will remain on the Header form.

To access a relationship from the Header form (for example, the FIELD-SEGMENTS form or the "Unselected" SUBJECT-RELATIONSHIPS form), the user must have entered under the appropriate headings on the CATEGORY NBR/NAME line:

- Use authorization to the category, and
- View authorization to the status code

However, use authorization to the category entered in the RELATIONSHIPS TO field is not required.

If the category and status code requirements are not met for the category and subject named on the CATEGORY NBR/NAME line, an error message will be issued and the user will remain on the Header form.

To access a "SEGMENT in DATABASE" form from the Header form, the user must have:

- Use authorization to the SEGMENT category, and
- View authorization to the status code of the SEGMENT subject

Use authorization to the secondary category, DATABASE, is not required nor is view authorization to the status code of the DATABASE subject named in the SEGMENT IN DATABASE field required. If the category and status code requirements are not met for the SEGMENT subject, an error message will be issued and the user will remain on the Header form.

Access Control on Subject Definition Forms: Access control may affect processing if:

- The user does not have update authorization to the status code. If the user has transferred to a subject definition form with only view authorization to the subject's status code, the user will be able to view only subject data (for example, attributes or text data on lower level forms). Any attempt to update subject data will be rejected with an error message.
- The REUSE action is used. In order to reuse a subject definition form, the user must have view authorization to the status code of the specified subject. If this requirement is not met, an error message will be issued and the definition of the specified subject (if it exists) will not be displayed.
• Transfer to a relationship form is performed. Use authorization to the related category is not required when transferring to a relationship form from a subject definition form. For example, once on the FIELD form, the user may transfer to the FIELD-SEGMENTS form, regardless of whether or not the user has use authorization to the SEGMENT category.

Access Control on Relationship Forms: Keeping in mind the level of access control provided by the Header form and the subject definition forms that allow direct transfer to a relationship form (for example, the FIELD form allows a direct transfer to the FIELD-SEGMENTS form), it is possible and even probable that the user will be able to see the names of related subjects in categories to which the user is not use authorized, or subjects that have status codes to which the user is not view authorized, or both. However, once on a relationship form, the user will never be permitted to transfer to the definition form for a related subject unless the user has both use authorization to the category and view authorization to the status of the subject.

Access Control on the Command Form: It is possible to transfer to the Command form from any display form after successful sign-on. On the Command form, commands will be individually checked for command, category, and status code authorization, as appropriate.

CMDGRP Specification of NONE: The profile statement CMDGRP ONLINE=None will mean that a user can sign-on in online mode but will only be allowed to display subjects on the formatted display forms and issue "automatically authorized" commands on the command form. The use of the EDIT or DELETE function keys would not be blocked; however, any changes attempted by these actions would be rejected with a command authorization error message.

The profile statement CMDGRP BATCH=None will mean that a user can sign-on in batch mode but only "automatically authorized" commands will be allowed.

Access Profile Changes from Release 5 to Release 6: Certain functions have been modified in Release 6 to allow for DBCS users. DBCS users can access the SETEXPLAIN command when ONLINE=None or BATCH=None. DBCS users using a terminal from the 5550 Multistation family can specify a DBCS string for user-id in the sign-on command. The addition of HDRKMOD and HDRKSEC allows DBCS facilities the same access profile options as with HDRMOD and HDRSEC.

Allowing Selective Use of Dictionary Commands

Many of the Dictionary commands are designed for a particular task and are not necessarily needed by all users. In fact, you might want to prevent the use of some commands. For example, DBD_OUT produces data base definition statements appropriate for a DDBGEN process, and your installation may have controls in place for allowing only certain personnel to perform DDBGEN. Some commands operate on many Dictionary subjects and, for performance reasons, it may be desirable that users be prevented from using these commands in the online environment.
The Dictionary assists you in controlling command use. Dictionary user access profiles can contain detailed command authorization lists. There can be one list authorized to an individual user for batch use and a different one for online.

The technique is to group sets of commands and assign an abbreviated name to each group. Authorization to the group gives authorization to each command in the group. One or more of these command groups make up the total command authorization profile. For example, the STRUCTURES__OUT and RECALCULATE__SEGMENT commands have to do with language data structures. Their assigned group name might be "LANGOUT."

The Dictionary supplies a number of these groups as a starter set (see Figure 4 on page 27), but you can change the list of included commands. You can also define your own group names and specify which commands belong in each group.

The user working with display forms can access the Dictionary Command form from any other form. You can, however, restrict the command usage. The access profile for online use can stipulate one or more groups of commands available to that profile. When user access control is active, each command that is entered on the Command form is first checked to see if that particular command is authorized. Following the check, the command is checked for its access implications.

Further details on how to define command groups and include them in the access profile are presented later in this chapter.

**Internally Generated Dictionary Commands**

Several Dictionary functions generate standard Dictionary commands and, optionally, submit them for execution. These functions are: COBOL_IN, PLI_IN, EXTEND__RELATIONSHIP, RECALCULATE__SEGMENT, display forms, and batch forms. The EXPORT function also generates commands; however, they are written to an output file instead of being submitted for execution.

In batch mode, command authorization checking is not done for internally generated commands. Consequently, authorization to the function implies authorization to any commands generated by that function. However, subjects referenced in internally generated commands are checked for category and status authorization.

In online mode, all commands internally generated by the display forms are authorization checked. To perform edit functions on all display forms, a user must be authorized to all of the update commands named in the DFORMS command group.
When user access control is active, there are specific requirements for the RUNSTACK and RUNCLIST commands:

- A user of RUNSTACK must be authorized to both RUNSTACK and the command cited in the RUNSTACK operands. For example, a user of the command:

  \[ \text{RUNSTACK RECALCULATE\_SEGMENT } \&\text{subj UPDATE=YES;} \]

must be authorized to RUNSTACK and RECALCULATE\_SEGMENT commands and the SETSTACK command that is not shown. However, the COBOL\_IN, PLI\_IN, and STRUCTURES\_OUT commands generated by RECALCULATE\_SEGMENT are implicitly authorized. Furthermore, the ADD and ADD\_RELATIONSHIP commands generated by the COBOL\_IN and PLI\_IN commands are also implicitly authorized. As the command is executed for each of the stack entries, it will be checked for access implications.

- A user of RUNCLIST must be authorized to the RUNCLIST command and any commands contained in the CLIST. As each command is executed, the subject(s) named in the command will be checked for access implications. Commands generated by a command in the CLIST are implicitly authorized. An example of a RUNCLIST command and associated CLIST is shown below.

  1  RUNCLIST
  2   \text{RUNSTACK RECALCULATE\_SEGMENT } \&\text{subj}
  3   \text{STRUCTURES\_OUT}
  4     \text{COBOL\_IN}
  5     \text{ADD}
  6     \text{ADD\_RELATIONSHIP}
  7     \text{REPORT SEG SEG A}
  8     \text{ADD SEG SEG B}

  The commands on lines 2, 7, and 8 are the only ones in the CLIST. The presence of internally generated commands is depicted by indentation.

  The commands on lines 1, 2 (both commands), 7, and 8 must be explicitly authorized by the user's access profile. Those commands on lines 3, 4, 5, and 6 need not be authorized by the user's profile, because they are implicitly authorized as a result of being authorized to the outer level command.

  Assume that the user profile authorizes all commands except ADD. The ADD command on line 5 will not fail because it is authorized as part of the RECALCULATE\_SEGMENT operation. However, the ADD command on line 8 will fail because it is entered directly from the CLIST.

**Specifying Subject Name Default Qualifiers**

You can override the installation subject name default qualifiers defined in the DBDGFALT module through the DEFAULTS keyword. You can set the following defaults in the user's access profile:

- STAT—subject name status code default
- OCR—subject name occurrence number default
Establishing Dictionary User Access Control

There are five installation defaults which affect Dictionary access control. These defaults are:

- **SECDICT**—specifies whether or not Dictionary access control is active.
- **SECIMS**—specifies whether or not the Dictionary will be operating under the security provisions of a host system.
- **SECSTAT**—specifies the status code the Dictionary is to use when verifying sign-on.
- **SECTHLD**—specifies the number of failing sign-on attempts to allow before notifying the computer operator.
- **SECREL**—specifies access control processing on the creation, maintenance, and deletion of mixed-status relationships.

The values for the defaults may be set in the **DBDGFLTS** macro and included in the **DBDGFA** module. For information on how to set these operands and their defaults, see *OS/VS DB/DC Data Dictionary Installation Guide*.

A recommended procedure to start up Dictionary user access control is:

1. Choose a status code to be used for DDUSER subjects.
2. Define DDUSER subjects, using the status code of Step 1, for key Dictionary users such as those in the areas of security or Dictionary administration.
3. If you are using the host system's security, build the user identifications and passwords of Step 2 into the security tables for IMS/VS or CICS/OS/VS.
4. Verify that sign-on can be accomplished for the DDUSER subjects created in Step 2. This entails the following:
   - Create a suitable **DBDGFA** module:
     - Activate user access control by specifying SECDICT=Y
     - Specify the status code of Step 1 with SECSTAT
     - Specify any other desired options such as SECREL, SECIMS, and SECTHLD
     - Specify any additional command groups or changes to existing ones
     - Assemble and link-edit the **DBDGFA** module
   - User verification in batch mode can be done for all users with one command stream that contains a SIGN_ON command for each user.
• User verification in online mode requires that access control be activated and the HDRSEC display form be used. Only one user can be verified per IMS/VS or CICS session.

5. Now sign on with one of the access profiles from Step 4 and continue to add other users into the DDUSER category and perform verification.

Note that the user access control options can be changed at any time with the relink-editing of DBDGFALT, even making Dictionary user access control inactive. For this reason, access to the Dictionary program library should be well protected. After verified access profiles exist, any change to the value of the SECSTAT option requires a similar change to all previously verified DDUSER subjects. See OS/VS DB/DC Data Dictionary Installation Guide for information on how to set up and specify Dictionary defaults and how to assemble and link-edit the DBDGFALT module.

Establishing Command Groups

In order to control command authorization for end users, you need to:

• Evaluate which commands and Dictionary facilities are required for individual users, or for groups of users.

• Group commands into collections for which you assign a command group name.

• Specify each command group with a DBDGSGRP macro.

• Assemble the Dictionary defaults module, DBDGFALT, containing all DBDGSGRP macros and DBDGFRLS macro specifications.


How Command Groups Are Defined

Dictionary defaults for your installation are contained in the DBDGFALT module which is link-edited into the program library. The module is assembled from source statements consisting of Dictionary-supplied macros: a DBDGFALT macro followed by one or more DBDGSGRP macros. Each DBDGSGRP macro defines both the group name and the commands that are included in that group. For example:

```
SCAN           DBDGSGRP (SC,REP)
GROUP1         DBDGSGRP (A,CI,CN,DEL,DD)
```

The command group name is a 1- to 8-character label (first character alphabetic); in the example, SCAN and GROUP1. The commands are specified with abbreviated names; for example, SC for SCAN and CI for CHANGE__IN.
Abbreviations Used in Command Groups

As explained above, each Dictionary command has a predefined abbreviation. These abbreviations are required by the DBDGSGRP macro. You cannot change any of the abbreviations. You will probably recognize many of the abbreviations as the short form of the command verbs. However, some of the abbreviations do not correspond exactly to a particular command. Several commands have different types of use, such as the REPORT command, and each type is assigned a unique abbreviated name. The full list of command abbreviations is shown in Figure 3.

<table>
<thead>
<tr>
<th>DBDGSGRP</th>
<th>Abbreviation</th>
<th>Command Name/Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ADD</td>
<td>ADD_RELATIONSHIP</td>
</tr>
<tr>
<td>AR</td>
<td></td>
<td>ADD_RELATIONSHIP</td>
</tr>
<tr>
<td>BF</td>
<td>Batch Forms</td>
<td></td>
</tr>
<tr>
<td>CBI</td>
<td>COBOL_IN</td>
<td></td>
</tr>
<tr>
<td>CI</td>
<td>CHANGE_IN</td>
<td></td>
</tr>
<tr>
<td>CKP</td>
<td>CKPT (Checkpoint)</td>
<td></td>
</tr>
<tr>
<td>CN</td>
<td>CHANGE_NAME</td>
<td></td>
</tr>
<tr>
<td>COP</td>
<td>COPY</td>
<td></td>
</tr>
<tr>
<td>CRD</td>
<td>CHANGE_RELATIONSHIP_DATA</td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>DELETE_DATA</td>
<td></td>
</tr>
<tr>
<td>DDO</td>
<td>DDT_OUT</td>
<td></td>
</tr>
<tr>
<td>DEL</td>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>DI</td>
<td>DBD_IN</td>
<td></td>
</tr>
<tr>
<td>DO</td>
<td>DBD_OUT</td>
<td></td>
</tr>
<tr>
<td>DR</td>
<td>DELETE_RELATIONSHIP</td>
<td></td>
</tr>
<tr>
<td>DRD</td>
<td>DELETE_RELATIONSHIP_DATA</td>
<td></td>
</tr>
<tr>
<td>DS</td>
<td>DELETE_STRUCTURE</td>
<td></td>
</tr>
<tr>
<td>EXE</td>
<td>EXECUTE</td>
<td></td>
</tr>
<tr>
<td>EXP</td>
<td>EXPORT</td>
<td></td>
</tr>
<tr>
<td>INS</td>
<td>INSTALL</td>
<td></td>
</tr>
<tr>
<td>PI</td>
<td>PSB_IN</td>
<td></td>
</tr>
<tr>
<td>PO</td>
<td>PSB_OUT</td>
<td></td>
</tr>
<tr>
<td>PUN</td>
<td>PUNCH</td>
<td></td>
</tr>
<tr>
<td>PLI</td>
<td>PLI_IN</td>
<td></td>
</tr>
<tr>
<td>PRE</td>
<td>PREVIEW</td>
<td></td>
</tr>
<tr>
<td>RCL</td>
<td>RUNCLLIST</td>
<td></td>
</tr>
<tr>
<td>REG</td>
<td>REPORT (Many subjects—glossary)</td>
<td></td>
</tr>
<tr>
<td>REI</td>
<td>REPORT (One subject—ISR report)</td>
<td></td>
</tr>
<tr>
<td>REP</td>
<td>REPORT (Subject specific—GUIDE report (Display form equivalent))</td>
<td></td>
</tr>
<tr>
<td>RL</td>
<td>RELOCATE</td>
<td></td>
</tr>
<tr>
<td>RS</td>
<td>RECALCULATE_SEGMENT</td>
<td></td>
</tr>
<tr>
<td>RST</td>
<td>RUNSTACK</td>
<td></td>
</tr>
<tr>
<td>SC</td>
<td>SCAN</td>
<td></td>
</tr>
<tr>
<td>SCK</td>
<td>SETCKPT</td>
<td></td>
</tr>
<tr>
<td>SCL</td>
<td>SETCLLIST</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3 (Part 1 of 2). Dictionary Assigned Abbreviations with the DBDGSGRP Macro
DBDGSGRP
Abbreviation | Command Name/Function
--- | ---
SLI | SETLOGI
SLU | SETLOGU
SNA | SNAP
SO | STRUCTURES_OUT
SPU | SETPUNCH
SPR | SETPRINT
SST | SETSTACK
STR | STRUCTURE_REPORT
SIO | STAGE__1__OUT
TRA | TRACE
XR | EXTEND__RELATIONSHIP

Figure 3 (Part 2 of 2). Dictionary Assigned Abbreviations with the DBDGSGRP Macro

Assigning Commands to Command Groups

To assist you in grouping commands, the Dictionary supplies several groupings of commands with related function and each group has a corresponding group name. The Dictionary-supplied command groups are provided with the DBDGFLAT module. These groupings are listed in Figure 4. You can add to or delete from the command list belonging to a group, or define a completely different list for that group name. Alternatively, you can define your own group names with their accompanying list of command abbreviations. A command abbreviation can occur in more than one group. You must, however, have unique group names.

<table>
<thead>
<tr>
<th>Group Name</th>
<th>Authorized Commands</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFORMS</td>
<td>BF</td>
<td>Batch forms</td>
</tr>
<tr>
<td>CLIST</td>
<td>(SCL,RCL)</td>
<td>Command lists</td>
</tr>
<tr>
<td>CONTROL</td>
<td>(CKP,SCK,SLI,SLU)</td>
<td>Program control</td>
</tr>
<tr>
<td>COPYDEL</td>
<td>(COP,DS)</td>
<td>COPY and DELETE__STRUCTURE</td>
</tr>
<tr>
<td>DFORMS</td>
<td>(A,AR,CI,CRD,DEL,DD,DR,DRD,RL)</td>
<td>Display forms facility</td>
</tr>
<tr>
<td>EXECPAF</td>
<td>EXE</td>
<td>Program Access facility</td>
</tr>
<tr>
<td>EXPORT</td>
<td>(EXP,PRE)</td>
<td>Portability</td>
</tr>
<tr>
<td>EXTENS</td>
<td>INS</td>
<td>INSTALL command use</td>
</tr>
<tr>
<td>LANGINP</td>
<td>(CBI,PLI)</td>
<td>Input language source</td>
</tr>
<tr>
<td>LANGOUT</td>
<td>(SO,RS)</td>
<td>Output language source</td>
</tr>
<tr>
<td>LPREPROC</td>
<td>(SCL,RCL,PUN,SO,SLI)</td>
<td>Language Preprocessor</td>
</tr>
</tbody>
</table>

Figure 4 (Part 1 of 2). Dictionary-Supplied Command Groups
<table>
<thead>
<tr>
<th>Group Name</th>
<th>Authorized Commands</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPTMANY</td>
<td>(REG,STR)</td>
<td>Reports for structures</td>
</tr>
<tr>
<td>REPORTS</td>
<td>(REP,REI,REG,STR,PUN)</td>
<td>Reports for many subjects</td>
</tr>
<tr>
<td>RPTONE</td>
<td>(REP,REI)</td>
<td>Report a single subject</td>
</tr>
<tr>
<td>ROUTE</td>
<td>(SPU,SPR)</td>
<td>Output routing</td>
</tr>
<tr>
<td>SCAN</td>
<td>(SC,REP)</td>
<td>SCAN and glossary</td>
</tr>
<tr>
<td>SERVICE</td>
<td>(TRA,SNA)</td>
<td>Diagnosis</td>
</tr>
<tr>
<td>STACK</td>
<td>(SST,RST,EXP,SC,STR)</td>
<td>Stack processing</td>
</tr>
<tr>
<td>STACKBLD</td>
<td>(SST,SC,STR)</td>
<td>Stack building</td>
</tr>
<tr>
<td>STACKUSE</td>
<td>(SST,RST,EXP)</td>
<td>Stack usage</td>
</tr>
<tr>
<td>SYSDEFI</td>
<td>(DI,PI)</td>
<td>Input for IMS SYSDEF</td>
</tr>
<tr>
<td>SYSDEFO</td>
<td>(DO,PO,DDO,S1O)</td>
<td>Output for IMS SYSDEF</td>
</tr>
<tr>
<td>UPDATE</td>
<td>(A,AR,CI,CN,CRD,DEL, DD,DR,DRD,XR,RL)</td>
<td>Update activity</td>
</tr>
</tbody>
</table>

Figure 4 (Part 2 of 2). Dictionary-Supplied Command Groups

Examine the list of the Dictionary-suggested command groups. They are defined in such a way as to match many of the tasks covered in other chapters of this guide or in OS/VS DB/DC Data Dictionary Applications Guide. Note that the DFORMS command group conveniently defines all commands implicitly needed to operate with the display forms facility.

Remember, the command group name is used within the user access profile and each must be predefined within the Dictionary defaults module before it can be a valid specification in the profile. If you omit the DBDGSGRP macro corresponding to the command group, this may inconvenience any end user who needs the commands authorized by that group. In fact, any profile that refers to that group name will fail at sign-on time.
Defining Dictionary Users with Commands

Subject Names for Dictionary Users

The access authorization attributes of a particular Dictionary user are described to the Dictionary as a subject in the DDUSER category. The category is part of the SYSTEM data base.

The general format of a subject name in the DDUSER category is:

(status,U,user-id,0)

The name by which a user is identified (user-id) can be up to 31 alphanemic characters, beginning with an alphabetic character. For DBCS users, the name must be a uniform string of up to 14 DBCS characters. The user-id specifies the identification the Dictionary user will enter when completing a sign-on for the Dictionary. If the request to access the Dictionary is by a SIGN__ON command, the user-id is specified with the ID keyword. For display forms, it is entered in the SIGN-ON ID field on the Sign-On Header form.

There are some special considerations for the user-id when the Dictionary system is using the security provisions of either IMS/VS or CICS/OS/VS. This is indicated by specifying SECIMS=YES. The user identification for IMS/VS can be up to 8 bytes long. It is entered with the IMS/VS command /SIGN. In CICS, the user-id is associated with the operator name specified on the CSSN transaction. Operator name and operator id are specified in the OPNAME and OPIDENT operands, respectively, in the DFHSNT macro. It is the value specified by the OPIDENT operand that is passed to the Dictionary. In either IMS/VS or CICS, this ID is subsequently passed to the Dictionary with no further need for entry in online environments. For DBCS users, the user-id does not accept DBCS characters.

Subjects in the DDUSER category must have a subject code of 'U'. The occurrence number must be '0'. Although DDUSER subjects can be defined with any status code, the code for an individual user must match the profile status code value specified for SECPSTAT in the Dictionary defaults module before it can be verified or be operational for that user.

Subjects in this category cannot be related to subjects in any other Dictionary category.
Using the ADD Command

The data attributes common to every DDUSER subject entry are described as keywords applicable to the ADD command.

The format of the ADD command for access control is:

```
ADD
  DDUSER
  DDU
  U
  subjectname

Required

Optional
  > PASSWORD=password
  SECSNBR=n
  SECLNBR=line
  SECUDPTR=(n, line)
  SYSUSERn=(nnn, 'text')
```

The operands for the command are explained below:

ADD
The command name.

DDUSER | DDU | U
This is the category name.

subjectname
This specifies a subject name as described in the previous section. The user name portion corresponds to the user identification. For DBCS users, the user name must be a uniform string.

PASSWORD=\password
Specifies the password a Dictionary user will give when completing a sign-on to the Dictionary. Up to 8 bytes will be accepted. A blank password will be rejected during sign-on time. DBCS users should not use DBCS characters in the password.

SECUDPTR=(n, line)
Specifies the User Data segment number (n) and starting line number (line) of the access profile. User Data segments contain the access descriptions for any specific DDUSER subject. The segment number has a range of 1-5. A particular line number within the range from 1 to 999 has to be specified as the line at which the access profile begins. The default is segment number 1 and line number 001. Your installation will most likely want to standardize this for all users.
The two values specified by the SECUDPTR keyword can be individually entered by the two keywords below.

**SECSNBR=n**
This keyword specifies the User Data segment number that is to contain the access profile. The range is 1-5.

**SECLNBR=line**
This keyword specifies the starting line number for the access profile. The range is 1-999.

**SYSUSERn=(nnn,'text')**
Specifies a single line of an access profile. n is the User Data segment number, nnn is the line number, and text is specified in a special format described in “Access Profile Keywords.”

The access profile associated with each subject in the DDUSER category is specified as User Data for that subject. Because the DDUSER category is contained in the SYS data base, the keyword SYSUSERN is used to specify each line of the access profile.

Changes to the access profile are discussed under “Maintaining Access Profiles” on page 46.

**Example of Creating a User Profile with the ADD Command**

This example creates the Dictionary administration profile as shown in “Maintaining Access Profiles” on page 46.

```
ADD DDU (S,U,DICTADM,0);                                           Adds a new DDUSER subject
ADD DDU (S,U,DICTADM,0) +                                          Adds the attribute data
  SECUDPTR=(3,900) +                                                 for an existing user
  PASSWORD=SUPER835;
ADD DDU (S,U,DICTADM,0) +                                          Adds access profile
  SYSUSER3=(900,'UPDTSTT=ALL');                                     keywords to an existing
ADD U" SYSUSER3=(910,'VALCAT=ALL');                                 user
ADD U" SYSUSER3=(920,'CMDCRP ONLINE=ALL');                         
ADD U" SYSUSER3=(925,'CMDCRP BATCH=ALL');                          
ADD U" SYSUSER3=(930,'END');                                       
```

Note that, instead of using the SECUDPTR keyword, you can use SECSNBR=3 and SECLNBR=900. Also, you can avoid repeating the subject name on each command by the use of a " mark adjacent to the category code (U").

**Access Profile Keywords**

The User Data lines, which specify the access profile, contain **keywords with accompanying values**. The identifying line numbers do not have to be contiguous, so that the increment can be more than one. The keywords can be placed in any order. Keywords begin in position 1 of the line, and there can be only one keyword per line.
The keywords that can be specified in an access profile are:

- **UPDTSTAT**
- **VIEWSTAT**
- **CMDGRP**
- **DEFAULTS**
- **VALCAT**
- **END**

The keywords and their possible values are:

**UPDTSTAT**(list) | code | TEST | ALL | NONE

This keyword specifies the set of Dictionary status codes that the specified user is allowed to update. Updating of Dictionary subjects whose names contain a status code not in this list will not be allowed.

The status codes can be placed in a list that is enclosed in parentheses. A single code does not require the parentheses. The list items can be separated with commas or blanks, or have no intervening space between codes. Acceptable values are any set or all of codes 'A' through 'T', '0' through '9', or the '*' code.

The value TEST specifies all the above codes with the exception of: 'P', production status, '*', installed status, and the code selected for your installation as the status code for user access profiles in the DDUSER category.

The value of ALL specifies that all status codes are available for update.

The value of NONE excludes the user from any update activity. NONE is also the default value if the UPDTSTAT keyword is not coded.

**VIEWSTAT**(list) | code | TEST | ALL

This keyword specifies the set of Dictionary status codes the specified user is allowed to view in addition to those specified for UPDTSTAT. Access to Dictionary subject definitions whose names contain a status code not in this list will not be allowed.

The status codes can be placed in a list that is enclosed in parentheses. A single code does not require the parentheses. The list items can be separated with commas or blanks, or have no intervening space between codes.

The acceptable values in the list format, or for a single code, are any set or all of the codes 'A' through 'T', '0' through '9', or the '*' code. The meaning of the TEST and ALL values is the same as for UPDTSTAT, although allowing only view and not update. You will notice that the value of NONE is not an option; the user must be able to see at least one status.
All codes specified in the UPDTSTAT keyword are also automatically valid for view authorization and need not be respecified. Consequently, the VIEWSTAT specification need only contain additional codes not specified for the UPDTSTAT keyword. If the VIEWSTAT keyword is omitted, the default will be the values specified for UPDTSTAT.

**CMDGRP ONLINE | BATCH=(list) | group | ALL | NONE**

This keyword specifies one or more command group names for the online or batch environment. The group names are defined in the DBDGFALT module. If the keyword is not coded, the default is to allow all commands in both batch and online environments.

The acceptable operand values for CMDGRP are:

**ONLINE | BATCH**

Specify ONLINE to declare that the command group names below identify the commands authorized for online use. Specify BATCH for batch use. If ONLINE is not coded, the default is to allow all commands for online usage. If BATCH is not coded, the default is to allow all commands for batch usage.

**(list) | group**

Specifies a list of command group names that have been predefined using the DBDGSGRP macro. A single command group name (group) need not be enclosed in parentheses. The list items must be separated with a comma or a single blank.

**ALL**

Specifies that all Dictionary commands and batch forms are authorized.

**NONE**

Only the following commands are authorized (in online or batch): SIGN__ON, FLUSH, DEFAULTS, SETOCR, SETSTAT, SETDBTP, SETPSIZE, SETLANG, and SETEXPLAIN.

For online mode, a user can access all display forms with a specification of NONE. However, on the COMMAND display form, only the set of commands listed above (except SIGN__ON) can be used. This effectively allows a read only capability for online mode. For batch mode, a specification of NONE is allowed but of little practical value.

A separate specification is required for each environment—online and batch.

There is no provision for continuing a CMDGRP specification to another line. However, it can be repeated as often as necessary to specify the complete list of command group names.

**DEFAULTS SECREL=ONE | BOTH STAT=x OCR=nnn DBTP=x LANG=x**

This keyword specifies overrides for some installation defaults defined in the DBDGFALT module. The operands of this keyword can specify the required access authorization rule when establishing relationships between
Dictionary subjects and will allow the installation to initialize an individual user's subject name default qualifiers during sign-on processing.

The acceptable operand values for DEFAULTS are:

\texttt{SECRET=ONE | BOTH}  
For \texttt{SECRET}, select one value:

\texttt{ONE}  
Means that update authorization is required to one of a pair of subjects and view authorization is required to the other to establish/maintain a relationship between that pair of subjects.

\texttt{BOTH}  
Means that update authorization is required to both subjects to establish/maintain a relationship between those subjects.

\texttt{STAT=x}  
The subject name status code default usually set with the \texttt{SETSTAT} command.

\texttt{OCR=nnn}  
The subject name occurrence number default usually set with the \texttt{SETOCR} command.

\texttt{DBTP=x}  
The subject name data base type default usually set with the \texttt{SETDBTP} command.

\texttt{LANG=x}  
The subject name language default usually set with the \texttt{SETLANG} command.

When the DEFAULTS keyword is not in the user's profile or some operands are not specified, the installation default will apply to this user.

\texttt{VALCAT=ALL | STANDARD | FORMAT | STANDARD-FORMAT | ALL-FORMAT}  
This keyword specifies the Dictionary subject categories available for use. The value chosen also affects how the category list on the Sign-On Header display form will appear for this user. An attempt to use (update or view) a category that is not specified is rejected and a DBD6008 message results.

The acceptable operand values for \texttt{VALCAT} are:

\texttt{ALL}  
Means that this user has all Dictionary categories available for use. \texttt{ALL} would result in a Header display form containing an entry of "ALL CATEGORIES AVAILABLE." This phrase would appear in the position usually occupied by the first line of the category number and name pairs. \texttt{ALL} is intended as shorthand notation for a user, most likely in administration, who does not need names and numbers for prompting.
STANDARD
Means the standard set of 11 Dictionary categories is authorized and will be displayed on the Sign-On Header display form. This list is in the same order as the Dictionary categories that appear in the standard Header form distributed with the Dictionary. STANDARD is the default if a VALCAT keyword is not coded.

FORMAT
Means that a list of categories is authorized. The list is specified as a series of formatted User Data lines, known as category records. This allows a tailored Sign-On Header display form for each user.

One or more lines containing a list of allowed category number and name pairs must start on the next line, immediately after the VALCAT keyword. There must be no intervening blank or keyword lines. The format of these records is described below. The content of each line specifies one or more categories that are available to the user. These records appear as category number and name pairs in the body of the Sign-On Header form. The category records are terminated by the next access keyword encountered.

STANDARD-FORMAT
Means that the standard set of 11 Dictionary categories is authorized. However, you are able to include category records and thus tailor the appearance of the Sign-On Header form.

ALL-FORMAT
Means that the user is authorized for all subject categories. However, you are able to include category records and thus will be able to tailor the appearance of the Sign-On Header form. The list of category number and name pairs does not have to include all categories; for online usage, the partial list can remind the user of a selected group of category number/name pairs.

END
This keyword marks the end of the access profile for this user, and has no operand value. A warning message (DBD6010) is issued during sign-on if the end of the User Data segment is encountered before the presence of the END keyword.

Access Profile Syntax
The syntax rules are summarized as follows:

- The series of keyword specifications can occur in any order.
- The first keyword must be coded at the line number specified for the start of the access profile.
- No blank lines can be included before the END keyword.
- Only one keyword may appear on a line, and each keyword must begin in column 1.
- Keywords, with the exception of CMDGRP, may only be used once for each user access profile.

- Any type of keyword terminates a stream of category records following a VALCAT keyword.

- Each of the category records must contain at least one category number and name pair.

**Syntax for Category Records**

The *category record format* is shown in Figure 5.

### Starting Position

<table>
<thead>
<tr>
<th>#</th>
<th>l-1Name</th>
<th>l</th>
<th>#</th>
<th>l-1Name</th>
<th>l</th>
<th>#</th>
<th>l-1Name</th>
<th>l</th>
<th>#</th>
<th>l-1Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>nnn-xxxxxxxx</td>
<td>nnn-xxxxxxxx</td>
<td>nnn-xxxxxxxx</td>
<td>nnn-xxxxxxxx</td>
<td>nnn-xxxxxxxx</td>
<td>nnn-xxxxxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5. Category Record Format**

The category record can contain from one to five category number and name pairs in the format of *nnn-xxxxxxxx*. The *nnn* specifies a 3-digit number and is immediately followed by a hyphen (-) as a separator character. The *xxxxxxxx* is replaced by a category name that can be from 1 to 8 bytes. When entering the profile with the Dictionary command language, the category records must be enclosed in single quotation marks (because this is text data), with the data in the positions shown above.

1. *nnn* is a 3-digit category number that must be coded, and must start in column 1, 16, 31, 46, or 61. Leading and trailing blanks are allowed for 1- or 2-digit category numbers. Numbers in the range from 56 to 255 are valid for installation-defined categories. See "Category Number and Name Pairs" on page 15 for the valid numbers for Dictionary-supplied categories.

2. A *separator* character, '-' or 'l', must be coded in column 4, 19, 34, 49, or 64 between the category number and name.

3. A category name, *xxxxxxxx*, must be coded beginning in column 5, 20, 35, 50, or 65, and cannot consist of more than 8 bytes. DBCS users can use a uniform string for the category name.

It must be emphasized that it is the *category number* that is used by the Dictionary to authorize access to the subjects in the category. The category number specifications are checked to see if they are numeric during sign-on time. However, when the User Data lines are being entered, or when they are being processed during sign-on, there is no check on the category records to verify that:
• The number corresponds to some installed category
• The name corresponds to some installed category
• The name and number are a matched pair
• Correct category numbers have been specified

Syntax for CMDGRP Specification

• **CMDGRP** must be coded starting in column 1 and be followed by one or more blanks.

• **ONLINE | BATCH** must be followed by the equal (=) symbol.

• **(list) | group | ALL | NONE**

    Immediately following the = symbol, code a list of command group names enclosed in parentheses. Each item in the list must be separated by a comma or a single blank. A single command group name need not be enclosed in parentheses. **ALL** or **NONE** cannot be specified as the name for a single command group.

There can be one or more command group specifications. Usually, there will be a set for online and a set for batch, although these two types can be intermixed. If the access profile is specifically for the online environment, **CMDGRP BATCH=NULL** should be specified in addition to the ONLINE specification. Similarly, for a batch-only profile, **CMDGRP ONLINE=NULL** should be specified in addition to the BATCH specification.

If you code a group specification of **ALL** or **NONE** for a batch or online environment, other declarations for that environment are redundant and will cause an error when the access profile is processed.

If you construct a list, the closing parenthesis must be coded within the 80-byte limit. Remember, your installation can use the 72-column version of the User Data display form (specified in the Dictionary defaults module with the **UDFORM** keyword). If this is the case and you intend to maintain the access profiles with display forms, you will need the closing parenthesis to occur at or before column 72.

Syntax for DEFAULTS Specification

• The profile statement identifier, **DEFAULTS**, must start in column 1. One or more blanks must separate the identifier **DEFAULTS** from the first keyword-value pair and each of the other keyword-value pairs specified on the statement. A comma is not allowed for a separator character.

• The keywords **SECREL**, **STAT**, **OCR**, **DBTP**, and **LANG** must be immediately followed by an equal sign (=) and an associated value.
Examples of User Access Profiles

In the examples that follow, the first line is the user's id (user name of DDUSER subject), while PASSWORD and SECUDPTR are attributes of the DDUSER category and follow on separate lines. The actual profile specifications follow with appropriate line numbers for the SYSUSERn text.

Security Administration Access Profile

SECURITY
  PASSWORD=SEC797
  SYSUSER1
    001 UPDTSTAT=S
    002 VALCAT=FORMAT
    003 012-DDUSER
    004 CMDGRP BATCH=ALL
    005 CMDGRP ONLINE=ALL
    006 END

DICTADM
  PASSWORD=SUPER835
  SECUDPTR=(3,900)
  SYSUSER3
    900 UPDTSTAT=ALL
    910 VALCAT=ALL
    920 CMDGRP ONLINE=ALL
    925 CMDGRP BATCH=ALL
    930 END

Developer Access Profile

DEVELOPER
  PASSWORD=APPL22
  SECUDPTR=(3,100)
  SYSUSER3
    100 UPDTSTAT=(A B T)
    110 VIEWSTAT=P
    120 VALCAT=FORMAT
    124 001-DATABASE 009-JOB 010-TRANSACT 056-SPECS
    128 002-SEGMENT 007-PROGRAM 011-SYSDEF 057-PROJECT
    132 003-FIELD 008-MODULE
    136 004-PCB 005-SYSTEM
    140 006-PSB
    150 CMDGRP ONLINE=(UPDATE,REPORTS)
    160 DEFAULTS SECRET=ONE STAT=T OCR=000 DBTP=L LANG=C
    170 END
Command User Access Profile

PROGRAMR
  PASSWORD=R56789
  SECUDPTR=(1,10)
  SYSUSER1
    010 VIEWSTAT=P
    020 UPDTSTAT=(T,9)
    030 CMDGRP ONLINE=(UPDATE,REPORTS,BFORMS)
    032 CMDGRP ONLINE=(LANGINP,LANGOUT)
    034 CMDGRP ONLINE=(STACKBLD,STACKUSE,CLIST)
    038 DEFAULTS SECREL=ONE STAT=T OCR=000 DBTP=L LANG=C
    040 VALCAT=STANDARD
    050 END

Language Preprocessor Profile

PLIPREPR
  PASSWORD=LP6789
  SYSUSER1
    001 VIEWSTAT=(P T)
    002 UPDTSTAT=NONE
    003 CMDGRP BATCH=LPREPROC
    004 VALCAT=STANDARD
    005 END

Defining Dictionary Users with Display Forms

In this section we assume you are familiar with the online display forms. For more information, see “Using Display Forms” in Chapter 3 of *OS/VS DB/DC Data Dictionary Terminal User’s Guide and Command Reference*.

Using the DDUSER Form

The DDUSER form is used to enter a user identification and password for the access profile. The blank form in edit mode is shown in Figure 6 on page 40.
Figure 6. DDUSER Form

Accessing a DDUSER Form

You may access a DDUSER form from the Header form. You enter the user identification as a subject, together with a category name of DDUSER (or abbreviation DDU or U), or you can use a category number of '012'.

Content

The name and status are those entered on the Header form. (Occurrence number '0' and subject code 'U' are assumed.) The DESC field may be used to enter one line (up to 72 bytes) of description text for the user. For DBCS users, mixed strings are allowed.

The labeled fields are for the following operands:

PASSWORD
This field specifies a sign-on password. Up to 8 bytes may be entered. For DBCS users, no DBCS is allowed.

USERDATA-SEGME-NBR
This field specifies the number of the User Data segment in which the access profile is to be stored. Values from 1 through 5 are valid. A value of 1 is assumed if nothing is specified.
USERDATA-LINE-NBR

This field specifies the starting line in the selected User Data segment for the access profile.

Although all lines from 1 through 999 are available for use, the access profile must begin on the line specified for USERDATA-LINE-NBR. A value of 1 is assumed if nothing is specified.

Working with the Form

If the user subject name at the top of the form is not highlighted, you can enter new values for the three attributes, then select the PROC option (PF key 1). To change a password, just write over the old password and remove extraneous characters. The name corresponds to the user identification used for sign-on, so that the exact spelling of the name, and the password, should be carefully checked.

To enter data for an additional user, you may alter either or both of the NAME and STAT fields, and select the REUSE action (PF key 9). The REUSE action is the only way to change these fields and indicates that the same screen is to be used for another DDUSER subject.

The access profile is not entered on this form; it is entered on a USER DATA form. In order to obtain this form, type the desired User Data segment number (same number as used in USERDATA-SEGM-NBR field or the number 1 for the default) in the field adjacent to the USERDATA-NO action, and then:

- Press PF key 6, or
- Enter a “6” next to “ACT” and press the ENTER key.

The format of the access profile is described in “Access Profile Syntax” on page 35. The format is also described in the EXPLANATION form obtained by selecting the DDUSER form’s EXPLAIN action (PF key 11).

Additional text data can be entered in user data segments, both in the access profile segment and the others available, but no text should be positioned within the set of lines used by the access profile.

Using the USER DATA Form for Access Profiles

The USER DATA form is used to enter information into the User Data segment selected to contain the access profile, or to display that information. Refer to OS/VS DB/DC Data Dictionary Terminal User’s Guide and Command Reference for details on the use of this form.

The 15-line version of the USER DATA form, with an illustrative access profile, is shown in Figure 7 on page 42.
Working with the Form

The User Data segment to which this form refers is identified by the appropriate number (from 1 to 5) in the display form title. The user identification is displayed in the field at the top of the form. Access profile data already stored in the Dictionary for this user is displayed in line-number order.

When the form is first presented, the window is positioned at the TOP of the requested segment of User Data. All User Data text for this segment can be displayed on this form, even text not related to the access profile. If other User Data text is being stored in this segment ahead of the access profile, then you must use PF key 6 to scroll through this text until the lines for the profile appear. If no other text lines have been defined before the start of the access profile, you may begin to work on the first display.

The category record lines following a VALCAT keyword line require the category names to be placed in particular columns; see “Access Profile Syntax” on page 35 for specifications. Each line containing a category record will require any necessary blanks to be typed in.

You do not have to assign consecutive numbers to successive lines, but you must not have lines consisting of all blanks separating the access profile lines. After you have checked your entry, and corrected it if necessary, select the PROC action to store the new User Data.
Use of a Model Access Profile

Because of the category record requirements for number-name pairs in fixed positions, the use of an access profile model by your installation is recommended. Figure 8 on page 44 illustrates a recommended model and suggests a name of (S,U,MODEL,0). Use of a model requires that you keep the original unchanged and make a copy for modification.

Create a new user profile by displaying the model on the DDUSER form. From the DDUSER form, access the COMMAND form and make a copy of the DDUSER subject (S,U,MODEL,0). You can give the copy a name (the new user's id) for whom you wish to define an access profile. When finished making a copy, return to the DDUSER form and use the REUSE action (PF key 9) to display this new user. Make the necessary changes for DESC, PASSWORD, and profile location and process the form with the PROC action. Next, select the USERDATA-NO action (PF key 6) to display the access profile where the other necessary changes can be made.

To specify the values for the UPDTSTAT, VIEWSTAT, CMDGRP, and DEFAULTS keywords, you replace the operands LIST, TEST, and so forth with actual values. The NNN-XXXXXXXX entries are replaced by actual category number-name pairs. Any lines not needed can be erased by writing over the line number in the LNO field with minus signs.

If more category records were needed, add more lines. Additional lines must be entered in an empty line at the bottom of the form. The User Data lines are sequenced by line number. After the line(s) have been entered, use the PROC action (PF key 1) to complete the update and redisplay the form with User Data in line number sequence.

If the access profile does not need to override the installation defaults, the DEFAULTS record can be deleted. Otherwise, enter the appropriate values. Remember that the END keyword must have a higher line number than the other access profile records, and that no blank lines are allowed within the access profile.
### Customizing Header Forms

The specification of the VALCAT keyword in the access profile has a direct impact on the content of the Sign-On Header form. Each category record image, containing one or more category number and name pairs, appears as a line in the body of the Sign-On Header form. The category number/name pairs can be as dense as five for each line or as sparse as one for each line. The effect of the fixed positions is to create columns of categories that are easily scanned. Also, the category number is associated with the name. Entries will appear on the Header form exactly as coded. Note that the Sign-On Header form is a window form, and will allow eight of these formatted lines to be displayed in the body of the form for each frame.

As an illustration, the Developer profile given in “Examples of User Access Profiles” on page 38 is used for the Header form shown in Figure 9 on page 45. Notice the effect of the column syntax, the reordering of the standard categories, and that the number and name of two installation-defined categories appear.

---

**Figure 8. USER DATA Form with Model Access Profile**
Authorizing and Verifying Access Profiles

It is recommended that the DDUSER subject and the associated access profile be tested in batch mode using a SIGN_ON command. The online method using the Sign-On Header form can be used, but can be more time consuming because only one error or warning message can be displayed.

No validation of the access profile keyword specifications is done until the user attempts sign-on. Errors found at that time may cause sign-on to be refused, depending on whether an error message of severity higher than W (Warning) is issued.

The Dictionary will ascertain whether a user can gain access by looking for a DDUSER subject name in the form:

(status,U, user-id, 0)

The status code is taken from the Dictionary default for the DDUSER profile status code. If this DDUSER subject exists, the password check will be made.
Verifying Sign-On in Batch

Each user identification and the associated access profile can be checked for errors in its specification by submitting a **SIGN__ON** command for each user. For example:

```
SIGN_ON ID=DICTADM PASSWORD=SUPER835
```

This command produces an extensive check on the accuracy of access keywords coded in the User Data segment for this DDUSER and an immediate verification of the user identification and password value.

Verifying Sign-On Online

The user identification and password are entered on the initial display of the Sign-On Header form. The same validation checking that is performed for the access profile in batch occurs for the display form user. Any message concerning the sign-on verification appears in the response line.

If IMS/VS or CICS/OS/VS security is being used and the **SECIMS** option is **YES** in the Dictionary defaults module, an automatic sign-on is performed by the Dictionary the first time an entry is made on the Sign-On Header form. The user identification that was supplied as input to the host system is passed to the Dictionary, and the password check is assumed to have already occurred. If an ID and password are manually entered on the Header form, they will be ignored.

Verifying Category–Number Pairs

The category number and name pairs specified in the access profile with **VALCAT** are not validated at sign-on time. Therefore, specification errors may result in a user being denied access to a desired category or given access to the wrong category.

For example: The **DATABASE** category is defined in the Dictionary as category number 1. If the access profile contained a specification of 002-DATABASE, then the user would be authorized to the **SEGMENT** category (because 2 is its number), although 002-DATABASE was displayed on the Sign-On Header form.

One method of checking for this type of error uses display forms. You enter the category numbers one by one on the Header form, with a fictitious subject name, and check the title of the display form that is displayed. It should, of course, be the equivalent name of the category shown on the Header form.

Maintaining Access Profiles

The access profile is most easily altered with the USER DATA display form, though the **CHANGE__IN** command can be used.
The use of CHANGE_IN is convenient to provide routine alteration of passwords. For example:

```
CI DDU (S,U,DICTADM,0) FROM PASSWORD=SUPER835 +
      TO PASSWORD=222222 ;
```

The CHANGE_IN command can also be used to change one entire line of the access profile. For example:

```
CI DDU (S,U,DICTADM,0) FROM EXISTING +
      TO SYSUSER3=(900,'UPDSTAT=*') ;
```

Another method of entering or maintaining data for the Dictionary user and the access profile is to use batch forms. Use the DB/DC Data Dictionary Text-Data Form. The preprinted form has column numbers to assist with the category record format.

**Reports for DDUSER Subjects**

You can use REPORT and SCAN commands to inspect the DDUSER subject definitions and their associated access profiles. You use the category name DDUSER, or the abbreviation DDU or U, with these commands.

For example, to obtain a glossary report containing the total number of Dictionary users, though not including the access profiles, you can use the command:

```
REPORT GLOS DDUSER; or REPORT GLOS *SYO CODE=U;
```

The access authorization data for an individual Dictionary user can be obtained from a subject-specific report. For example, to request the data for DDUSER subject (S,U,PGRMR47,0) you could use the command:

```
REPORT DDU (S,U,PGRMR47,0);
```

The attributes of the user could be reported by adding a specification of NAME=DDUSER to the above command. This would give a report similar to the display form shown in Figure 6 on page 40.

**Allowing Users to Define Their Own Access Profiles**

Although, in this chapter, it has been assumed that the control of access to the Dictionary data lies with administration personnel, this restriction can be relaxed.

The potential user could be given a sign-on identification that would allow access to the DDUSER category, but with a chosen status code different from the the status code used for verified access profiles. The user would define a user identification, password, and access profile. Security administration would then be informed of the new definitions and would examine the access request, the required categories and status codes, and authorize their use. Subsequently, the DDUSER subject would be copied to the status code reserved for verified user access profiles, and have validation checking performed. For example,

```
COPY DDU (R,U,USERX,0) TO DDU (S,U,USERX,0) UPDATE=YES
```
An alternative mode of operating with Dictionary access control is to make the status code for user access profiles generally known, and to have the users be responsible for their own access control. The protection of Dictionary data would then be controlled by Dictionary users. Access control would be achieved with the unique user identifications and individual passwords.

If you use either of the above procedures, you would have to notify all users of the command group names established for your installation. The details of which commands are authorized within each group would have to be given. Any additional requirement for command authorization would entail an update to the DBDGFALT module prior to verification of the access profile.
Chapter 3. Planning for Dictionary Extensions and Customization

This chapter provides a description of the support offered to administrative personnel for Dictionary extension, and contains the basic conceptual information required to understand subsequent chapters.

The development activities by which the scope of the Dictionary system can be enhanced are:

- The definition of installation-defined subject categories, and the further definition of relationship-types that exist between subjects in these categories, or between subjects in these categories and subjects in standard categories

- The provision of validation routines for subject names and attribute validation

- The definition of structure-types, which determines the categories to be included and how the relationship-types existing between those categories are to be interpreted

- The preparation of installation-written programs to retrieve Dictionary data and perform subsequent reporting or other processing

The routines and other parts of the Dictionary that allow you, as Dictionary administrator, to create customized subject categories and relationship-types (that is, to "extend" the Dictionary) are collectively called the Extensibility facility.

You can also provide name validation routines to check user names specified for subjects in any installation-defined or standard category. In addition, you can provide attribute validation routines to check input values specified as attributes of a subject in an installation-defined category, or as relationship data for an installation-defined relationship-type.

A structure-type definition allows you to define a structured relationship over a set of categories. These definitions can include both standard and installation-defined categories. You can then copy, delete, report, and export subjects in these structures.

The content of the Dictionary data bases can be examined using the REPORT, STRUCTURE__REPORT, and SCAN commands, or with display forms. This data can also be retrieved by programs that are developed by your installation. Such programs can customize the content and format of individual reports. Alternatively, they could perform other processing on the retrieved data. For example, a program could validate a set of definitions for completeness.
The ability for installation-written programs to extract data and produce output using printer, punch, terminal devices, or Dictionary stacks assigned to the Dictionary, is provided by the Program Access facility.

**Installation-Defined Dictionary Categories and Relationship-Types**

Definitions for new subject categories and relationship-types for addition to the Dictionary system can be developed concurrently with the usual usage. The definitions are entered with Dictionary update commands or may be constructed using display forms. However, the developer is required to reach a point in the development at which refinement of the definition and attributes of the customized subject category, or of the kind of the relationship-type, ceases and the definition is considered complete. At this point, the definition is installed. This process causes the defined category or relationship-type to be made an operational part of the Dictionary. The installed definition is subsequently used by the Dictionary programs when they process commands referring to those entities.

When the installation-defined subject category or relationship-type is made a "permanent" part of the Dictionary (that is, made generally operational), users can define subjects in the category and establish instances of the relationship-type, using Dictionary update commands and display forms. In this way, the data structure of the Dictionary can be "extended" to other classes of data resource.

**Subjects and Subject Categories**

Subject categories are the basic structural elements of the Dictionary. In addition to the standard Dictionary categories, you can establish up to 200 categories of your own choosing. The content of these installation-defined categories can document types of resources that are of specific interest within the context of your installation's data processing operations. In defining each new subject category, the installation can specify attributes that are to be recorded as subject data for subjects in that category, and simple naming conventions that are to be enforced for the subject names. After the definition of a new subject category has been completed and made operational, the users of the Dictionary can enter definitions of subjects in that new category in much the same way as those of subjects in standard categories.

The specification of installation-defined subject categories is described in Chapter 5, "Defining Subject Categories" on page 87. The documentation of these categories is assisted by Dictionary reports both before and after they are installed. Chapter 7, "Installing Categories and Relationship-Types" on page 111 describes two special kinds of reports: the GUIDE report that summarizes the information a general user has to know in order to enter data for subjects in the category, and the FORMAT report, which reveals how that data will be stored in the Dictionary data base.
Relationships and Relationship-Types

"Relationships" are the other key element of the Dictionary. The Extensibility facility enables you to define new types of relationships that may be established:

- Between pairs of subjects in installation-defined categories
- Between subjects in an installation-defined category and subjects in a standard category

Extensibility relationship-types cannot be defined for relationships between pairs of subjects in the standard categories.

Each type of relationship is known by a pair of keywords.

The standard relationships in the Dictionary are denoted by the use of the keyword WITH (as in SEGMENT WITH DATABASE). For an Extensibility relationship-type, you must specify the keyword to be used in identifying the relationship. There are two keywords to specify: a "forward" keyword to be used when referring to the relationship in one direction (that is, from a subject in one category to a subject in another category), and an "inverse" keyword to be used when referring to the relationship in the other direction (that is, from a subject in the latter category to a subject in the first category).

The ability to assign "forward" and "inverse" names or keywords to a relationship-type is useful, because those names can indicate the explicit interpretation of an instance of that type of relationship. For example, you might define a relationship-type that is used to record that an application ACCESSES a particular data base (or the inverse, that the data base is ACCESSED_BY that application). Those names, ACCESSES and ACCESSED_BY, would then be used as keywords in any commands that were issued to establish, update, or delete relationships of that type. The keyword that is used in referring to a relationship is determined by the order in which the two subjects are specified. In the example above, a user would be able to add the relationship:

Application ACCESSES Data base

or, the inverse

Data base ACCESSED_BY Application

but an attempt to add a relationship

Application ACCESSED_BY Data base

would be rejected as invalid (or as referring to an undefined type of relationship).

You will notice that the forward and inverse names appear in Dictionary reports and on appropriate relationship display forms.

You can establish more than one type of relationship between the same pair of Dictionary categories.

The ability to establish more than one type of relationship between the same pair of Dictionary subjects is important, because a pair of subjects might reasonably be
"related" in two or more ways. For example, you can establish two different types of relationships between the pair of subjects representing a programmer and a program: the programmer WROTE the program, and the programmer UPDATED the program.

You can establish relationships between pairs of subjects in the same installation-defined category.

Relationships often exist between data or other resources of the same classification. An example might be EMPLOYEE MANAGES EMPLOYEE. When viewed from the subordinate employee, the same relationship could be thought of as EMPLOYEE MANAGED_BY EMPLOYEE.

The ability to establish a relationship between pairs of subjects in the same subject category should be distinguished from the Dictionary concept of aliases. The Dictionary has an example of this type of nonalias relationship in the ELEMENT CONTAINS ELEMENT relationship.

The forward and inverse keywords for a relationship can be the same.

A relationship need not make any distinction as to which category is specified, first or last. For example, MAINFRAME ATTACHED_TO CHANNEL and CHANNEL ATTACHED_TO MAINFRAME.

Each type of relationship is uniquely identified by its keywords and the categories it connects.

If two types of relationships, PGMR WROTE PROGRAM and PGMR UPDATED PROGRAM, were defined, the relationship-types would be distinguished by the different keyword names. If the additional relationship PGMR WROTE MEMO was defined, it would be distinguished from the PGMR WROTE PROGRAM relationship-type by the use of the different category name MEMO.

The use of the relationship keywords and the order in which the subjects are specified to distinguish relationship-types are further illustrated by an example of two relationships that are similar in appearance, but that differ in their logical implications.

A PGMR CALLED ENDUSER (or ENDUSER CALLED_BY PGMR) type of relationship could be defined to record information about telephone calls between programmers and end users, where the programmer initiated the call.

A second type of relationship, ENDUSER CALLED PGMR (or PGMR CALLED_BY ENDUSER), could also be defined to record information about telephone calls that were initiated by end users.

Note that, when referring to a particular PGMR-ENDUSER relationship, the type of relationship would be stipulated by the combination of the keyword that is used and the order in which the two subjects are specified.
You can sequence instances of a relationship between a subject and the subjects to which it is related.

You can specify that the relationships of a particular type, between a given subject and the subjects to which it is related, are to be ordered or sequenced on the values specified for a particular attribute on those relationships. This capability is important in that relationships may be used to represent some sort of "structure," where the order of the components within the structure is important.

For example, you might define a JOB CONTAINS JOBSTEP relationship-type to record information about the job steps that make up different jobs. Because the order of the steps within a job is important, you could define a "step-number" attribute for the relationship-type that would be used to sequence the instances of that type of relationship, between a given JOB subject and the JOBSTEP subjects to which it is related.

If a relationship-type is sequenced, you can define more than one instance of that relationship between the same pair of Dictionary subjects.

The ability to establish more than one instance of the same type of relationship between the same pair of subjects is useful in those situations in which the relationship represents some type of "event" that might occur multiple times. For example, in the case of the PGMR UPDATED PROGRAM relationship, a given programmer might update the same program several different times.

In order to be able to establish more than one instance of the same type of relationship between the same pair of subjects, the relationship-type must be sequenced. If the PGMR UPDATED PROGRAM type of relationship were sequenced on, for example, an attribute indicating the date on which the update was made, the different instances of the relationship (representing different "updates") would then be distinguished by the different "date" values associated with those instances.

A relationship can be defined to be "directed."

A relationship-type may have an implied "structure," meaning that one subject is a component of the other. This type of "directedness" would, for example, be inherent in the JOB CONTAINS JOBSTEP relationship given earlier; each instance of the relationship would relate a job to one of its component job steps.

Another aspect of "directedness" is the dependency of one subject on another, for example, the relationship between an application and a database that it accesses.

If an Extensibility relationship-type is specified to be "directed," then the standard Dictionary status rules will be enforced for instances of that relationship. Specifically, when the relationship-type is operational, the end user will not be permitted to relate subjects in the "subordinate" category that are in test status to subjects in the "superior" category that are in production status 'P'. As examples, test level JOBSTEP subjects could not be related to a production-level JOB subject, and a relationship could not be established, indicating a dependency of a production application on a test-level data base.

Not all relationships need to be directed. For example, a relationship-type COMMAND ISSUED FROM TERMINAL might be defined to keep track of
which commands could be issued from which terminals. Because new commands that are being tested might be issued from "production" terminals (that is, terminals that are permanently attached to your system), and "production" commands might be issued from a terminal that is still being tested, you would specify that the COMMAND ISSUED FROM TERMINAL relationship-type is not directed, so that the status rules would not be enforced for relationships of that type.

Details of the specification of relationship-types, and the use of reports to assist in their documentation, are presented in Chapter 6, "Defining Customized Relationship-Types" on page 99. Chapter 7, "Installing Categories and Relationship-Types" on page 111 describes the GUIDE report the general user requires to establish instances of a new relationship, and the FORMAT report that reveals how the relationship data will be stored in the Dictionary data base.

Attributes of a Subject or Relationship

An "attribute" expresses some property of a subject or relationship; for example, the "access method" to be used by a particular data base, or the "starting position" of a data element within a segment (the latter being an attribute of the relationship between the element and segment definitions). These attributes are specified with keywords on update commands: for the above examples, ACCESS or START.

You can define the set of attributes to be recorded for the subjects in an installation-defined category or an installation-defined relationship-type, together with the keywords identifying those attributes. Each keyword must be unique within the set chosen for the particular category or relationship-type. DBCS users must enter a uniform string for the keyword name.

An attribute can "repeat"; in this case, multiple instances of the property or attribute are recorded for a given subject or relationship. Repeating values are specified by adding a numbered suffix to the keyword name when it is used in commands. For example, the PASSWORD attribute could repeat three times, and the values would be entered with the syntax, PASSWORD(1)=XXXXX, PASSWORD(2)=YYYYY, PASSWORD(3)=ZZZZZ.

You can control the order in which attributes of the subjects in an installation-defined category or the instances of a relationship-type will appear in reports. That same order is present on display forms: the attributes of a subject on the EXTENSIBILITY SUBJECT form, and the attributes of a relationship on the RELATIONSHIP DATA form.

The same general property or "attribute-type" might underlie one or more of the attributes to be recorded for the subjects in different categories (or for different types of relationships); indeed, two or more of the attributes to be recorded for the subjects in a given category might reflect the same underlying property. For example, several different categories and relationship-types might have attributes that are "dates."

For this reason, the Dictionary permits you to define attribute-types (such as the "date" type of attribute) as separate entities, and then associate them with category and relationship-type definitions. Every attribute you define for a category or relationship-type definition has a corresponding base or reference attribute-type,
and is identified by a keyword name that is unique to that definition. For example, two different attributes to be recorded for subjects in an EMPLOYEE category, indicating the dates on which the employee was hired and terminated, would both be of the "date" attribute-type, but would be identified by different keywords such as HIREDATE and TERMDATE. The keyword TERMDATE might also be used in another context; for example, the ending date for a project as being an attribute of a subject in the PROJECT category.

The definition for an attribute-type has three distinct components:

- A description of what the values of the attribute represent (that is, what "property" of a subject or relationship is described by the values of the attribute)
- A description of the "format" of the values for the attribute (that is, how values for the attribute are specified in commands and on display-forms)
- Specifications of the manner in which the input values for the attribute are to be validated

The description of the attribute-type itself (that is, what the values for the attribute represent) may be entered in Description and User Data text.

In specifying the "format" for the values for an attribute-type, you can establish minimum and maximum lengths, and can specify basic characteristics of the values (that is, whether the values must be numeric, all alphabetic, and so forth).

An attribute value can be checked against a list of valid values, or range tested. Also, you can specify that an exit routine is to be invoked to validate and either accept or reject the input values for an attribute. (These routines are written by your installation.)

Details of the specification of attribute-types, keyword names, and the order of the attributes are given in Chapter 4, "Defining Attribute-Types" on page 75.

How Customized Entities Are Defined

Defining an extensibility subject category, relationship-type or attribute-type, is similar to defining any other type of subject: The definition is contained as a subject in a Dictionary subject category. Four categories contain, as subjects, the definitions of installation-defined entities:

- CATEGORY, for subject category definitions (category number 20)
- RELTYPE, for relationship-type definitions (category number 21)
- ATTRTYPE, for attribute-type definitions (category number 22)
- STRTYPE, for structure-type definitions (category number 23)

These categories are known as extensibility control information categories.
Definitions that reside in these subject categories can be entered, updated, and deleted using Dictionary update commands and display forms.

Each subject definition can include the standard Description and User Data text. Furthermore, for CATEGORY and RELTYPE subjects, you can set aside one set of User Data to serve as “explain” text for the category or relationship-type definition. This explanatory text (which you must develop) is then displayed to online users when they select the EXPLAIN action.

The way you define attributes for a new category or relationship-type, is to establish an instance of a relationship between an ATTRTYPE subject and the CATEGORY or RELTYPE subject. The ATTRTYPE subject definition has data type specifications appropriate for the attribute. The relationship operands control the ordering and repeat factors for the attribute, and also specify the keyword name for the attribute.\(^1\)

An example of installation-defined categories and their relationships to the CATEGORY, RELTYPE, and ATTRTYPE categories are shown in Figure 10.

![Diagram of installation-defined categories and relationships](image)

**Figure 10.** Example of Installation-Defined Categories and Relationships to CATEGORY, RELTYPE, and ATTRTYPE Categories

In Figure 10, the installation-defined category named PGMR is defined for programmers and USERPROG is defined for programs written and used at our installation. The attribute-type DATE is defined for use as a date attribute with an installation-defined category. The category named PGMR has an attribute that is used to define the date a programmer wrote a program. This attribute is identified by the keyword PROGDATE on the CATEGORY/HAS/ATTRTYPE relationship between PGMR and DATE. The category named USERPROG has an attribute that is used to define the date a program was last updated. This attribute is identified by the keyword UPDATE on the CATEGORY/HAS/ATTRTYPE

---

\(^1\) CATEGORY and RELTYPE subjects each participate in a type of relationship to ATTRTYPE subjects. In addition, the ATTRTYPE subjects can participate in two types of relationships, to either CATEGORY or RELTYPE subjects.
relationship between USERPROG and DATE. Note that the category, ATTRTYPE, contains the subject, DATE, which is used to define both PROGDATE and UPDATE.

The Installation Process

Before a user of the Dictionary can enter actual definitions for subjects in a new subject category, or establish instances of a new relationship-type, the definition for that category or relationship-type must be made operational by using the INSTALL command (available in both online and batch modes.)

Installing a subject category (or relationship-type) definition involves a complete validation of the definition (and the definitions for the associated attribute-types). Successful installation results in internal tables and other control information that will be used by the Dictionary to process commands and access display forms for the subjects in that new category (or relationships of that type). Using the INSTALL command to make a definition operational for the general user is called the installation process.

When the new subject category (or relationship-type) is installed, its definition is given the special "installed" status, denoted by status code ‘**’. During installation, definitions for all of the attribute-types associated with a subject category (or relationship-type) are also validated and changed to installed status.

Note: The rules regarding the status of subjects in a hierarchy of relationships are extended to permit subjects having status ‘**’ to occur under subjects having any other status—including 'P'. Refer to Chapter 2 in OS/VS DB/DC Data Dictionary Applications Guide for further details.

Restrictions for Deleting Installed Entities

After the definition for a subject category, relationship-type, or attribute-type is placed in installed status, certain restrictions are present:

- Category definitions cannot be deleted as long as the Dictionary contains definitions for subjects in that category.

- Relationship-type definitions cannot be deleted if there are instances of that relationship-type.

- A category may not be deleted if it is associated with any installed relationship-type.

- An attribute-type definition that is installed cannot be deleted if it is related to any installed subject category or relationship-type definition.
Modifying Installed Entities

In the previous section, it was indicated that the Dictionary has some provisions for protecting data that has been created for installed customized entities. The concept is that of not allowing a definition to be deleted from the Dictionary while other definitions or existing data depend on that definition. There may be reasons, however, for you to modify a definition that has been installed and has existing data.

Suppose you find, for example, that an additional attribute is needed for an previously installed category. You would like to add that attribute to the definition so that the end user can enter it, using the display form for the category or a command entry. In addition, there could be a need to change the validation of an attribute, or resequence the order so that the display form was more conveniently arranged. Although you make these changes, any existing data should be retained.

Again, suppose a change to one of the (nonsequencing) attributes used to collect relationship data is needed. You would like to delete an attribute. You would like to inhibit the appearance of this attribute on the RELATIONSHIP DATA display form and inhibit data entry with the old keyword, as permitted by the commands.

The procedure you follow when modifying an installed definition is called the reinstall process. The procedure includes the creation of a new definition, the use of the INSTALL command to replace existing definitions, and processing reports that assist you in verifying the changes and assessing the impact upon existing data, but the procedure is not significantly different from the installation process. Further examples of why an installed definition may have to be changed are in Chapter 8, "Modifying Installed Category and Relationship-Type Definitions" on page 135.

There are some restrictions in the scope of the changes that you can make to installed definitions. However, the definition can be modified and the existing data migrated to the new definition.

When you are defining customized entities, you should be aware of the reinstall constraints. Subsequent chapters that explain how to define customized entities emphasize these constraints. However, it is logical to thoroughly investigate the definition requirements before initially installing a definition and to keep the data collection activity relatively fixed. If the data does not need to be migrated, you can modify a definition that is installed by deleting it and installing a new one.

Validation Routines

You can provide routines to validate and either accept or reject:

- User names specified for subjects in any installation-defined or standard category

- Input values specified as attributes of a subject in an installation-defined category, or as relationship data for an instance of an installation-defined relationship-type
Planning for Name Validation Routines

Your installation can establish name validation requirements and develop a routine to check each user name as it is entered into the Dictionary. A particular routine can be used for an individual installation-defined or a standard category, or the same routine may be used for several categories. You can add the requirement for name validation at any time, change the name of the routine that is invoked, or drop the requirement. When you adjust the implied name specifications by altering a validation routine, ensure that existing names are not made invalid, at least not without a plan to correct them.

The exit routine is invoked after basic name validation is performed by the Dictionary, that is, a check for valid minimum and maximum lengths, and a check for valid name type (such as all alphabetic characters). The routine cannot change the subject name that is passed to it. But it can supply some message text and, by the action of a return code, cause the Dictionary to issue the message and continue processing based on the severity of that message. You would probably want to reject inappropriate names using a message of level 'E', although the routine can cause messages of severity level '0', '4', or '8' to be used.

You specify name validation routines as part of the category definition. If you are adding a validation routine to an installation-defined category, you may add it to the category definition before or after installation. If you are adding a routine to a standard category, you may add it to the corresponding installed category definition that was supplied with the Dictionary.

If you add a name validation routine as part of an installation-defined or standard category definition, the routine is invoked whenever a subject in that category is specified in a command that may result in the addition of the subject or is added to the Dictionary during command processing. Figure 11 summarizes the commands or functions which may result in the addition of one or more subjects and the categories of those subjects.

<table>
<thead>
<tr>
<th>Command/Function</th>
<th>Categories of Subjects Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Category specified in command</td>
</tr>
<tr>
<td>ADD__RELATIONSHIP</td>
<td>Categories specified in command</td>
</tr>
<tr>
<td>Batch forms</td>
<td>Categories of all subjects specified</td>
</tr>
<tr>
<td>CHANGE__NAME</td>
<td>Category of new name</td>
</tr>
</tbody>
</table>

Figure 11 (Part 1 of 2). Summary of Commands or Functions for Name Validation Processing
<table>
<thead>
<tr>
<th>Command/Function</th>
<th>Categories of Subjects Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>COBOL__IN</td>
<td>ELEMENT and SEGMENT</td>
</tr>
<tr>
<td>COPY</td>
<td>Category of new name, and with the DEPENDENTS or AFFECTED options, the categories included in the specified structure-type definition</td>
</tr>
<tr>
<td>DBD__IN</td>
<td>ELEMENT, SEGMENT, and DATABASE</td>
</tr>
<tr>
<td>Display forms</td>
<td>Categories of subjects on key forms and categories of related subjects on relationship forms</td>
</tr>
<tr>
<td>EXTEND__RELATIONSHIP</td>
<td>ELEMENT or SEGMENT and category of target subject</td>
</tr>
<tr>
<td>PLI__IN</td>
<td>ELEMENT and SEGMENT</td>
</tr>
<tr>
<td>PSB__IN</td>
<td>PROGRAM, PSB, PCB, TRANSACTION, SEGMENT</td>
</tr>
</tbody>
</table>

Figure 11 (Part 2 of 2). Summary of Commands or Functions for Name Validation Processing

Planning for Attribute Value Validation Routines

The second type of validation routine gives you the ability to examine a value specified as an attribute on a display form or with a command for a category or relationship-type defined by your installation. You use a routine instead of range checking or matching to a list of valid values. This check is in addition to the basic syntax checking performed by the Dictionary, that is, the type of data and its allowed number of bytes.

Each individual attribute-type defined by the installation can include value checking by an exit routine in its specification. Several attribute-types can use the same routine. If a category (or relationship-type) uses an attribute-type that has a validation routine specified, the attribute-type will be assigned a keyword name in the usual way. After installation, any user input attempting to add a value for that name will cause the corresponding routine to be invoked. Value validation can only be specified during attribute-type definition and the routine name is fixed by the installation process.
General Requirements for Validation Routines

The validation routines can be written in COBOL, PL/I, or Assembler language. They execute in the same region as the Dictionary. Each validation routine must be link-edited with a Dictionary linkage module that is appropriate for the source language of that routine. The placement of validation routines in libraries and the programming details are described in Chapter 10, "Validation Routines" on page 175.

Understanding Dictionary Structures

A structure is a collection of subjects grouped together by relationships. Figure 12 illustrates an instance of a structure consisting of data base subjects, segment subjects, and element subjects. A structure-type contains a list of relationship-types. These relationship-types identify the categories of the subjects in the structure and the connecting relationships between the subjects.

Figure 13 on page 62 illustrates the structure-type definition for the structure illustrated in Figure 12. This structure-type consists of the DATABASE WITH SEGMENT relationship-type, the SEGMENT WITH ELEMENT relationship-type, and the ELEMENT CONTAINS ELEMENT relationship-type. For example, the relationship between the DBS1 and SEG1 subject in Figure 12 is an instance of the DATABASE WITH SEGMENT relationship-type in the structure-type definition shown in Figure 13.

```
```

Figure 12. An Example of a Structure
Figure 13. A Partial Structure-Type of Standard Categories

Figure 13 illustrates three of the standard relationship-types contained in the IBM-supplied structure-type. In addition, your installation may want to define structure-types to meet its own requirements. You can define structure-types which contain standard relationship-types and installation-defined relationship-types.

Representing a Dictionary Structure

A structure-type may be represented as a list of relationship-types, or graphically, as in Figure 13, with ellipses and lines connecting them. The ellipses represent a subject category and the lines represent the relationship-types defined between the different categories.

Relationship-types connect two categories with a relationship keyword, for example, DATABASE WITH SEGMENT. A relationship-type may have direction, that is, one category is superior to the other category. A superior category is one that is above another category in that hierarchic path. Similarly, a subordinate category is one that is below another category in that hierarchic path.

Relationship-types that have a direction are represented with lines that are "arrows." These arrows point from a superior category to a subordinate category. Relationship-types that have no direction are represented with straight lines. All relationship-types in Figure 13 have a direction.
Figure 14. An Installation-Defined Structure-Type

Figure 14 illustrates a structure-type your installation might define for your use. In the DEPT HAS PGMR relationship-type the DEPT category is superior to the PGMR category. The PGMR OWNS MODULE relationship-type shows that the PGMR category is superior to the MODULE category, which is superior to the FIXES category. The PGMR USES TERML relationship-type shows that the PGMR category is neither superior nor subordinate to the TERML category. This relationship-type has no direction. The PGMR HAS BACKUP PGMR relationship-type has no direction either. The PGMR HAS SKILL relationship-type shows that the PGMR category is superior to the SKILL category. The SKILL REQUIRES EDUC relationship-type shows that the SKILL category is superior to the EDUC category.

Understanding Structure-Type Usage

Dictionary functions using structure-types will process the relationships in a structure according to the way the relationship-types have been defined in the structure-type. The structure-type definition identifies how relationships are to be processed and which relationships are to be processed.

If a relationship-type is included in an installed structure-type it becomes part of that structure-type. It will be included in any processing done by the Dictionary function that is using the structure-type.

Which relationships will be processed is determined by whether the relationship-type has been defined as part of the structure-type. Both the direction and processing action will determine how the relationship in the structure will be processed. These concepts are discussed in the following sections. Some relationship-types may have neither a direction nor a processing action specified in the structure-type definition. However, because it is part of that structure-type, there is some default processing done for that relationship-type entry.
Defining Superior and Subordinate Categories

The direction of a relationship-type is determined by specifying a superior category, SUPERCAT (and, implicitly, a subordinate category). A relationship-type that has its left-hand category defined as superior, will have the right-hand category as the subordinate category. Therefore, when processing a subject in the left-hand category, the relationship to the subject in the right-hand category is considered to have a downward direction. If the right-hand category has been defined as the superior category, when processing a subject in the right-hand category, the relationship to the subject in the left-hand category is considered to have a downward direction. Similarly, if the right-hand category has been defined as the superior category, when processing a subject in the left-hand category, the relationship to the subject in the right-hand category is considered to have an upward direction.

You can define a relationship-type to have direction in one of two ways:

- By specifying DIRECTED=YES for the relationship-type definition. When DIRECTED=YES is specified, the left-hand category is by definition the superior category. Status rules for instances of this relationship-type will be enforced. Status rules are discussed in Chapter 2. "Using the OS/VS DB/DC Data Dictionary" in OS/VS DB/DC Data Dictionary Applications Guide.

- By specifying DIRECTED=NO for the relationship-type, but specifying SUPERCAT=L (or R) for the relationship-type definition in the structure-type definition. SUPERCAT=L indicates that the left-hand category is superior to the right-hand category. Status rules for instances of a relationship-type defined in this manner are not enforced.

That is, a relationship-type will have direction if:

\[
\begin{align*}
\text{DIRECTED}=\text{YES} & \quad \text{or} & \quad \text{DIRECTED}=\text{NO} \\
\text{SUPERCAT}=\text{L} & \quad \text{(Status rules enforced)} & \quad \text{SUPERCAT}=\text{L} \text{ (or R)} & \quad \text{(Status rules not enforced)}
\end{align*}
\]

A relationship-type has no direction if it has been defined with SUPERCAT=N. This means that neither the left-hand nor the right-hand category is considered superior.

Understanding Processing Action

A subject may be processed and a relationship followed, that is, the relationship is processed and the related subject is considered for further processing. That is, relationships to subjects on this hierarchic path can be processed and followed to process each of the related subjects on this hierarchic path.
Relationships can be followed only when the relationship-type has been defined as having direction. That is, the relationship-type must have a superior category (and implicitly a subordinate category). Relationships can then be followed in an upward direction and/or a downward direction.

<table>
<thead>
<tr>
<th>DIRECTED=YES</th>
<th>DIRECTED=NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUPERCAT=L</td>
<td>SUPERCAT=L (or R)</td>
</tr>
<tr>
<td>FOLLOWUP</td>
<td>FOLLOWUP</td>
</tr>
<tr>
<td>FOLLOWDN</td>
<td>FOLLOWDN</td>
</tr>
</tbody>
</table>

A relationship will not be followed if SUPERCAT=N was specified for that relationship-type.

Processing action for following relationships is determined by directional operands on the following structure-dependent functions:

<table>
<thead>
<tr>
<th>COPY</th>
<th>DEPENDENTS</th>
<th>AFFECTED</th>
<th>FOLLOW DOWN</th>
<th>FOLLOW UP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE_STRUCTURE</td>
<td>DEPENDENTS (implied, but not specified on the command)</td>
<td>FOLLOW DOWN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPORT</td>
<td>DEPENDENTS</td>
<td>FOLLOW DOWN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRUCTURE_REPORT</td>
<td>DEPENDENTS</td>
<td>AFFECTED</td>
<td>FOLLOW DOWN</td>
<td>FOLLOW UP</td>
</tr>
</tbody>
</table>

A relationship is followed in a downward direction if:

- The DEPENDENTS operand is specified (or implied) on the structure-dependent function.

- The relationship has direction and the subject just processed is the superior subject.

- Y (YES) is specified for the FOLLOW DOWN flag for the relationship-type.

A relationship is followed in an upward direction if:

- The AFFECTED operand is specified on the structure-dependent function.

- The relationship has direction and the subject just processed is the subordinate subject.

- Y (YES) is specified for the FOLLOW UP flag for the relationship-type.

A relationship-type with no direction (SUPERCAT=N) will be processed by structure-dependent functions (for example, it will be reported), but it will not be followed.

Depending on the processing function’s directional operands, either the FOLLOW UP or FOLLOW DOWN values are used to determine if the related subject and its relationships are to be processed.
Using Dictionary Structure-Types

A structure-type may be defined to include only the relationships that are of interest to a particular application. However, this particular view may be part of a more comprehensive view that is being used in a different way by some other applications. That is, a structure-type may be defined for use which contains relationship-types that are also represented in other structure-types. Relationship-types in a more comprehensive structure-type that have not been defined in a subset structure-type are called extrastructural relationship-types for that structure-type. Relationships will be ignored by Dictionary structure-dependent functions if they have not been defined in the structure-type referenced for that function.

Dictionary supplied structure-dependent functions, such as COPY, DELETE__STRUCTURE, EXPORT, and STRUCTURE__REPORTS require the name of the structure-type to determine which relationships are to be processed, and how they are to be processed. The STRUCTURE operand on these commands will cause these functions to process only those subjects that are part of the structure, as defined by the structure-type named by this operand.

The structure-type illustrated in Figure 14 on page 63, named DEPT, may be viewed as the most comprehensive structure-type. It contains the definitions for relationship-types which may be of interest to different applications. For example, you may have an application that processes all the data for a programmer. This would require access to both the MODULE category and the SKILL category. You could obtain all the information needed for a certain programmer by referencing the structure-type named DEPT.

You may also have an application that only processes the modules owned by a specific programmer. You could obtain information for all the modules owned by a specific programmer by only referencing the structure-type named MAINT, as shown in Figure 15 on page 67. You can see that MAINT is really a subset of the more comprehensive DEPT structure-type. On the other hand, you may have an application which only processes the skills of a programmer. To obtain the data on the skills for that programmer, you would only need to reference the structure-type named TASK, as shown in Figure 16 on page 67. Figure 17 on page 68 shows the structures mapped by the TASK and MAINT structures.
Figure 15. MAINT Structure-Type

Figure 16. TASK Structure-Type
The Default Structure-Type

A structure-type must be defined in order to drive structure-dependent functions. This structure-type may be one defined by your installation, or the one supplied with the Dictionary. If you do not specify a structure name on those commands that process structures, the default structure-type will be used.

The default structure-type is named DEFAULT. The default structure-type supplied with the Dictionary contains all of the relationship-types supplied with the Dictionary. Figure 18 on page 69 illustrates relationship-types for the standard categories, (however, not all of the relationship-types supplied with the Dictionary are shown).

The DEFAULT structure-type assumes an installation with DL/I applications. If you are using a non-DL/I application, the DEFAULT structure-type could be represented and processed with a structure-type as shown in Figure 19 on page 70.
Figure 18. Hierarchy of Categories Defined in the DEFAULT Structure-Type
Accessing the Dictionary Data with User Programs

The Program Access facility enables programs written for your installation to retrieve data from the Dictionary data bases. The programs can be written in COBOL, PL/I, or Assembler language. They execute in the same region as the Dictionary. The programs can be invoked both in batch and online.

The retrieved data is transmitted between the Dictionary and the programs in logical "collections"; for example, a set of attribute data or a list of aliases for a subject. The programs can exercise "selectivity" in determining which specific data items, or how many items, they are to receive from the Dictionary, and the location of those data items in a data area.

After it is in control, the program invokes the Dictionary services with calls that have an entry point of CBLTDBD, PLITDBD, or ASMTDBD, corresponding to COBOL, PL/I, or Assembler language, respectively. Operands in the call identify control blocks that have been established by the program to describe its request. Data is returned to the program in its designated data area and in a format determined by a controlling table.

If Dictionary user access control is enabled, a program is subjected to access authorization checking. During the verification of the operands, it is determined, for each call, whether the user who requested execution of the program is authorized to access the information requested. A return code will be set in the program access control area if the user is not authorized.
Programs may be added to a program library for execution in batch or online modes of operation concurrent with the usual Dictionary usage, and may be modified to fit new requirements.

For information on writing Program Access facility programs, see Chapter 12, “Writing Programs to Access Dictionary Data” on page 209.

Retrieving Attribute Data from the Dictionary

Programs can retrieve:

- Attribute data associated with a given Dictionary subject
- Attribute data associated with an occurrence of a Dictionary relationship

The program can request all or only a part of the available attribute data, and can specify where the data is to be placed in an input area.

Retrieving Other Dictionary Data

Programs can request variable lists of data. These requests differ from attribute data calls because a desired number of items and retrieval positioning information are specified. You can retrieve:

- The aliases for a subject
- The primary names for the subjects in a specified category
- All names (including aliases) for the subjects in a subject category
- All or a specified subset of the relationships in which a given subject participates
- All of the possible relationships between a subject in a given category and other subjects
- Lines of text data (lines of Description, User Data, or PL/I Data)
- All the relationship-types included in a structure-type definition

The lists of data may be reported directly, or may be used by the program to scan a set of Dictionary subjects or a structure of interrelated subjects. A program could also, for example, examine the completeness of one or more related subject definitions.
User Program Output

Provision is made for user programs to send output to a Dictionary-assigned printer, punch, or terminal. When invoking an output operation, the program specifies:

- The destination of the output (printer, punch, or terminal)
- The carriage control character (for single spacing, double spacing, or starting a new page)
- The location and length of a data area that contains the output characters

When using the Dictionary in batch mode, output directed to the terminal goes to the printer.

If output routing has been invoked, the output from the user program is routed to the requested destination. For details of output routing, refer to Chapter 6 of OS/VS DB/DC Data Dictionary Applications Guide.

Provision is also made for user programs to create entries in a Dictionary stack. Prior to invoking a user program, the Dictionary user sets the location of the stack using the SETSTACK command. This command specifies the name of a Dictionary subject and a set of User Data belonging to that subject, which will contain the generated entries. It is also possible to direct stack output to a sequential data set or to a member of a PDS, as specified by the SETSTACK command operands. Refer to Chapter 7 of OS/VS DB/DC Data Dictionary Applications Guide for details on the uses and control of Dictionary stacks.

Communicating with a User Program

Programs that retrieve Dictionary data are invoked with the Dictionary EXECUTE command. The EXECUTE command can be used in both batch and online modes.

The user program can set a return code before exit. The return code value can determine whether or not subsequent Dictionary input is processed and affects the condition code for the job step.

User Program Characteristics

User programs execute in the same region as the Dictionary, subject to the following restrictions:

- Their entry point must be DBDWUP.
- They must not attempt to issue DL/I calls.
- They must not initiate any subtasking.
- They must not use STAE or ESTAE.
- They must free any virtual storage they allocate.
- COBOL user programs should not be compiled with the DYNAM option.
Dictionary-Supplied Programming Aids

To assist in the preparation of data retrieval programs, a number of programming aids are supplied with the Dictionary.

The macro library contains segments of source code for use with PL/I and COBOL. The blocks of source code statements assist programmers in specifying the detail of the CALL statements; they also initialize many of the control fields and return code values. The macro library contains macro support for CALL statements and initialization for programs coded in Assembler language.

Three sample programs are available, written in PL/I, COBOL, and Assembler language, respectively. Each program has different retrieval functions. It is recommended that a programmer assigned to develop a user program study the content of all three sample programs. The programs illustrate, in a variety of ways, how programs can use the Dictionary-supplied source code segments. They should also be useful in demonstrating the application of the Program Access facility to the general user. Therefore, the sample programs should be installed according to the instructions outlined in the next section.

These coding aids are described in more detail in Chapter 12, "Writing Programs to Access Dictionary Data" on page 209. This chapter also identifies the libraries and member names for the source code segments and sample programs.
Chapter 4. Defining Attribute-Types

This chapter describes ATTRTYPE subjects, and explains their roles when used with installation-defined categories or relationship-types. The chapter contains prerequisite information for use with category and relationship definitions. After initial study, you may wish to review the specifications for category and relationship-types and refer to specific sections in this chapter.

The definition for an attribute-type describes a basic data item. Attribute-type definitions are contained in the ATTRTYPE category. The definitions for attribute-types are treated as independent subjects that can be related to CATEGORY or RELTYPE subjects. You can develop the definitions for attribute-types incrementally, using update commands or display forms.

Attribute-Type Subject Names

The attribute-type subject name has the general format:

\[(\text{status}, \text{attr-type-name}, \text{ooc})\]

This subject name is used to access the attribute-type definition, and is the name that is specified when this attribute-type is related to a category or relationship-type.

The \text{status} of an attribute-type subject may be any of the standard codes while under development, or '*1 when installed.

No subject-code is specified in the subject name. The subject-code is determined implicitly by the subject category (that is, ATTRTYPE) of the definition.

The \text{attr-type-name} can be from 1 to 31 bytes. If the name is EBCDIC, the first character must be alphabetic. DBCS users must use a uniform string.

The \text{ooc} can be any number in the range from 0 through 255.
Regarding the use of ATTRTYPE subject names:

- The attr-type-name is not necessarily the same as the keyword name that is used by the end user to refer to the attribute. That keyword name is established when the attribute-type is associated with a category or a relationship-type.

- You can establish aliases for the ATTRTYPE subject.

- An ATTRTYPE subject must be in production status 'P' before the definition can be installed, though it can be developed in any standard status. The status rules require that ATTRTYPE subjects related to CATEGORY or RELTYPE subjects at 'P' status must also be at 'P' status. The ATTRTYPE subject has a status code of '*' when one of the categories or relationship-types to which it has been related is installed. The status is not changed from 'P' if the installation process for an installation-defined category or relationship-type encounters an error.

**Attribute-Type Definition Operands**

Two types of operands that define an attribute-type are:

- Operands that describe the syntax of data items and are made "permanent" when the attribute-type is installed

- Date, Description, and User Data text that can be modified after the attribute-type is installed

The operands are specified with update command keywords or as fields on display forms.

The operands for attribute-type definition are discussed below.

The DATATYPE operand is specified as a single letter code to represent the character constraints placed upon values for this attribute-type. Although the value of this operand can be specified or updated at any time during the development of the attribute-type definition, it must be specified before the definition is installed. This operand may not be modified after installation. There is no default value.

The values you can specify are:

- **Z** Numeric (0 through 9). No DBCS allowed.

- **A** Alphabetic (A through Z, @, #, or $). No DBCS allowed.

- **C** Alphameric (Numeric, alphabetic, or , /, __, or -). For DBCS users, may be a mixed string.

- **N** Name format. Alphameric, but first character must be alphabetic. For DBCS users, must be a uniform string.
L List format. Values take the form of a list of one or more character values separated by EBCDIC commas and enclosed in EBCDIC parentheses. For DBCS users, each of the list entries may be a mixed string.

Q Quoted string. Values take the form of a string of characters enclosed in EBCDIC single quotation marks. Any characters (including blanks) may appear within the string, except that EBCDIC single quotation marks occurring within the string must be represented by a pair of EBCDIC single quotation marks. For DBCS users, this may be a mixed string. DBCS quotation marks do not need to be doubled.

The following is an example of the list format:

(value1,value2,value3)

The following is an example of the list format using DBCS:

(<d1d2d3d4>,<dAdB>CD,ABCD)

where \textit{dn} is a DBCS character.

The following are examples of a quoted string:

'A QUOTED STRING EXAMPLE'

'USER'S ROUTINE'

'2 \times 2 = 4'

The following is an example of using DBCS:

'<d1d2d3d4> is an example'

The \textit{DATAMIN} and \textit{DATAMAX} operands are used with \textit{DATATYPE} to specify the basic criteria for the values that will be entered as subject data or relationship data into the Dictionary data bases.

\textit{DATAMAX} and \textit{DATAMIN} establish maximum and minimum permissible lengths for the values of the attribute-type. For the definition to be valid, \textit{DATAMIN} must be less than or equal to \textit{DATAMAX}. Each length can be in the range from 1 to 120.

The operands may be entered or changed at any time during the development of the attribute-type definition. \textit{DATAMAX} must be specified before the \textit{ATTRTYPE} subject can be installed; there is no default. \textit{DATAMIN} is optional, with a default value of '3' for a \textit{DATATYPE} of 'L' or 'Q' (to allow for the delimiters); otherwise, the default is '1'.
The VALIDATE operand indicates that further validation is to be accomplished for key values. You can specify:

**LIST**  The input values will be checked against a list of values specified by the VALLIST operand.

**RNG**  The input values will be range checked; the low and high limits are specified by the VALRNGL and VALRNGH operands.

**RTN**  The input values will be validated by an installation-written routine; the name of the routine is specified by the VALRTN operand.

The LIST and RNG options are valid only for attributes having a DATATYPE of 'Z', 'A', 'C', or 'N', and the attributes must have a DATAMAX less than or equal to '24'.

The VALLIST operand is specified as a collection of permitted values. You specify VALLIST(n)=value, where n can be in the range from '1' to '99', for an individual value. DBCS users may use a DBCS string if allowed by the DATATYPE. Each value must not contradict the above basic validation criteria. You must specify at least one value if VALIDATE=LIST.

The VALRNGL and VALRNGH operands establish minimum and maximum values. DBCS users may use DBCS strings if allowed by the DATATYPE. The Dictionary uses standard EBCDIC collating sequence to test whether a value is equal to VALRNGL or VALRNGH, or lies between them. (VALRNGL must be less than or equal to VALRNGH.) VALRNGL and VALRNGH must be specified if VALIDATE=RNG.

The operand VALRTN specifies the name of a validation routine used to check a value that is entered. You must specify a 1- to 8-character name exactly as the load module will be named in the user library. This operand is required if VALIDATE=RTN is specified. These routines are not supplied with the Dictionary and are assumed to be written to the requirements of the validation. The validation is performed in addition to the basic syntax check. A routine may, for example, enforce values permitted in more than one range. For DBCS users, DBCS characters are not acceptable as validation routine names and should not be used for the VALRTN operand.

The programming requirements are described in "Requirements for Validation Routines" on page 176.

Three points should be noted about the VALRTN specification:

- Validation by routine must be selected before installation. There is no provision for adding a VALRTN name or modifying the value of the VALRTN operand after the related category or relationship-type has been installed.

- The basic syntax check of the value implied by DATATYPE, DATAMAX, and DATAMIN will be applied to the input keyword value before the validation routine is invoked.
• The validation will be performed only for values specified in an ADD or
  ADD__RELATIONSHIP command, or in the TO operand of the
  CHANGE__IN or CHANGE__RELATIONSHIP__DATA command.
  Consequently, if the validation routine is changed in such a way as to make the
  values incompatible for some of the subjects already stored, the user will still
  be able to obtain reports on those subjects, and to change their values to those
  that are acceptable to the modified routine. The user will, however, be
  prevented from adding new subject data with invalid values, or from
  establishing new relationships with invalid relationship data.

DATE, DESC, and USERn form the second type of operands. You may enter or
update DATE or lines of text at any time during the development of the
attribute-type, or after installation.

The DATE operand may be used to associate a date with the attribute-type
definition. This attribute is not interpreted or maintained by the Dictionary; it is
provided merely for documentation purposes. The value is specified as 6 numeric
digits. There is no default.

The DESC and USERn (n is a value from '1' to '5') operands are used to define
lines of text documenting the attribute-type. The standard Description text and
five sets of User Data are available with respect to a subject specified for
ATTRTYPE. For DBCS users, the text portion of both DESC and USERn may be
entered using mixed strings.

Text lines are entered as they are for standard categories; for example:

```
ADD ATTRTYPE (8,'ACCNO,0) +
USER1=(10, 'OBJECTIVE OF VALIDATION ROUTINE');
```

Alternatively, you may use the DESCRIPTION or USER DATA display form to
enter this text.

**Relating an Attribute-Type to a Category or Relationship-Type**

In the definition of a category, the attributes that may be specified for a subject in
that category are defined by relationships between the category definition and the
definitions for the attribute-types. This method is also used for attribute-types that
become relationship data for an installation-defined relationship-type.

There are three operands (SEQ, KEYWORD, and REPEAT) that control how the
ATTRTYPE subjects are made a part of the definitions of categories and
relationship-types. The operands are defined with ADD__RELATIONSHIP and
CHANGE__RELATIONSHIP__DATA commands, or as fields on the
CATEGORY-ATTRIBUTES or RELTYPE-ATTRIBUTES display form.

SEQ is a sequencing operand that establishes an ordering within the set of
attributes associated with the category or relationship-type definition. This
ordering is reflected in reports and on display forms.

The same attribute-type definition can be used for more than one of the attributes
specified for a category or relationship-type. The placement of that attribute in the
list of keywords is distinguished by the required SEQ operand value. In this
situation, the SEQ operand is used both to indicate the ordering of the attributes for the category or relationship-type, and to distinguish among the different occurrences of the same attribute-type.

The relationship is specified by using CATEGORY or RELTYPE as the primary category, and a forward name of HAS and ATTRTYPE as the related category. You may also use the relationship keyword APPLIES TO as the inverse name when ATTRTYPE is the first category named in the relationship. If the same ATTRTYPE subject is related to the same CATEGORY or RELTYPE subject more than once, you must specify a different SEQ value.

The SEQ operand is required on any relationship command involving a subject in the ATTRTYPE category. The value must be specified in the range from '1' to '999'.

The KEYWORD operand specifies the keyword that will be used by the end user to reference the attribute in both commands and display forms. The KEYWORD names used for different attributes of a category or relationship-type must be unique to that category or relationship type. The keyword name must be no more than 8 bytes. For DBCS users, the name must be a uniform string. If the name is EBCDIC, the first character must be alphabetic. The keyword name can be added or changed at any point in the development of a category or relationship-type definition, but it must be present before the definition can be installed.

The REPEAT operand specifies the number of times the attribute may be repeated for an individual subject or relationship instance. This is an optional operand. You may specify an attribute to repeat up to '999' times. The effect of this operand is to reserve space for the specified number of values, each identified by the same keyword name and with the same data type characteristics. The values are specified by using the keyword name and a numbered suffix. For example, if the keyword SPACE were to have a REPEAT value of '3', the end user could specify any or all of SPACE(1)=, SPACE(2)=, or SPACE(3)=.

You can enter or update the value for REPEAT at any time in the development of the related attribute-type, but the value may not be changed after the category or relationship-type is installed.

In assigning SEQ values to repeating keywords, you should keep in mind that the order in which the keywords appear on an EXTENSIBILITY SUBJECT or RELATIONSHIP DATA display form is directly dependent on the SEQ values. This same order is preserved on reports. One aspect of repeated attribute keywords is that they occur together on a display. These items could be given higher SEQ numbers so that they don't fill up the initially displayed form. You should specify an adequate number of repetitions for any repeating attribute. However, each possible instance of a repeating attribute requires a separate entry in the attribute data area on the EXTENSIBILITY SUBJECT display form (or on the RELATIONSHIP DATA form). Also, the Dictionary reserves space for the maximum number of instances in the data base storage for a subject. From these perspectives, the REPEAT value specified for a repeating attribute should also be the minimum necessary. A large number of repetitions should not be specified merely to provide a safety factor. If you do need more, you could use the reinstallation process, which has the advantage of preserving the existing values.
Using Commands for Attribute-Type Definitions

The operands described in the previous sections are specified as standard keywords with update commands. The keyword names and valid values are summarized in Appendix C of OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference.

Attribute-type definitions may be defined and updated using the ADD and CHANGE_IN commands. Updates to attribute-type definitions that are installed are restricted to the DATE attribute, and to Description and User Data text.

The CHANGE_NAME command may be used to change the primary name or alias for an attribute-type definition. However, to change the status, the COPY command must be used to make a copy of the definition at the new status level. The ADD_RELATIONSHIP command (using the HAS_ALIAS keyword) is used to establish alias names for an attribute-type.

The DELETE_DATA command may be used to delete individual lines of Description and User Data text associated with an attribute-type definition at any status level.

The ADD_RELATIONSHIP command may be used to establish relationships between the attribute-type definition and the definitions of the subject categories and relationship-types with which the attribute is associated. You cannot relate ATTRTYPE subjects to subjects in standard categories. The CHANGE_RELATIONSHIP_DATA command can be used to change the operand SEQ, KEYWORD, or REPEAT. The DELETE_RELATIONSHIP_DATA command is not used to delete any of those operands. Instead, you use the DELETE_RELATIONSHIP command to delete the association of an attribute-type with a category or relationship-type.

Using the COPY Command

The COPY command may be used to create copies of an attribute-type definition under different subject names, to copy a definition from one of the standard status levels to another (but not to the installed status), and to copy an attribute-type with installed status to obtain another definition at one of the standard status levels.

Deleting an Attribute-Type Definition

For any of the standard status codes, an attribute-type definition may be deleted using the DELETE command. However, when an installed attribute-type is related to an installed category or relationship-type, the attribute-type cannot be deleted.
Procedure for Changing an Attribute-Type Definition

We have seen that an attribute-type definition is a general specification for a data item with integrated rules that control the value checking. Further, an attribute type can be used by either category or relationship-type definitions one or more times. Because a particular definition could be used as a multipurpose attribute, the Dictionary does not allow you to modify an installed attribute-type definition. You can, however, modify the DATE, DESC, and USERn operands at any time. You may also add and update relationships between installed ATTRTYPE subjects and category or relationship-type definitions that have not been installed.

Reporting Attribute-Type Development

The standard Dictionary subject-specific reports can be used to report the attribute-type definitions developed at your installation. In addition, the Dictionary glossary REPORT and SCAN commands can be used to obtain reports on the subjects in the ATTRTYPE category.

Dictionary subject-specific reports are obtained using the REPORT command, as described in Chapter 6 of *OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference*. The REPORT command can be used in the usual manner to obtain full or partial information about the definition of an installation-defined attribute-type (or, more precisely, about the ATTRTYPE subject), either before or after it has been installed. To obtain display form equivalent reports, the NAME option can specify ALIAS for ATTRTYPE subject aliases, DESC for the Description text, or USERn for a set of User Data text. The DETAIL options for subjects in “installation-defined categories” on the REPORT command can be used to restrict the information to be included in the output report.

Example of a Subject-Specific ATTRTYPE Report

Figure 20 on page 83 shows an example of a subject-specific report for an installed attribute-type named PROG_LANG_SKILL. Note that this is the standard Dictionary subject-specific report for a subject residing in an “Extensibility” subject category, and would usually show fields for all 99 instances of the repeating VALLIST attribute.

Under the RELATIONSHIPS: subtitle, the report shows that the attribute-type is related to the PGMR category; that is, as the definition for one of the attributes that end users can specify for PGMR subjects. The definition of a more “common” type of attribute (perhaps an attribute-type representing a date) might show relationships to many different CATEGORY and RELTYPE subjects.
Figure 20 (Part 1 of 3). Example of ATTRTYPE Subject-Specific Report
Figure 20 (Part 2 of 3). Example of ATTRTYPE Subject-Specific Report
VALLIST (93):
VALLIST (94):
VALLIST (95):
VALLIST (96):
VALLIST (97):
VALLIST (98):
VALLIST (99):

RELATIONSHIPS:

APPLIES_TO CATEGORY * PGMR 0
* * * END-OF-REPORT * * *

*SEQ: 025

Figure 20 (Part 3 of 3). Example of ATTRTYPE Subject-Specific Report
Chapter 5. Defining Subject Categories

Assuming that a group of resources has been classified and that you want to set up a new subject category for those resources, you define a subject that resides in the Dictionary category called CATEGORY. The definition of this subject will control what data is to be collected for subjects belonging to that new category.

This chapter specifies how to define the name and attributes of an installation-defined category. You can define and report on the CATEGORY definition with standard commands, as described in this chapter, or with display forms, as described in Chapter 9, "Display Forms for Customized Entities" on page 165.

This chapter is a prerequisite to understanding the concepts presented in Chapter 3, "Planning for Dictionary Extensions and Customization" on page 49, and the information about attribute-types given in Chapter 4, "Defining Attribute-Types" on page 75.

Defining a Category Subject Name

The Dictionary subject name for a CATEGORY definition has the general form:

`(status,,category-name,0)`

The `status` of a subject category definition may be any of the standard codes while under development, or `**` when installed.

No subject code is specified in the subject name.

The `category-name` portion of the subject name can be up to 8 bytes long. If it is an EBCDIC string, the first character must be alphabetic. For DBCS users, this must be a uniform string.

All category definition subject names must have an occurrence-number of '0'.

Regarding the use of CATEGORY subject names:

- The category-name portion of the subject name becomes the name of the new subject category. It will be the name known to the general user, and the name used on display forms and in commands.
• You may establish aliases for the name of the category before it is installed. The aliases become alternative names (synonyms) for the category that the general user can use in commands or when using display forms. For DBCS users, this must be a uniform string.

• If you develop a new category at a standard status, you must place it in production 'P' status before the definition can be installed. The status code is changed to '**' when the new subject category is installed (that is, made operational for the general user). For example, a category named PROJECT in production status specified as a subject name on a command would be (P,PROJECT,0). If it was successfully installed, the name (*.PROJECT,0) would be used.

Category Definition Operands

Three types of operands specify the definition for a CATEGORY subject:

• Operands that describe the syntax for the user names of subjects in the installation-defined category. These are made "permanent" when the subject category is installed (with the exception of validation routines).

• Operands that are concerned with documentation text and that can be updated after the subject category is installed (DATE, EXPTXT, and Description and User Data text).

• Operands that identify attributes associated with this category and their assigned keyword names. The keywords are used to specify subject data for a subject in "category-name." The attributes and keyword names become "permanent" when the category is installed.

The first two types of operands are described below. They are specified either as values for Dictionary keywords in update commands or as fields on display forms. The third type is associated with the new category by relating subjects in the ATTRTYPE category to it. The definition of these attributes is covered in "Relating an Attribute-Type to a Category or Relationship-Type" on page 79 and "Relating Attribute-Types to a Category" on page 92.

The SBJCODE operand is used to specify the "subject-code," a number that will be used to distinguish that category from other categories in the Dictionary Extensibility data base.

Every installation-defined category must have a unique SBJCODE value associated with it. The code can be any integer in the range from 56 through 255; it is recommended, however, that you assign codes in the high end of the range (that is, from 255 down to 56). There is no default. Your installation may choose to assign a range of numbers to particular groups of developers, but you have a limit of '200' categories. You can specify this value at any time during the development of the category definition, but it is a required operand before the category can be installed.

The user name and this number become the category name and number pair that can appear on a Sign-On Header form when access control is activated. When the
category is installed, any authorized users will require the number and name pair specified in their access profiles.

The **NAMETYPE**, **MINNAME**, and **MAXNAME** values are used to specify the basic criteria that the *user-names* for the individual subjects in the category must satisfy.

The **NAMETYPE** operand specifies the basic criteria for the names. The values that can be specified are:

**A**
All alphabetic characters (‘A’ through ‘Z’, ‘@’, ‘#’, ‘$’). No DBCS allowed.

**C**
Alphabetic characters (‘A’ through ‘Z’, ‘@’, ‘#’, ‘$’, ‘0’ through ‘9’, ‘.’, ‘/’, ‘_’, ‘-’). For DBCS users, must be a uniform string.

**N**
Name format. Alphabetic, with the first character alphabetic. This is the default. For DBCS users, must be a uniform string.

**Z**
All numeric characters (‘0’ through ‘9’). No DBCS allowed.

The **MINNAME** and **MAXNAME** operands establish minimum and maximum permissible lengths for the user-names. **MINNAME** must be less than or equal to **MAXNAME**. The length can be in the range from ‘1’ to ‘31’. The default for **MINNAME** is ‘1′, and for **MAXNAME** is ‘31′.

The **VALRTN** operand specifies the name of a validation routine to check the subject names for new subjects being entered. Specify the 1- to 8-byte name exactly as the module will be named in the library used for validation routines. For DBCS users, DBCS characters are not acceptable as validation routine names and should not be used for the **VALRTN** operand.

These routines are not provided by the Dictionary, and are assumed to be written to the requirements of the validation. The validation performed is in addition to the basic syntax check implied by **NAMETYPE**, **MINNAME**, and **MAXNAME**. A routine may, for example, enforce certain conventions regarding the format of the user-names, the use of certain occurrence-numbers, or status codes. The programming requirements are described in “Requirements for Validation Routines” on page 176.
These points should be noted about the VALRTN operand:

- You can add or update the requirement for name validation to a category definition at any time, before or after installation of the category definition. If a validation routine name is present in an installed definition at the time a Dictionary subject name needs to be added, the routine will be invoked.

- The basic syntax checks of the user-name implied by NAMETYPE, MINNAME, and MAXNAME will be applied to the input subject names before the validation routine is invoked.

- The validation will be performed only for those subject names that are specified in an ADD or ADD__RELATIONSHIP command, as the new name in a COPY command, display form, or batch form, the bulk loading commands (PLI__IN, COBOL__IN, PSB__IN, and DBD__IN), as the TO name in a CHANGE__NAME command. Thus, if the validation routine is changed in such a way as to make some of the subject names already stored in the category incompatible, the user will still be able to obtain reports on those subjects, and to change their subject names to names that are acceptable to the new version of the routine. The user will, however, be prevented from adding new subjects with "invalid" names, or from establishing new relationships to those subjects.

The DATE, EXPTEXT, DESC, and USERn operands form the second type of operand. You may enter or update this information at any time during the development of the category definition, or after the category definition is installed.

The DATE operand may be used to associate a date with the category definition. Note that this attribute is not interpreted or maintained by the Dictionary, it is provided merely for documentation purposes. The value is specified as six numeric digits. There is no default. The value may be set or altered before or after installation.

The EXPTEXT operand identifies the set of User Data text that has been set aside to serve as explanation text for online users. The explanation of the purpose and use of the new category is entered as lines of text in the specified User Data set. This information is then readily available to the online user of display forms through the EXPLAIN processing option. Specify a number from '1' to '5'. There is no default.

The DESC and USERn ("n" is a value from '1' to '5') operands are specified like those for the ATTRTYPE subject in Chapter 4, "Defining Attribute-Types" on page 75. Alternatively, you may use the DESCRIPTION or USER DATA display forms to enter this text. If you have specified one set of User Data for "EXPLAIN" text, the lines you define are divided into frames that appear on successive displays of an EXTENSIBILITY EXPLANATION form. Each frame can hold up to 18 lines. For DBCS users, the text portion of both DESC and USERn can be a mixed string.

See Appendix C of OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference for a list of all the keywords that may be specified for a CATEGORY subject.
Using Commands for Category Definitions

The operands described in the previous section are specified as standard keywords with update commands. The keyword names and valid values are summarized for your convenience in Appendix C of OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference. You may use the abbreviation CAT for CATEGORY in commands and on display forms.

Adding and Updating a Definition

Subjects belonging to the CATEGORY subject category may be defined and (with certain restrictions) updated and deleted using update commands.

The ADD and CHANGE__IN commands may be used to define and update, and the DELETE command to delete, the definition for a subject category in any of the standard status levels.

For example, to add a CATEGORY named PROJECT:

ADD CATEGORY (E,,PROJECT,0) +
    DATE=100582 DESC=(1,'CURRENT PROGRAMMING PROJECTS') +
    SBJCODE=100 EXPTXT=1 +
    MAXNAME=5;

To change the User Data number where the explain text is to be defined:

CHANGE_IN CATEGORY (E,,PROJECT,0) +
    FROM EXISTING TO EXPTXT=5;

To establish an alias for the category:

ADD_RELATIONSHIP CATEGORY (E,,PROJECT,0) +
    HAS_ALIAS CATEGORY (E,,PROJ,0);

The DELETE_DATA command may be used to delete individual lines of the Description or User Data text associated with a subject category definition (at any status level).

Before the category is installed, the CHANGE__NAME command may be used to change the primary or an alias subject name for a subject category definition. You cannot use the CHANGE__NAME command to change the name of an installed subject category definition.

Using the COPY Command

The COPY command is used to create a copy of a subject category definition under a new subject name, to copy a category definition from one of the standard status levels to another (but not to the installed status), or to copy an "installed" category definition to create another definition at one of the standard levels.
Deleting a Subject Category Definition

An installation-defined category definition may be deleted using either the DELETE or DELETE__STRUCTURE command for any of the standard status codes.

The DELETE__STRUCTURE command may be used to delete the definition for a subject category, together with the definitions for any of its attribute-types that are not related to some other definition (that is, the standard DELETE__STRUCTURE rules will apply). However, it is likely that the attribute-types would be of general value to other developers of new categories, so that you might delete certain of the relationships before using the DELETE__STRUCTURE command.

There are restrictions when the category has been installed. The deletion of an installed category definition is prohibited unless:

- There are no subjects in that category represented in the Dictionary.
- There are no installed relationship-types associated with that category.

If the category is installed, you must use the DELETE__STRUCTURE command or the DELETE command with the AND ALIASES option to delete a CATEGORY subject.

Relating Attribute-Types to a Category

The definition of attributes that may be specified for the subjects in a category requires that appropriate attribute-types be related to the category definition. The three operands used to specify the relationship of the individual attribute-types to the CATEGORY subject are SEQ, KEYWORD, and REPEAT. Refer to "Relating an Attribute-Type to a Category or Relationship-Type" on page 79 for specifications. The SEQ operand is required on all relationship commands: that is, ADD__RELATIONSHIP, CHANGE__RELATIONSHIP__DATA, and DELETE__RELATIONSHIP. Altering relationships between the definition for an installed subject category and the definitions for the attribute-types related to that category is prohibited.
The standard ADD_RELATIONSHIP command is used to establish a relationship between a category definition and the definition for an attribute-type. The command requires the operand SEQ and the relationship keyword HAS. For example:

```
ADD_RELATIONSHIP CATEGORY (E,,PROJ,0) +
    HAS ATTRTYPE (E,,DATE,0) +
    SEQ=1 KEYWORD=STARTDAY;
```

The "inverse" relationship keyword used for ATTRTYPE-CATEGORY relationships (that is, the keyword used in relationship commands where the attribute-type is identified first) is APPLIES_TO. For example, to achieve the same result as the previous command:

```
ADD_RELATIONSHIP ATTRTYPE (E,,DATE,0) +
    APPLIES_TO CATEGORY (E,,PROJ,0) +
    SEQ=1 KEYWORD=STARTDAY;
```

The CHANGE_RELATIONSHIP_DATA command may be used to change any (or all) of the SEQ, KEYWORD, and REPEAT operand values associated with a category/attribute relationship.

The DELETE_RELATIONSHIP command is used to delete the relationship between a subject category definition and the definition for one of its associated attribute-types.

Any of the above relationship commands can specify an ATTRTYPE subject that has already been installed by including the '***' status code as part of the ATTRTYPE subject name.

**Procedure for Changing an Installed Category**

The developer of a category definition will probably want to clarify the documentation and add usability information to a category that has been installed. Modifications can be made to documentation portions of the definition. The DATE, Description, and User Data text can be added to or altered. The EXPTEXT operand can also be changed or added in order to specify a set of User Data text to be used for display form explanation frames. Also, a name validation routine can be specified when the definition is in installed status.

If you must make any other changes to the definition for an installed subject category (such as the set of attributes that may be specified for the subjects in that category), you must perform a **reinstallation process** to reestablish the category definition. This process lets you make changes to the category definition with automatic migration of subjects to the new definition.

The reinstallation process for a category definition is described in Chapter 8, "Modifying Installed Category and Relationship-Type Definitions" on page 135.

When you are developing the category definition, you should be aware of the constraints of the reinstallation process. Figure 21 on page 94 summarizes the various parts of a subject category definition and indicates whether the definition
operands are fixed for the life of the definition, can be modified with the
reinstallation process, or are alterable at any time.

<table>
<thead>
<tr>
<th>Operand</th>
<th>Can Be Modified Any Time</th>
<th>Can Be Modified with Reinstall</th>
<th>Fixed</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias name</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subject name</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESC</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPTEXT</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEYWORD</td>
<td></td>
<td>X</td>
<td></td>
<td>This applies to attributes that can be specified for subjects in the category.</td>
</tr>
<tr>
<td>MAXNAME</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINNAME</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAMETYPE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REPEAT</td>
<td></td>
<td>X</td>
<td></td>
<td>This applies to attributes that can be specified for subjects in the category.</td>
</tr>
<tr>
<td>SBICODE</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ</td>
<td></td>
<td>X</td>
<td></td>
<td>This applies to attributes that can be specified for subjects in the category.</td>
</tr>
<tr>
<td>USERN</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALRTN</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 21. Summary of Definition Operands for an Installation-Defined Category

If you do need to delete an installed category definition entirely, this requires:

- Deleting all subjects in that category from the Dictionary
- Deleting all installed relationship-types naming that category

Note that all instances of relationship-types are deleted by the action of deleting subjects participating in those instances.
Reporting Subject Category Development

Four types of reports are available for reporting the definitions of subject categories developed at your installation:

- Standard Dictionary subject-specific reports, of interest to a Dictionary user who develops such definitions
- A processing report produced by the action of the INSTALL command that confirms whether a category has been installed (or would be installed) successfully
- GUIDE reports used by the general user to obtain information about an installation-defined category that has been installed
- FORMAT reports detailing the control-information structure that has been generated for an installation-defined subject category

In addition, the Dictionary glossary REPORT and the SCAN command can be used to obtain reports on CATEGORY subjects.

The GUIDE report documents the category for the end user and includes the keyword names and values allowed for the attributes. Therefore, this report supplements the command and display form information given in OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference.

The GUIDE and FORMAT reports can be obtained at any point during the development of a category as optional output from an INSTALL command. Use the UPDATE=NO option to prevent actual installation of the category. These two reports and the processing report that is generated by the INSTALL command are described in Chapter 7, "Installing Categories and Relationship-Types" on page 111. If the category is installed, the GUIDE and FORMAT reports can be obtained directly with the REPORT command as administrative reports.

Subject Category Definition Reports

Dictionary subject-specific reports are obtained using the REPORT command, described in Chapter 6 of OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference. The REPORT command can be used in the usual manner to obtain full or partial information about the definition of an installation-defined category (or, more precisely, about the CATEGORY subject), either before or after it has been installed. To obtain display form equivalent reports, the NAME option can specify ALIAS for CATEGORY subject aliases, DESC for the Description text, or USERn for a set of User Data text. The standard DETAIL options for subjects in installation-defined categories on the REPORT command can be used to restrict the information to be included in the output report.

In addition, if the category has been installed, a list of the different installed relationship-types that have been defined for the category (that is, those relationship-types in which subjects in the category may participate) is also automatically included in the subject-specific report. If you want to obtain only this section of the report, use the DETAIL=RELTYPES option. For each
relationship-type, the following information is provided (report headings are given in parentheses):

- The relationship keyword (KEYWORD column) that is used to identify the relationship-type for the subjects in the category

- The name of the category (CATEGORY column) in which the related subjects reside

- The “inverse” relationship keyword (INV KEYWORD column) that is used to identify the relationship in commands when the subject in the related category is specified first

- An indication (in the SEQ column) of whether the instances of the relationship are sequenced (Y entry) or not (N entry)

- If the relationship is sequenced, the name of the sequence attribute (SEQ-ATTR column) for the relationship

- An indication (in the DIRECT column) of whether or not the relationship is “directed” and, if so, the sense of the direction:
  - 'N' indicates that the relationship is not directed.
  - 'D' indicates that the relationship is directed “downward” from the subjects in the category to the subjects in the related category.
  - 'U' indicates that the relationship is directed “upward” from the subjects in the category to the subjects in the related category.

**Example of a Subject-Specific CATEGORY Report**

Figure 22 on page 97 shows an example of a subject-specific report for an installed PGMR category. This is the standard Dictionary subject-specific report for an Extensibility subject, except for the special table of installed relationship-types for the category following the "RELATIONSHIP TYPES THAT MAY BE DEFINED" title.

You will notice that the ALIASES section shows two alias names for the CATEGORY subject:

- A standard alias (* PROGMR 0)
- A special (* *CAT 57) “alias”

The standard alias (PROGMR) provides a “synonym” that end users can use in commands, and so forth, in place of the primary category name. The second “alias” name is automatically created by the INSTALL processor. This “alias” is used by the Dictionary to locate installed CATEGORY definitions when only the subject code for the category is known (the occurrence qualifier in the special “alias” matches the SBJCODE for the category), and is not generally seen by end users.
DICTIONARY DATA BASE: EXT
CATEGORY: CATEGORY

NAME: * PGMR 0

ALIASES: * PROGRM 0
* CAT 57

DESCRIPTION:
001 SUBJECTS IN THIS CATEGORY REPRESENT PROGRAMMERS

ATTRIBUTES:

<table>
<thead>
<tr>
<th>DATE</th>
<th>101282</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAMETYPE</td>
<td>N</td>
</tr>
<tr>
<td>MAXNAME</td>
<td>31</td>
</tr>
<tr>
<td>EXPTXT</td>
<td>5</td>
</tr>
</tbody>
</table>

RELATIONSHIPS:

<table>
<thead>
<tr>
<th>HAS</th>
<th>ATTRTYPE</th>
<th>* JOB_TITLE 0</th>
<th>*SEQ: 005</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAS</td>
<td>ATTRTYPE</td>
<td>* OFFICE_ID 0</td>
<td>*SEQ: 010</td>
</tr>
<tr>
<td>HAS</td>
<td>ATTRTYPE</td>
<td>* INSTALLATION_LOCATION 0</td>
<td>*SEQ: 015</td>
</tr>
<tr>
<td>HAS</td>
<td>ATTRTYPE</td>
<td>* INTERNAL_PHONE_EXTENSION 0</td>
<td>*SEQ: 020</td>
</tr>
<tr>
<td>HAS</td>
<td>ATTRTYPE</td>
<td>* PROG_LANG_SKILL 0</td>
<td>*SEQ: 025</td>
</tr>
</tbody>
</table>

RELATIONSHIP TYPES THAT MAY BE DEFINED:

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>CATEGORY</th>
<th>INV KEYWORD</th>
<th>SEQ</th>
<th>SEQ-ATTR</th>
<th>DIRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>WROTE</td>
<td>MEMO</td>
<td>WRITTEN_BY</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>RESP_FOR</td>
<td>PROGRAM</td>
<td>BELONGS_TO</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>UPDATED</td>
<td>PROGRAM</td>
<td>UPDATED_BY</td>
<td>Y</td>
<td>MODDATE</td>
<td>N</td>
</tr>
<tr>
<td>WROTE</td>
<td>PROGRAM</td>
<td>WRITTEN_BY</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>AUTHORIZED</td>
<td>SYSTEM</td>
<td>AUTHORIZED</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Figure 22. Example of a Subject-Specific CATEGORY Subject Report
Chapter 6. Defining Customized Relationship-Types

If the interdependence of subjects in two categories (or in the same category) has been determined and you want to define a relationship-type between them, you must define a subject in the RELTYPE category. The subject and any related attribute-types constitute the complete definition of the relationship-type. This definition will control how pairs of subjects are to be related, and what relationship data may be entered.

You can define relationship-types that can be used to relate:

- Subjects in two different installation-defined categories
- Subjects in the same installation-defined category
- Subjects in an installation-defined category and subjects in a standard category

You cannot:

- Define new relationship-types to exist between pairs of subjects in standard categories
- Create any relationship-types involving the DDUSER category
- Define new relationship-type definitions to include any of the extensibility control information categories: ATTRTYPE, CATEGORY, RELTYPE, or STRTYPE

Defining a Relationship-Type Subject Name

The subject name for a relationship-type definition has the general form:

\[(\text{status,},\text{relationship-type-name},0)\]

The \text{status} of a relationship-type definition may be any of the standard status codes or the installed status "**".

No subject code is specified in the subject name.

The \text{relationship-type-name} portion of the subject name is the formal identifying name. The name can be from 1 to 31 bytes. If the name is EBCDIC, the first character must be alphabetic. For DBCS users, the name must be a uniform string.
All subject names for relationship-type definitions must have an occurrence-number of 0.

The following points should be noted regarding RELTYPE subjects:

- The RELTYPE subject name will not be the one that is used by the end user to relate a pair of subjects. Relationships will be established using a forward or an inverse name.

- You can establish alias names for a relationship-type definition prior to it being installed.

- You can use the CHANGE __NAME command to alter the user name of a RELTYPE subject or its aliases at any time.

- The subject will assume a status of 'S' after the definition has been installed. A definition may be developed at any of the test status levels, but must be placed in production status 'P' before the relationship-type can be installed.

Suggested Naming Convention

A relationship-type-name for a RELTYPE subject can implicitly reflect the connection between related subjects. For example, "DEPT/RUNS/PROJECT" as the name of the relationship-type, records that subjects in the installation-defined category "DEPT" (Department) have a relationship using the forward name "RUNS" with subjects in the installation-defined category "PROJECT." The RELTYPE subject name is:

(stat, DEPT/RUNS/PROJECT, 0)

An alias name is also established:

(stat, PROJECT/RUN__BY/DEPT, 0)

where stat is the status code. This name implicitly gives the inverse name RUN__BY when the relationship-type is thought of as subjects in the "PROJECT" category related to subjects in the "DEPT" category. The relationship-type names following this convention directly reveal the category and relationship keyword specifications used in Dictionary commands.

Relationship-types named with this convention also help specify the correct subject name on the REPORT command when requesting a GUIDE report for the relationship-type. The general user, knowing of an installation-defined category, can discover, for example, from the SUBJECT__RELATIONSHIPS display form, what relationship name is defined and what other category name is required for a relationship-type. A subject name can be directly constructed for use on the REPORT command for an administrative report.

This convention is not enforced by the Dictionary, though it is highly recommended.
Relationship-Type Definition Operands

Three types of operands that define a relationship-type are:

- Relationship keywords and properties that are made "permanent" when the relationship-type is installed.

- DATE, EXPTEXT, and Description and User Data text that can be updated after the relationship-type is installed.

- Keywords that identify relationship data for relationships of this type. The attributes and keyword names become "permanent" when the relationship-type is installed.

The first two types of operands are described below. The third type is associated with the new relationship-type by relating subjects in the ATTRTYPE category to it. The definition of these attributes is addressed in the sections "Relating Attribute-Types to a Relationship-Type" on page 104 and "Relating Attribute-Types to a Category" on page 92.

LCATNAME and RCATNAME specify the "left-hand" category name and the "right-hand" category name to be used in relationship specifications. The names can be from 1 to 8 alphameric characters long; the first character must be alphabetic. For DBCS users, the name must be a uniform string from 1 to 8 bytes. There is no default name for either category.

Although the values for LCATNAME and RCATNAME can be specified separately and at any time during the development of the relationship-type, they are required operands before the definition can be installed. In addition, the categories specified must be installed prior to installing the RELTYPE. Either the RCATNAME or LCATNAME must be an extensibility subject category.

FORNAME and INVNAME specify the forward and inverse keywords for a relationship. Each keyword name can be from 1 through 12 alphameric characters long; the first character must be alphabetic. For DBCS users, each keyword name must be a uniform string from 1 through 12 bytes long. There is no default name for either operand.

Although the values for FORNAME and INVNAME can be specified separately and at any time during the development of the relationship-type, they are required operands before the definition can be installed. The values for FORNAME and INVNAME need not be different.

If a subject in the LCATNAME category is specified first in a relationship command, then the forward name, identified by FORNAME, is used as the relationship keyword. If a subject in the RCATNAME category is specified first, then the inverse name, identified by INVNAME, is used as the relationship keyword.

The SEQOPT attribute indicates whether or not the instances of the relationship-type are sequenced. This is a required operand before the definition can be installed. Specify the single character 'Y' if the instances of the relationship are sequenced; specify the character 'N' if not.
The **SEQATTR** operand is used to identify the keyword of the attribute that sequences the instances of this relationship-type, thereby allowing more than one instance of the same relationship-type to exist between the same two subjects. You specify the keyword name used to identify the particular attribute-type together with **SEQOPT=Y**. DBCS users must enter a uniform string from 1 to 8 bytes for the keyword.

The sequencing attribute keyword can be entered at any time during the development of the relationship-type, and you do not need to have that attribute fully defined or related to the relationship-type definition at the time of entry. However, it is a required operand when **SEQOPT=Y** is specified, and the corresponding **ATTRTYPE** subject must be related to the **RELTYPE** subject for the relationship-type definition to be installed.

There are other constraints placed upon the **SEQATTR** specification because it is a special use of one of the attribute-types that is related to a **RELTYPE** definition. These are described in "Attribute-Types As Sequencing Attributes" on page 105. If **SEQOPT=N**, you do not have to enter any value for this operand.

The **DIRECTED** operand indicates whether or not the relationship-type is "directed." This is a required operand before the definition can be installed.

Specify one of two values, there is no default:

- **Y** The relationship is directed; the **LCATNAME** category is regarded as the superior category, and the **RCATNAME** is regarded as the subordinate category. The status rules apply. See Chapter 2 of *OS/VS DB/DC Data Dictionary Applications Guide* for the status rules.

- **N** Not directed.

If the relationship-type definition specifies an identical category name (**LCATNAME** and **RCATNAME** are the same), and the relationship keywords are the same (**FORNAME** is equal to **INVNAME**), this is called a **symmetrical relationship** and the following restrictions apply:

- The relationship-type cannot be sequenced; **SEQOPT=N** is required.

- The relationship-type cannot be directed; **DIRECTED=N** is required.

The **DATE**, **EXPTEXT**, **DESC**, and **USERn** operands form the second type of operands. You may enter or update this information at any time during the development of the relationship-type, or after the relationship-type is installed.

The **DATE** operand can be used to associate a date with the relationship-type definition. This is an optional operand with no default. You will notice that this operand is not interpreted or maintained by the Dictionary; it is provided merely for purposes of documentation. The value is specified as six numeric digits. You may alter the value of this operand, or supply a value, after the relationship-type has been installed.

The **EXPTEXT** operand may be used to identify the one set of User Data text that is set aside to serve as explanation text for display form users. Specify a number
from 1 to 5. This is an optional operand with no default. You may add a value for EXPTEXT after the relationship-type has been installed, or alter the value you previously entered.

The standard Description and User Data text are supported for relationship-type definitions. This text is entered in the same way as text for ATTRTYPE subjects. You may enter or update lines of text at any time during the development of the relationship-type or after the definition has been installed. DBCS users can enter or update mixed strings for DESC and USERn text.

For a list of keywords that may be specified for a RELTYPE subject, see Appendix C of OS/VS DB/DC Data Dictionary Terminal User’s Guide and Command Reference.

Using Commands with Relationship-Type Definitions

Adding and Updating a Definition

Subjects belonging to the RELTYPE category may be defined, updated, and deleted using standard update commands.

The standard ADD and CHANGE__IN commands may be used to define and update, and the DELETE command to delete, the relationship-type definition at any of the standard status levels.

For example, to add a RELTYPE named “DEPT/RUNS/PROJECT”:

```
ADD RELTYPE (E,'DEPT/RUNS/PROJECT',0) +
  DATE=101082 +
  DESC=(1,'DEPARTMENTAL CONTROL OF PROJECT BUDGET') +
  LCATNAME=DEPT RCATNAME=PROJECT +
  FORNAME=RUNS INVNAME=RUN BY +
  SEQOPT=Y SEQATTR=PRIORITY +
  EXPTTEXT=1;
```

The CHANGE__NAME command may be used to change the primary name or an alias of a relationship-type definition at any time before or after the RELTYPE subject has been installed. The CHANGE__NAME command does not permit the user to change the status code in the subject name. However, the COPY command can be used to make a copy of the definition at the new status level.

The DELETE__DATA command may be used to delete individual lines of the Description or User Data text associated with a relationship-type definition (at any status level). If you want to delete a specific value of a relationship-type definition, you must use the CHANGE__IN command.
Using the COPY Command

The COPY command may be used to create copies of a relationship-type definition under new subject names, to copy a relationship-type definition from one of the standard status levels to another (but not to ** status), or to copy an installed relationship-type definition to create another definition at one of the standard levels.

Deleting a Relationship-Type Definition

A relationship-type definition at any of the standard status levels may be deleted using either the DELETE or the DELETE__STRUCTURE command. Deletion of a relationship-type definition that has an “installed” status is prohibited as long as there exists a single instance of that relationship-type exists in the Dictionary.

The DELETE command using the UNRESTRICT option is invalid for installed relationship-types. If a RELTYPE subject has aliases, you can delete the relationship-type if you specify the AND ALIASES option.

Alternatively, you can use the DELETE__STRUCTURE command to delete the definition for a relationship-type, together with the definitions for any of its attribute-types that are not related to some other category or relationship-type definition (the standard DELETE__STRUCTURE rules will apply).

Relating Attribute-Types to a Relationship-Type

Attributes that record relationship data for an instance of a relationship-type are defined by relating appropriate attribute-types to the RELTYPE subject. These attributes are used to define relationship data for the relationship-type. The three operands used to specify the relationship of the individual attribute-types to the RELTYPE subject are SEQ, KEYWORD, and REPEAT. Refer to “Relating an Attribute-Type to a Category or Relationship-Type” on page 79 for specifications.

The standard ADD__RELATIONSHIP command is used to establish relationships for a relationship-type definition. The command requires the keyword SEQ and the relationship keyword HAS. HAS is the forward name when relating attribute-types to the RELTYPE subject.

For example:

```
ADD_RELATIONSHIP RELTYPE (E,,PROJECT/RUN_BY/DEPT,0) HAS +
  ATTRTYPE (E,,RANKING_ATTRIBUTE,0) +
  SEQ=1 KEYWORD=PRIORITY;
```

The inverse keyword is APPLIES__TO for use when the ATTRTYPE subject is named first in the command.

The CHANGE__RELATIONSHIP__DATA command may be used to change any (or all) of the SEQ, KEYWORD, and REPEAT values (before installation).
The DELETE__RELATIONSHIP command may be used to delete the relationship between a relationship-type definition and the definition for one of its associated attribute-types.

Any of the above relationship commands can specify an ATTRTYPE subject that has already been installed, by including the "**" status code as part of the ATTRTYPE subject name, but not if the RELTYPE definition has been installed.

The DELETE__RELATIONSHIP__DATA command cannot be used to delete groups of data for the installation-defined relationship-types. If you want to delete a particular attribute, use the CHANGE__RELATIONSHIP__DATA command.

Attribute-Types As Sequencing Attributes

If the specification of a relationship-type requires that instances of the relationship are to be sequenced, you must select an attribute to control the sequencing value.

The attribute must conform to several criteria:

- The chosen attribute-type must be related to the RELTYPE subject.
- The value of the SEQ operands specified on the relationship between the RELTYPE subject and the ATTRTYPE subject must be such that no other RELTYPE-HAS-ATTRTYPE relationship precedes it.
- The KEYWORD operand specified on the relationship must be the name specified for the SEQATTR operand of the relationship-type definition.
- The REPEAT operand cannot have a value greater than 1.
- The DATATYPE value for the attribute-type cannot be 'L' or 'Q'.
- The DATAMAX value for the attribute-type can be no larger than 8.

Procedure for Changing an Installed Relationship Type

The developer of a relationship-type will probably want to clarify the documentation and add usability information to the installed definition. Modifications can be made to documentation portions of the definition. The DATE, Description, and User Data text can be added to or altered. The EXPTEXT operand can also be changed or added in order to specify a set of User Data text to be used for display form explanation frames.

If you must make any other changes to the definition of an installed relationship-type (such as the set of attributes that may be specified for relationship data), you must use the reinstall process to reestablish the definition. After the definition is reinstalled in its modified form, all of the relationships that existed before the relationship-type was modified are retained.
The reinstallation process for a relationship-type definition is described in Chapter 8, "Modifying Installed Category and Relationship-Type Definitions" on page 135.

When you are developing the relationship-type definition you should be aware of the constraints of the reinstall process. Figure 23 summarizes the various parts of a relationship-type definition and indicates whether the definition operands are fixed for the life of the definition, can be modified during the reinstall process, or are alterable at any time.

<table>
<thead>
<tr>
<th>Operand</th>
<th>Can Be Modified Any Time</th>
<th>Can Be Modified with Reinstall</th>
<th>Fixed</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias name</td>
<td>X</td>
<td></td>
<td></td>
<td>You can change the name but cannot add or delete aliases.</td>
</tr>
<tr>
<td>Subject name</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESC</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIRECTED=N</td>
<td></td>
<td>X</td>
<td></td>
<td>You cannot change 'Y' because existing occurrences of this relationship may conflict with this new definition (for example, status rules).</td>
</tr>
<tr>
<td>DIRECTED=Y</td>
<td></td>
<td>X</td>
<td></td>
<td>You can change to 'N' because status rules were enforced when existing occurrences of this relationship were stored and those rules are nullified by this change.</td>
</tr>
<tr>
<td>EXPTEXT</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORNAME</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INVNAME</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEYWORD</td>
<td></td>
<td>X</td>
<td></td>
<td>This applies to attributes that can be specified for relationship-types.</td>
</tr>
</tbody>
</table>

Figure 23 (Part 1 of 2). Summary of Definition Operands for an Installation-Defined Relationship-Type.
<table>
<thead>
<tr>
<th>Operand</th>
<th>Can Be Modified Any Time</th>
<th>Can Be Modified with Reinstall</th>
<th>Fixed</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCATNAME</td>
<td></td>
<td>X</td>
<td></td>
<td>You can specify the primary or alias name of the category; you cannot change categories.</td>
</tr>
<tr>
<td>RCATNAME</td>
<td></td>
<td>X</td>
<td></td>
<td>You can specify the primary or alias name of the category; you cannot change categories.</td>
</tr>
<tr>
<td>REPEAT</td>
<td></td>
<td>X</td>
<td></td>
<td>This applies to attributes that can be specified for relationship-types.</td>
</tr>
<tr>
<td>SEQ</td>
<td></td>
<td>X</td>
<td></td>
<td>This applies to attributes that can be specified for relationship-types.</td>
</tr>
<tr>
<td>SEQATTR</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQOPT</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER:Rn</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 23 (Part 2 of 2). Summary of Definition Operands for an Installation-Defined Relationship-Type

If you do need to delete an installed relationship-type definition entirely, this requires that you first delete all instances of that relationship from the Dictionary.

If you delete an installed relationship-type definition and that relationship-type was used in an installed structure-type definition, the entry in the list of relationship-types is deleted automatically. You may want to investigate the presence of a relationship-type in structure-type definitions before the delete action. You can do this by:

- Requesting a subject-specific report for the RELTYPE subject with the operand RELATION=(CONTAINED_IN,STRTYPE)—this will give a list of all STRTYPE subjects that include the specified RELTYPE subject as part of the structure-type

- Accessing the “Selected” SUBJECT-RELATIONSHIPS display form, which will display a list of the STRTYPE subjects with a CONTAINED_IN relationship to the specified RELTYPE subject
Reporting Relationship-Type Development

Four types of reports are available for reporting the definitions of relationship-types developed at your installation:

- Standard Dictionary subject-specific reports, of interest to a Dictionary user who develops such definitions

- A processing report produced by the action of the INSTALL command that confirms whether a relationship-type has been installed (or would be installed) successfully

- GUIDE reports used by the end user to obtain information about an installation-defined relationship-type that has been installed

- FORMAT reports detailing the control-information structure that has been generated for an installation-defined relationship-type

In addition, the Dictionary glossary REPORT and SCAN commands can be used to obtain reports on the subjects in the RELTYPE category.

The GUIDE report documents the relationship-type for the end user and includes the related category names, the relationship keywords, and the keyword names and values permitted for the relationship data. Therefore, this report supplements the command and display form information given in OS/VS DB/DC Data Dictionary Terminal User’s Guide and Command Reference.

The GUIDE and FORMAT reports can be obtained at any point during the development of a relationship-type as optional output from an INSTALL command: Use the UPDATE=NO option to prevent the actual installation of the relationship-type. These two reports and the processing report that is generated by the INSTALL command, are described in Chapter 7, “Installing Categories and Relationship-Types” on page 111. If the relationship-type is installed, the GUIDE and FORMAT reports can be obtained directly with the REPORT command as administrative reports.

Relationship-Type Definition Reports

Dictionary subject-specific reports are obtained using the REPORT command, as described in Chapter 6 of OS/VS DB/DC Data Dictionary Terminal User’s Guide and Command Reference. The REPORT command can be used in the usual manner to obtain full or partial information about the definition of an installation-defined relationship-type (or, more precisely, about the RELTYPE subject), either before or after it has been installed. To obtain display form equivalent reports, the NAME option can specify ALIAS for RELTYPE subject aliases, DESC for the Description text, or USERn for a set of User Data text. The standard DETAIL options for subjects in installation-defined categories on the REPORT command can be used to restrict the information to be included in the output report.
Example of a Subject-Specific RELTYPE Report

Figure 24 shows an example of a subject-specific report for an installed “PGMR/UPDATED/PROGRAM” relationship-type. Note that this is the standard Dictionary subject-specific report for a subject residing in an “Extensibility” subject category. In addition to the attribute data for the RELTYPE subject, Figure 24 on page 109 shows a list of the relationships that have been established between the RELTYPE subject and the ATTRTYPE subjects defining the attributes (“relationship data”) that can be specified for instances of the relationship-type.

* * * * * * * DB/DC DATA DICTIONARY REPORT 11/11/83 16:04:04
SUBJECT SPECIFIC PAGE: 01

DICTIONARY DATA BASE: EXT CATEGORY: RELTYPE

NAME:
  * PGMR/UPDATED/PROGRAM 0

ALIASES:
  * PROGRAM/UPDATED_BY/PGMR 0

DESCRIPTION:
  001 PROGRAMMER - UPDATED - PROGRAM RELATIONSHIP

ATTRIBUTES:

  DATE: 102782
  RCATNAME: PROGRAM
  INVNAME: UPDATED_BY
  SEQATTR: MODDATE
  EXPIEXT: 5

  LCATNAME: PGMR
  FORNAME: UPDATED
  SEQOPT: Y
  DIRECTED: N

RELATIONSHIPS:

<table>
<thead>
<tr>
<th>HAS</th>
<th>ATTRTYPE</th>
<th>DATE</th>
<th>*SEQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAS</td>
<td>ATTRTYPE</td>
<td>FAILURE_REASON</td>
<td>0</td>
</tr>
<tr>
<td>HAS</td>
<td>ATTRTYPE</td>
<td>PROBLEM_DESCRIPTION</td>
<td>0</td>
</tr>
<tr>
<td>HAS</td>
<td>ATTRTYPE</td>
<td>CORRECTION_DESCRIPTION</td>
<td>0</td>
</tr>
<tr>
<td>HAS</td>
<td>ATTRTYPE</td>
<td>DATE</td>
<td>0</td>
</tr>
</tbody>
</table>

* * * END-OF-REPORT * * *

Figure 24. Example of a Relationship-Type Subject-Specific Report
Chapter 7. Installing Categories and Relationship-Types

This chapter explains the use of the INSTALL command to make category and relationship-type definitions, developed by your installation, operational for the general user. It also describes the reports that are produced as part of the command processing.

Preparation for Installing

Before Dictionary users can begin to define subjects in an installation-defined category, the definition for that category must be installed. Similarly, users cannot establish relationships involving installation-defined categories unless the appropriate relationship-type definition has been installed. The purpose of the INSTALL command is to make these definitions operational. An attribute-type definition is automatically installed the first time any one of the subject category or relationship-type definitions to which it is related is successfully installed.

It is possible to make certain types of changes to a definition even after it is installed. In fact, you carry out a reinstall process as discussed in Chapter 8, “Modifying Installed Category and Relationship-Type Definitions” on page 135. The INSTALL command has a special REPLACE option only used for already-installed definitions. When modifying a definition, there is the added consideration of any existing data that may be in the data base, created under the control of the “old” definition. Although the Dictionary offers some support to help migrate the data, there is always the chance that existing data may have to be converted or manipulated in some way, and this could entail detailed checking and reentry. Also, you would have to inform the end users of the change and coordinate the transition to the new definition.

To simplify the install process, this chapter leaves the discussion of these topics to Chapter 8, “Modifying Installed Category and Relationship-Type Definitions” on page 135. If you need to plan for the reinstall procedure, be sure you are familiar with the basic install process first.

An option of the INSTALL command permits you to validate any definition that is a candidate to be installed without actually installing it. At the same time, the command generates an installation processing report, and can produce prototype GUIDE and FORMAT reports as a part of the command processing.
These three reports add to the body of information about a category or relationship-type under development. Because the above reports can be obtained at any point in the development, they are useful:

- To judge the progress of the development
- To assess any problems that the general user would have using the GUIDE report if the category were installed
- To examine the distribution of attribute data when stored in the Extensibility data base, as indicated by the FORMAT report

After the category or relationship-type definition is installed, the GUIDE and FORMAT reports are obtained by means of the REPORT command.

The INSTALL Command

This command is responsible for the installation of customized entities, and validates the definitions of CATEGORY and RELTYPE subjects in addition to any ATTRTYPE subjects that are related to them. Successful validation allows the command to establish control information, internal to the Dictionary, so that the definitions of categories or relationship-types can become operational. The INSTALL command is available in both batch and online command processing.

The general format of this command is:

```
INSTALL    CATEGORY    subjectname  +
             RELTYPE
```

```
Required

Optional

$UPDATE=  NO       YES
REPLACE=  NO       YES
PROCRPT=  detail
         (detail1,...,detail4)

$RPT=
  FORMAT
  GUIDE
  ALL
  NONE
DEST=     L
         T
```

The options for the command are explained below:

**INSTALL**

The command verb. There is no abbreviated form.
CATEGORY | RELTYPE
A required operand that specifies the category of the definition to be installed. Specify one of two values:

CATEGORY For an installation-defined subject category. (The abbreviation CAT is valid.)

RELTYPE For an installation-defined relationship-type.

subjectname
A required operand that specifies the subject name of the category or relationship-type definition that is a candidate for installation. The status code of the subject must be 'P' when UPDATE=YES is specified. DBCS users must use a uniform string in the user name portion of the subject name.

UPDATE=NO | N | YES | Y
An optional operand that specifies the ultimate action of the command. The value NO requests that all validation processing be performed, but that the category or relationship-type definition not be installed. With UPDATE=NO, the attribute-types related to the definition are also validated, but their status is not changed. If UPDATE=NO, the definition will be validated, the "Installation Processing" report (and, optionally, a prototype of the GUIDE report) will be generated, but the definition will not be placed in installed status, and there will be no modification of the current Dictionary data. The value YES requests that the subject and all attribute-types that are not installed but are related to the subject be installed. If omitted, the default is NO.

REPLACE=NO | YES
An optional operand that specifies whether or not the definition to be installed is to replace an existing installed definition. If REPLACE=NO, the new definition will be added to the Dictionary. If REPLACE=YES, the new definition will replace the existing installed definition.

PROCRPT=detail | (detail1,...,detail4)
An optional operand that specifies the detail to be generated on the Installation Processing report. The valid values are:

VALIDATE Requests the "Validation Processing" section of the report. This section contains information about the validation of the definition.

ALTER Requests the "Category Alterations" or "Relationship-Type Alterations" section of the report. This may be specified only when REPLACE=YES has been specified. This section contains information on the differences between old and new definitions.

TRUNC Requests the "Conversion Processing" section of the report that identifies the data that will be truncated when the definition is installed. This may be specified only when REPLACE=YES.
CHECK Requests the "Conversion Processing" section of the report that identifies the data that will be invalidated when the definition is installed. This may be specified only when REPLACE=YES.

ALL All sections of the report will be generated.

NONE None of the above sections of the report will be produced.

Regardless of the values specified for the PROCRPT option, messages will be issued, and the "Installation Summary" section of the report will be produced. Up to four detail values may be specified on this operand.

If REPLACE=NO, PROCRPT=VALIDATE will be the default, if omitted. If REPLACE=YES, PROCRPT=(VALIDATE,ALTER,TRUNC) will be the default, if omitted.

RPT=FORMAT | ALL | GUIDE | NONE
An optional operand that specifies the report options. The meaning of the options are:

ALL GUIDE and FORMAT reports are to be produced.

GUIDE Only a GUIDE report is to be produced.

FORMAT Only a FORMAT report is to be produced.

NONE Neither report is to be produced.

If omitted, the default is FORMAT.

DEST=L | T
An optional operand used to specify the destination of the processing report generated by the INSTALL command. Valid values are:

L Line Printer

T Terminal

If omitted, the default is 'L' for batch and 'T' for online.

The UPDATE option of the INSTALL command permits the user to choose whether or not the category or relationship-type is to be installed. With UPDATE=NO specified, the same processing reports are produced as complete validation of the definitions is performed. In this way, you can assess the completeness of the definition and discover what defaults will be used.

The INSTALL command may be executed using a RELTYPE or CATEGORY subject that is at any of the standard status levels, but the definition must be in 'P' status before it can be installed.

If UPDATE=YES was specified on the command, and validation is successful, message DBD5500 will indicate that the installation of the category or relationship-type definition is completed. The status code of the definition and of
all related ATTRTYPE subjects that were not previously installed is changed from 'P' to '*'.

Validations Performed by the INSTALL Command

The validations performed by the INSTALL command are described below.

Validation for a Category Definition

The category definition is first checked to see if it exists as a CATEGORY subject.

Validation of the category definition involves a number of different checks and tests, including:

- If UPDATE=YES, the subject must have an assigned status of 'P'. Failure to do so causes the installation to fail. If the validation is successful, and UPDATE=YES has been specified, the definition is automatically placed in status '*'. If UPDATE=NO, the status code is not changed.

- The definition for the category is validated. A check is made to determine that the subject code specified by the SBJCODE operand has not been used by any previously installed category.

- Defaults are used for optional operands for which values have not been specified. The default for NAMETYPE is 'N', for MAXNAME is '31', for MINNAME is '1', and MINNAME must be less than or equal to MAXNAME.

- A check is made to determine that the keywords used to identify the different attributes related to the category are all unique names for that category.

- Each attribute-type definition related to the category that has not been previously installed is validated.

- If UPDATE=YES has been specified, the control information and other internal tables that will be used to process the definitions for subjects in the category are constructed and appended to the category definition and the definition is placed in status '*'. In this case, a special "alias" is created for the category ('*','CAT,nnn'), where nnn is the SBJCODE value; this is explained in "Example of a Subject-Specific CATEGORY Report" on page 96.

Should any aspect of the category definition (or any of the definitions for any of the attribute-types related to the category) fail to satisfy any of the validation criteria, installation of the category will continue for validation purposes only, but the category will not be installed.

Validation processing for a category definition does not verify the availability of the installation-written routine that may be specified for name checking. If you fail to make the routine available, commands used to enter new subjects will fail. Similarly, routines to validate attribute values are assumed to have been added to a library.
The instructions for adding routines to a library are in Chapter 10, "Validation Routines" on page 175. Results of validation will be reported in the "Validation Processing" section of the Installation Processing report.

Validation for a RELTYPE Definition

The relationship-type definition is first checked to see if it exists as a RELTYPE subject.

Validation of the relationship-type definition involves a number of different checks and tests, including:

- If UPDATE=YES, the subject must have an assigned status of 'P'. Failure to do so causes the installation to fail. If the validation is successful, and UPDATE=YES has been specified, the definition is automatically placed in status 'U'. If UPDATE=NO, the status code is not changed.

- The definition for the relationship-type is validated, and checks are made to see if all required operands have been specified.

- The two subject categories nominated in LCATNAME and RCATNAME are checked to ensure that they have been defined and installed. If either of these operands specifies standard Dictionary categories, the category is considered to have been installed. The RCATNAME and LCATNAME are checked to ensure that at least one is an installation-defined subject category.

- The FORNAME and INVNAME operands are checked to see if the combination of LCATNAME/FORNAME/RCATNAME or RCATNAME/INVNAME/LCATNAME has been specified for any previously installed relationship-type.

- The KEYWORD names used to identify relationship data are checked to see if they are unique under this relationship-type.

- Each attribute-type definition related to the relationship-type is validated.

- If an attribute-type is specified for the SEQATTR operand, the criteria for its use are checked against the criteria given in the section "Attribute-Types As Sequencing Attributes" on page 105.

- If UPDATE=YES has been specified, the control information and other internal tables used to process the data regarding the individual instances of the relationship-type are constructed and appended to the relationship-type definition, and the definition is installed.

Should any aspect of the relationship-type definition (or the definitions for any of the attribute-types that are related to it) fail to satisfy the validation criteria, the installation of the relationship-type will continue for validation purposes only, but the relationship-type will not be installed.

Results of validation will be reported in the "Validation Processing" section of the Installation Processing report.
Validation for an Attribute-Type Definition

Validation of the attribute-type definition involves a number of different checks and tests, including:

- A check to determine that all required attribute values have been specified. Defaults are used where appropriate.

- A check to determine that the values specified for the various attributes are complete and consistent.

Should the attribute-type definition fail to satisfy part of the validation, processing continues for the attribute-type. However, the attribute-type and its related category or relationship-type will not be installed.

Validation processing for an attribute-type definition does not verify the availability of the installation-written validation routine that may have been specified. If you fail to make the routine available, commands or display forms used to enter or update values requiring validation by that routine will fail during command processing.

The instructions for adding routines to a library are given in "Installing Validation Routines" on page 180.

Installation Processing Reports

When you execute an INSTALL command for a CATEGORY or RELTYPE subject, an installation processing report is generated. This section describes the "Validation Processing" and "Installation Summary" portions of the Installation Processing report.

The validation processing output is interspersed with numbered messages to indicate where defaults have been applied, what ATTRTYPE definitions are already installed, and what errors (if any) were encountered. The report organization is similar for category or relationship-type subjects, consisting of two parts: the validation of the attribute values for the subject definition followed by the validation of the related attribute-types.

The section headed "ATTRIBUTES" is a list of attribute-types headed by six columns. The first three columns identify the attribute-type subject name: status code (ST), user name (USERNAME), and occurrence (OCC).

The remaining columns specify how those attribute-types are to be used by the category or relationship-type. The KEYWORD column identifies the keyword name for the attribute, later to be used in update commands and on display forms. If the attribute is specified as repeating, the REP column will show a 3-digit number for the repeat factor. The order of the attributes in the list is determined by the sequence operand value shown in the SEQ column. You can determine the order in which attributes are given in reports, and the order in which they are stored in Dictionary data base segments, by examining the list sequence. The list also indicates the order in which the attribute keywords appear on the
EXTENSIBILITY SUBJECT display form for a category, or, for a relationship-type, the order on the RELATIONSHIP DATA form.

The installation summary is a series of messages that summarizes the outcome of the installation process. It indicates whether or not the attribute-types and the definition itself have been successfully installed.

Report for Installing a Subject Category

Figure 25 on page 119 is an example of a typical installation processing report for a simple category named "PGMR." Three specific items appearing within the report will be noted during a basic installation:

- No minimum user-name length (MINNAME) has been specified in the definition for the category. A default minimum length of '1' has been assumed (message DBD5543).

- No minimum value length (DATAMIN) has been specified for the JOB__TITLE attribute-type. A default minimum length of '1' has been assumed (message DBD5572).

- The DBD5573 messages associated with some attribute-type entries indicate that no validation-type (VALIDATE) has been specified for that attribute-type.

None of these items constitutes an "error" in the definition for the subject category or its related attribute-types; the messages in the installation summary confirm that all the attribute-types and the subject category itself have been successfully installed.
Figure 25. Example of a Subject Category Installation Report

Report for Installing a Relationship-Type

Figure 26 on page 120 is an example of a typical installation processing report for a simple relationship-type named PGMR/UPDATED/PROGRAM during a basic installation. You will notice that the DATE attribute-type has been used to define two different attributes of the relationship-type (the attribute MODDATE, and the attribute VERDATE), and was already installed at the time of the relationship-type installation processing.
INSTALLATION PROCESSING REPORT  11/11/85  13:45:00
PAGE: 0001

COMMAND OPTIONS:

INSTALLATION OF RELTYPE - PGMR/UPDATED/PROGRAM
UPDATE: YES          DEST: L
*REPLACE: NO          REPORT: GUIDE
*PROCRTL: ALL
NOTE: "*" MEANS DEFAULT VALUE TAKEN

VALIDATION PROCESSING:

ALIAS: PROGRAM/UPDATED_BY/PGMR

LEFT-HAND CATEGORY: PGMR
RIGHT-HAND CATEGORY: PROGRAM
FORWARD NAME: UPDATED
INVERSE NAME: UPDATED_BY
SEQUENCED: Y
SEQUENCE-ATTRIBUTE: MODDATE
DIRECTED: N
DATE: 042782       EXPTEXT: 5

ATTRIBUTES:

<table>
<thead>
<tr>
<th>ST NAME</th>
<th>OCC</th>
<th>SEQ</th>
<th>KEYWORD</th>
<th>REP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>000</td>
<td>002</td>
<td>MODDATE</td>
<td></td>
</tr>
</tbody>
</table>

* DATE      000 002 MODDATE
DBD5503 I ATTRTYPE ALREADY INSTALLED
* FAILURE_REASON 000 005 REASON
DBD5573 I NO VALIDATION SPECIFIED FOR ATTRIBUTE-TYPE
* PROBLEM_DESCRIPTION 000 010 PROBLEM
DBD5573 I NO VALIDATION SPECIFIED FOR ATTRIBUTE-TYPE
P CORRECTION_DESCRIPTION 000 015 CORRECTION
DBD5573 I NO VALIDATION SPECIFIED FOR ATTRIBUTE-TYPE
* DATE      000 020 VERDATE
DBD5503 I ATTRTYPE ALREADY INSTALLED

INSTALLATION SUMMARY:

DBD5504 I INSTALLATION SUCCESSFUL FOR ATTRTYPE FAILURE_REASON
DBD5504 I INSTALLATION SUCCESSFUL FOR ATTRTYPE PROBLEM_DESCRIPTION
DBD5504 I INSTALLATION SUCCESSFUL FOR ATTRTYPE CORRECTION_DESCRIPTION
DBD5500 I INSTALLATION OF RELTYPE P PGMR/UPDATED/PROGRAM COMPLETED

**** END OF INSTALLATION PROCESSING REPORT ****

Figure 26. Example of a Relationship-Type Installation Report
GUIDE Reports

After the definition of a new subject category has been completed and installed, the installation must inform its end users of the existence of the new category, and provide them with the information necessary to create and maintain subjects in that category. Similarly, after the definition of a new relationship-type has been installed, the installation must provide its end users with the information they require in order to establish and maintain instances of that relationship.

The GUIDE reports for CATEGORY and RELTYPE subjects provide end user information about the categories and relationship-types defined by the installation. Because they give keyword names, allowable values, and relationship keywords, they may be used to supplement the information provided for standard categories in OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference. These reports may be obtained using the Dictionary REPORT command after the category or relationship-type has been installed, or as an addendum to the processing report produced by the INSTALL command. In the latter case, a "prototype" GUIDE report for a category or relationship-type can be obtained, using the INSTALL command with the UPDATE=NO option specified.

Subject Category GUIDE Report

The subject category GUIDE report contains information about an installed category that users of the Dictionary need when they define subjects in the new category. The GUIDE report contains:

- The primary name and any "synonyms" for the subject category.
- The description text that has been stored in conjunction with the category definition.
- The minimum and maximum lengths and the required name format for the user-names of subjects in the category.
- If a validation routine has been provided by the installation to validate the subject names for subjects in the category, the name of that routine is also shown.
- A list of the attributes that may be specified for subjects in the category, including for each attribute:
  - The KEYWORD identifying that attribute.
  - The number of times that attribute may REPEAT for an individual subject.
  - The minimum and maximum lengths and the "data-type" of the values for that attribute.
  - Type of validation to be performed for input values.
  - If the attribute-type has a validation type of LIST or RNG, the corresponding list or range of valid values for the attribute is given.
• If the attribute-type has a validation type of RTN, the name of the installation-written routine used to validate values for the attribute.

• A list of the different types of relationships that subjects in the category may participate in, including, for each relationship-type given:
  
  – The relationship keyword that connects subjects in the principal category to other subjects (FORWARD NAME)

  – The name of the category in which the related subjects must reside (TO CATEGORY)

  – The name of the keyword to be used when the subject in the TO CATEGORY is specified first in relationship update commands (INVERSE NAME)

  – The code indicating whether instances of the relationship are sequenced or not (SEQOPT), and, if so, the name of the sequence attribute (SEQATTR)

  – An indication of whether or not the instances of the relationship are directed and, if so, the direction of that relationship. The DIRECTED column contains NONE if the instances are not directed. The value DOWN indicates that subjects in the related category are dependents of subjects in this category; value UP indicates the subjects in this category are dependents of those in the TO CATEGORY.

The list of relationship-types associated with a given subject category may expand as new relationship-types are defined and installed. Each installation should therefore establish some mechanism for either distributing new GUIDE reports for the subject categories affected when a new relationship-type is installed, or for informing interested end users so that they may request their own updated GUIDE reports for those categories.

**Example of a Subject Category GUIDE Report**

Figure 27 on page 124 shows an example of a typical GUIDE report for a simple Extensibility category for subjects representing programmers.

The principal name for the category is “PGMR”; a synonym is “PROGMR”.

The user-names for subjects in the category may be from 1 (MIN LENGTH) to 31 (MAX LENGTH) bytes long. The user name must conform to the NAMETYPE=N rules. No validation routine is provided for validating the subject names.

Five attributes may be specified for subjects in the PGMR category. The values for the TITLE, OFFICE, LOCATION, and PHONE attributes must satisfy the specified requirements as to length and data-type, but are not otherwise validated. The LANGSKIL attribute repeats five times; that is, values for LANGSKIL(1), LANGSKIL(2), LANGSKIL(3), LANGSKIL(4), and LANGSKIL(5) may be specified for a given PGMR subject. Each LANGSKIL value must be 3 to 5 alphanumeric characters long, and must be one of the following: COBOL, PL/I, RPG, ASM, or DL/I.
As shown in the RELATIONSHIP TYPES list in the report, subjects in the PGMR category may participate in five different types of relationships (this list would usually appear on the second page of the printed GUIDE report, as indicated by the second page header in the figure):

“WROTE” relationships to subjects in the installation-defined MEMO category (the inverse keyword is “WRITTEN__BY”).

“RESP__FOR,” “UPDATED,” or “WROTE” relationships to subjects in the standard PROGRAM category. (The inverse keywords are “BELONGS__TO,” “UPDATED__BY,” and “WRITTEN__BY,” respectively.)

“AUTHORIZED” relationships to subjects in the standard SYSTEM category. The inverse keyword is also “AUTHORIZED.”

The PGMR/UPDATED/PROGRAM type of relationship is sequenced on the MODDATE attribute; users would therefore be permitted to establish more than one instance of that type of relationship between the same PGMR and PROGRAM subjects (but with different MODDATE values) to record that a particular programmer has updated the same program more than once (but on different dates).
SUBJECTS IN THIS CATEGORY REPRESENT PROGRAMMERS

**CATEGORY**:
PGMR

**ALIASES**: PROGMR

**DESCRIPTION**: None provided.

**SUBJECT NAME RULES**:  
- **NAME TYPE**: N  
- **MIN LENGTH**: 01  
- **MAX LENGTH**: 31  
- **VAL ROUTINE**: None provided.

**ATTRIBUTES**:  

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>REPEAT</th>
<th>LENGTH</th>
<th>DATA</th>
<th>VALIDATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE</td>
<td>00</td>
<td>1 24</td>
<td>C</td>
<td>NONE</td>
</tr>
<tr>
<td>OFFICE</td>
<td>00</td>
<td>4 4</td>
<td>C</td>
<td>NONE</td>
</tr>
<tr>
<td>LOCATION</td>
<td>00</td>
<td>5 24</td>
<td>C</td>
<td>NONE</td>
</tr>
<tr>
<td>PHONE</td>
<td>00</td>
<td>4 4</td>
<td>Z</td>
<td>NONE</td>
</tr>
<tr>
<td>LANGSKIL</td>
<td>05</td>
<td>3 5</td>
<td>C</td>
<td>LIST COBOL,PL/I,RPG,ASM,DL/I</td>
</tr>
</tbody>
</table>

**RELATIONSHIP TYPES**:  

<table>
<thead>
<tr>
<th>FORWARD NAME</th>
<th>TO CATEGORY</th>
<th>INVERSE NAME</th>
<th>SEQOPT</th>
<th>SEQATTR</th>
<th>DIRECTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>WROTE</td>
<td>MEMO</td>
<td>WRITTEN_BY</td>
<td>N</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>RESP_FOR</td>
<td>PROGRAM</td>
<td>BELONGS_TO</td>
<td>N</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>UPDATED</td>
<td>PROGRAM</td>
<td>UPDATED_BY</td>
<td>Y</td>
<td>MODDATE</td>
<td>NONE</td>
</tr>
<tr>
<td>WROTE</td>
<td>PROGRAM</td>
<td>WRITTEN_BY</td>
<td>N</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>AUTHORIZED</td>
<td>SYSTEM</td>
<td>AUTHORIZED</td>
<td>N</td>
<td>NONE</td>
<td>NONE</td>
</tr>
</tbody>
</table>

* * * END-OF-REPORT * * *

**Figure 27. Example of a Subject Category GUIDE Report**

124 Administration and Customization Guide
Relationship-Type GUIDE Reports

The relationship-type GUIDE report contains information about a relationship-type that users of the Dictionary need when defining instances of the new relationship-type. The GUIDE report contains information on the specified relationship-type:

- The Dictionary subject name and any aliases for the relationship-type definition

You will notice that these subject names are not usually "visible" to end users. Rather, they use the forward and inverse keywords for the relationship-type when using commands or the displays to establish or update instances of the relationship.

- Description text stored with the relationship-type definition

- The names of the "left-" and "right-hand" categories for the subjects that may participate in the relationship-type

- The forward name and inverse name identified for the relationship-type for the subjects in those categories

- The SEQUENCE option indicating whether instances of the relationship-type are sequenced and, if so, the name of the sequence attribute

- The DIRECTED option indicating whether instances of the relationship-type are directed

- A list of the attributes that may be specified for individual instances of the relationship-type

For each attribute specifying relationship data:

- The KEYWORD identifying that attribute

- The number of times the attribute may be repeated for an individual relationship instance

- The minimum and maximum lengths and the data-type of the values for the attribute

- The validation type for values of that attribute

- If the attribute-type has a validation type of LIST or RNG, the corresponding list or range of valid values for the attribute

- If the attribute-type has a validation type of RTN, the name of the installation-written validation routine used to validate values for the attribute
Example of a Relationship-Type GUIDE Report

Figure 28 on page 127 shows an example of a typical GUIDE report for a simple "programmer-updated-program" relationship-type.

The user-name for the RELTYPE subject defining the relationship-type is "PGMR/UPDATED/PROGRAM"; an alias of "PROGRAM/UPDATED__BY/PGMR" has also been established. Because the relationship-type has been installed, the status code in both subject names is '1', and the occurrence qualifier is '0'.

Relationships of this type are to be established between subjects in the installation-defined PGMR category and subjects in the standard Dictionary category, PROGRAM.

The "forward" keyword for the relationship is "UPDATED" (as in PGMR x UPDATED PROGRAM y); the "inverse" keyword is "UPDATED__BY" (as in PROGRAM y UPDATED__BY PGMR x).

The instances of the relationship in which a particular PGMR or PROGRAM subject participates are sequenced on the MODDATE attribute values.

The instances of the relationship are not "directed."

Five attributes (or items of "relationship data") may be specified for instances of the relationship. The values for each of the five attributes must satisfy the specified requirements as to length and data-type, but are not otherwise validated. Also, because MODDATE serves as the "sequence attribute" for the relationship-type, a value for the MODDATE attribute must be specified in every command addressing an instance of this type of relationship.
FORMAT Reports

The special FORMAT reports that are available for CATEGORY and RELTYPE subjects provide the necessary information about the storage format that is calculated by the INSTALL command processor for a particular category or relationship-type.

Before describing these reports, some background information about how data is stored is appropriate. Following the report descriptions, an example is given to illustrate how you can optimize the use of storage.

How Extensibility Data Is Stored

During the installation processing for a subject category or relationship-type definition, the format in which the attribute data associated with the individual subjects in that category (or instances of that relationship) will be stored in the Dictionary "Extensibility" data base (DDSEXT) is determined and recorded in "control information" tables. These tables are stored in association with the installed CATEGORY or RELTYPE subject, and are subsequently referred to by
the Dictionary code when processing commands or generating reports for the subjects in the category or instances of the relationship.

Storage for Attribute Values

The attribute data for subjects in installation-defined subject categories is stored in one or more 121-byte EXTATTR segments. The first byte of each EXTATTR segment is used to store a 1-byte binary segment occurrence identification (ID). The remaining 120 bytes are used to store attribute data.

The relationship data associated with an instance of a relationship between two subjects in installation-defined categories is stored in the EXTEXT logical child segment representing the relationship, and subordinate EXTEXTA segments (if required). The attribute data for relationships between subjects in an installation-defined category and subjects in one of the standard categories is stored in the equivalent EXTxxx and EXTxxxA segments, where xxx is:

DTE
   For relationships to subjects in the standard ELEMENT category

SEG
   For relationships to subjects in the standard SEGMENT category

DBS
   For relationships to subjects in the standard DATABASE category

PCB
   For relationships to subjects in the standard PCB category

SYS
   For relationships to subjects in one of the standard categories stored in the Dictionary "System" data base

For purposes of this discussion, the EXTEXT and EXTEXTA segments will be referenced, recognizing that the other EXTxxx and EXTxxxA segments have the same format, and are used in a similar manner.

The EXTEXT segment contains three types of data:

- Bytes 1 through 54 and 63 through 91 contain fields that are used to identify the relationship, and are initialized and maintained by the Dictionary code.

- Bytes 55 through 62 are used to store the value for the sequence attribute associated with an instance of a sequenced relationship.

- Bytes 92 through 121 are used to store attribute data other than the value for the sequence attribute.

If the attribute data for a relationship exceeds the space available in the EXTEXT segment, it is stored in one or more occurrences of the subordinate EXTEXTA segment. The EXTEXTA segment has the same format as the EXTATTR segment: a 1-byte binary segment occurrence ID, followed by a 120-byte data area.
When determining the format for the attribute data to be stored for the subjects in an installation-defined category, the INSTALL processor executes a simple algorithm, summarized as follows:

- The attribute-types (ATTRTYPEs) related to the CATEGORY are processed in SEQ-value order.

- For each successive attribute, a field DATAMAX bytes long (that is, a field long enough to store the longest possible value for the attribute) is allocated at the next available offset in the "current" EXTATR segment occurrence, starting at OFFSET=1 in the EXTATR segment occurrence with ID=1. The fields for successive attribute values are allocated "compactly" within the EXTATR segment occurrences.

- In the case of an attribute that repeats 'N' times, 'N' successive DATAMAX-bytes fields are allocated to store the values for each of the 'N' instances of the attribute.

- An EXTATR segment occurrence is considered "filled" when the remaining space is insufficient to fully contain the field for storing the values for the next sequential attribute (or attribute instance). Attribute values are not split across EXTATR segment occurrences.

- When an EXTATR segment occurrence has been fully allocated, the occurrence ID is increased and the following attributes allocated to that occurrence (starting again at OFFSET=1).

A similar approach is used in determining the format for the attribute data to be stored for the instances of an installation-defined relationship-type, except that:

- The field for storing the values for the sequence attribute (if the relationship is sequenced) is allocated at OFFSET=54 in the EXTEXT segment.

  Note that the sequence attribute must have DATAMAX less than or equal to 8.

- The fields for storing the values for the other attributes are allocated, starting at OFFSET=91 in the EXTEXT segment, continuing to successive occurrences of the EXTExTA segment (if required).

- The ID for the EXTEXT segment is 0. The IDs for EXTExTA segments begin with '1'.

Storage of Validation Values

During the processing of each attribute-type definition, the INSTALL processor also extracts the validation specifications for the values for that attribute and places them in a separate "validation table." This "validation table" is stored in one or more 256-byte ECIVTAB segments associated with the installed CATEGORY or RELTYPE subject, and is subsequently referred to by the Dictionary code when processing commands to add or change the values for that attribute.

The first byte of each ECIVTAB segment contains a 1-byte binary segment occurrence ID; the remaining 255 bytes are used to store the validation...
information. The information that is stored varies according to the type of validation to be performed:

- For attributes whose values are to be validated against a “list” (VALIDATE=LIST), each valid value (VALLIST(n)) for the attribute is formatted and stored in a separate DATAMAX-byte field.

- For attributes whose values are to be validated against a “range” (VALIDATE=RNG), the low and high values defining the range (VALRNGL and VALRNGH) are formatted and stored in a pair of DATAMAX-byte fields.

- For attributes whose values are to be validated by an installation-written routine (VALIDATE=RTN), the name of the routine (VALRTN) is stored in an 8-byte field.

No validation information is stored for those attributes whose values require no special validation processing.

**CATEGORY and RELTYPE FORMAT Reports**

While not of interest to the general end users of the Dictionary, information about the format in which the attribute data for the subjects in an installation-defined category (or the instances of an installation-defined relationship-type) is stored may be required by individuals who want to develop special GIS/VS or other “query” applications to run against the Dictionary data bases. Furthermore, such information may be required in order to diagnose any errors detected in the Dictionary function.

The special CATEGORY and RELTYPE FORMAT reports provide the necessary information about the storage format that has been calculated by the INSTALL processor for a particular category or relationship-type. These reports may be obtained using the Dictionary REPORT command after the category or relationship-type has been installed, or as an addendum to the processing report produced by the INSTALL command. Note that, in the latter case, a “prototype” FORMAT report for a category or relationship-type may be obtained using the INSTALL command with the UPDATE=NO option specified.

The FORMAT report for a category or relationship-type has almost the same format and content as the corresponding GUIDE report discussed above, except that:

- The Description text for the CATEGORY or RELTYPE is omitted.

- The CATEGORY FORMAT report does not include the table of relationship-types for subjects in the category.
The table of attributes for the subjects in the category (or the instances of the relationship) contains a somewhat different set of information, and is the principal object of importance in the FORMAT report. For each attribute, the table lists:

- The KEYWORD identifying the attribute
- The instance number (INST) identifying the specific instance of a repeating attribute
  This field is zeros for an attribute that does not “repeat.”
- The length of the field (in bytes) that has been allocated for storing the values for the attribute or attribute instance (SPACE)
- The DATA TYPE of the values for the attribute
- The ID of the EXTATR or EXTEXTA segment occurrence in which the values for the attribute are stored
  This field is blank for “relationship data” attributes that are stored in the EXTEXT segment.
- The offset (OFFS) within that segment of the field allocated for storing the attribute values
- Information about the TYPE of validation applied to the values for the attribute, and the number (NO) and location of the specifications (ID and OFFS) to be used by the Dictionary in performing that validation

Example of a Subject Category FORMAT Report

Figure 29 on page 132 shows an example of a typical FORMAT report for a simple installation-defined subject category, the PGMR category. The corresponding GUIDE report is shown in Figure 27 on page 124.

Note that five separate entries are shown in the ATTRIBUTES table for the repeating LANGSKIL attribute. There is a separate entry for each instance of the attribute, indicating the ID of the EXTATR segment occurrence (in this case, ID=1 for all instances) and the OFFSET within that segment occurrence of the field allocated for storing the values specified for that instance.
Example of a Subject Category FORMAT Report

Figure 29. Example of a Subject Category FORMAT Report

Example of a Relationship-Type FORMAT Report

Figure 30 on page 133 shows an example of a typical FORMAT report for an installation-defined relationship-type. The PGMR/UPDATED/PROGRAM relationship GUIDE report is shown in Figure 28 on page 127.

Because this relationship-type is between subjects in an installation-defined category (PGMR) and subjects in the standard PROGRAM category (which is part of the Dictionary “System” data base), the attribute data for the instances of the relationship will be stored in EXTSYS and EXTSYSA segments.

Note that the MODDATE attribute is the sequence attribute for the relationship, and that the values for the attribute are stored at OFFSET=54 in the EXTSYS relationship segment.

The values for the REASON attribute are stored at OFFSET=91 in the EXTSYS segment; the values for the PROBLEM, CORRECTN, and VERDATE attributes occupy occurrences of the subordinate EXTSYSA segment with ID=1, ID=2, and ID=3, respectively.
Figure 30. Example of a Relationship-Type FORMAT Report

Optimizing Attribute Data Storage

Your principal concern in establishing the ordering or sequence for the attributes associated with a subject category or relationship-type should be that of ensuring that the attributes and their values appear in logical or convenient groupings in reports or on the display forms. But within this overall constraint, some consideration may also be given to selecting an ordering that minimizes unused space in the EXTATR, or EXTxxx and EXTxxxA segments that will be used to store the attribute data.

Because the algorithm used by the INSTALL processor allocates space for the attribute values sequentially, switching to a new segment occurrence each time the value for an attribute will not “fit” into the current occurrence may cause one or more bytes of space at the end of each data segment occurrence to not be allocated.

For example, the FORMAT report for the PGMR-UPDATED-PROGRAM relationship-type shown in Figure 30 indicates that:

- Six bytes will be unused at the end of the EXTSYS segment.

The REASON attribute value will occupy the first 24 bytes of the 30-byte data area, and the 120-byte value for the PROBLEM attribute must occupy a separate EXTSYSA occurrence.
The value for the VERDATE attribute will occupy the first 6 bytes of the
EXTSYSA occurrence with ID=3, the remaining 114 bytes of that segment
being unused and effectively "wasted."

If the attributes were reordered so that the VERDATE attribute occurs either
before or immediately after the REASON attribute (the MODDATE attribute,
being the sequence attribute for the relationship, must occur first), then fields for
storing the values for both the REASON and the VERDATE attributes would be
allocated in the EXTSYS segment, thereby avoiding the necessity of allocating
EXTSYSA segment ID=3. This latter situation is illustrated in the FORMAT
report for the relationship-type shown in Figure 31.

You should plan to obtain the "prototype" FORMAT reports for a subject
category or relationship-type before the category or relationship-type is actually
installed. The SPACE, ID, and OFFS entries for each attribute (instance) may
then be analyzed to determine whether the attributes might be reordered to
minimize unused storage space.

* * * * * * *
DB/DC DATA DICTIONARY REPORT 11/11/85 16:05:29
RELTYPE: FORMAT
RELTYPE: FORMAT
RELTYPES: PGMR/UPDATED/PROGRAM
ALIASES: PROGRAM/UPDATED_BY/PGMR

RELATIONSHIP TYPE RULES:
LEFT-HAND CATEGORY: PGMR
RIGHT-HAND CATEGORY: PROGRAM
FOREWARD NAME: UPDATED
INVERSE NAME: UPDATED_BY
SEQUENCE OPTION: YES
SEQUENCE ATTRIBUTE: MODDATE
DIRECTED: NO

DATE: 042785

ATTRIBUTES:

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>DATA TYPE</th>
<th>SEGMENT ID</th>
<th>OFFS</th>
<th>SEGMENT TYPE</th>
<th>SEGMENT ID</th>
<th>OFFS</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODDATE</td>
<td>00</td>
<td>6</td>
<td>Z</td>
<td>54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REASON</td>
<td>00</td>
<td>24</td>
<td>Q</td>
<td>91</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VERDATE</td>
<td>00</td>
<td>6</td>
<td>Z</td>
<td>115</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROBLEM</td>
<td>00</td>
<td>120</td>
<td>Q</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CORRECTN</td>
<td>00</td>
<td>120</td>
<td>Q</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* * * END-OF-REPORT * * *

Figure 31. Example of FORMAT Report with Optimized Storage
Chapter 8. Modifying Installed Category and Relationship-Type Definitions

Before modifying categories and relationship-type definitions, you should read Chapter 7, "Installing Categories and Relationship-Types" on page 111 to understand how they are installed. The INSTALL command used to modify installed categories and relationship-types is also discussed in that chapter.

After you have installed a definition of an extensibility control information category or relationship-type, the installed definition is protected by the Dictionary from further changes by being placed in a special status with a code of \*\*1.

Because the install process creates internal control information, any changes to an installed definition would impact that control. Indeed, data may already exist for a customized entity, formatted and stored in the Dictionary data base in conformance with that control information. Also, your installation requirements probably include a careful review of any definition that has been developed and many of the services provided by the install process, such as Installation Processing reports, are of value in documenting the development.

For these reasons, the modification of a definition is actually a repeat of the installation process. Part of that process highlights the type of changes you have made. Another part gives information about the effect of the changes upon existing data. Because there are these kinds of differences, the procedure for modifying already installed definitions is described as the reinstall process.

The Dictionary allows you to modify a definition of a category or relationship-type that your installation has already installed. There are a number of reasons why you might want to do this:

- The definition of an installed category requires modification of its attributes; there are omissions or the data specifications are in error.

- The attributes for an installed entity are causing confusion to the users, possibly because the data format is inappropriate or the keyword name is not clear.

- Experience with data entry using the display forms has shown some fields to be inconveniently placed.

- A problem has occurred regarding the validation of subject names for an installed category or a definition's attribute validation is incorrect.
- The definition of an installed relationship-type requires changes in the relationship data.

- You may want to add new attributes.

- You may want to remove attributes that are no longer needed.

- You may want to reorder the sequence of the attributes for a category or relationship-type.

The Dictionary can help you respond to these problems and accomplish such modification as is necessary. There are some restrictions in the types of alterations you can make, but the majority of the above situations can be resolved.

Planning for Different Types of Changes

There are two general areas where you can make changes to category or relationship-type definitions that have already been installed:

- Changes to the definition operands

- Changes to the attributes, which may be specified for subjects in the category (subject data) or for instances of the relationship-type (relationship data)

The changes you can make to relationship-type definitions are not as extensive as those for installation-defined category definitions. For either subject data or relationship data, the allowed changes are the same.

Changing a Category's Definition Operands

You can change most of the specifications associated with the CATEGORY subject except the primary name and the CATEGORY subject code (SBJCODE). The primary name and subject code must remain the same.

The changes you can make for the CATEGORY subject are:

- The values of the DATE or EXPTEXT attributes.

- The Description and User Data text associated with an installed CATEGORY subject.

- Add or delete CATEGORY alias names.

- The subject name validation specifications for name type (NAMETYPE), minimum (MINNAME), and maximum (MAXNAME) lengths of the subject name.

- Add, change, or delete the specifications for name validation by exit routines (VALRTN) associated with an installed CATEGORY subject.
A summary of the definition operands you can change with the reinstall process are shown in Figure 32.

<table>
<thead>
<tr>
<th>Operand</th>
<th>Can Be Modified Any Time</th>
<th>Can Be Modified with Reinstall</th>
<th>Fixed</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias name</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subject name</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESC</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPTEXT</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEYWORD</td>
<td>X</td>
<td>X</td>
<td></td>
<td>This applies to attributes that can be specified for subjects in the category.</td>
</tr>
<tr>
<td>MAXNAME</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINNAME</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAMETYPE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REPEAT</td>
<td>X</td>
<td>X</td>
<td></td>
<td>This applies to attributes that can be specified for subjects in the category.</td>
</tr>
<tr>
<td>SBJCODE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ</td>
<td>X</td>
<td></td>
<td></td>
<td>This applies to attributes that can be specified for subjects in the category.</td>
</tr>
<tr>
<td>USERn</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALRTN</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 32. Summary of Definition Operands for an Installation-Defined Category

Changing Attributes Specified for Subjects in the Category

The changes that may be made for attributes that may be specified for subjects in the category are:

- Adding one or more new attributes
- Deleting one or more existing attributes
- Changing the data type (DATATYPE), minimum (DATAMIN), or maximum (DATAMAX) lengths for the values of one or more attributes
- Changing the validation specifications for the values of one or more attributes
- Increasing or decreasing the number of instances of a repeating attribute
- Reordering the sequencing of attributes
- You may not change the KEYWORD attribute.
Changing a Relationship-Type's Definition Operands

The changes you can make for the RELTYPE subject are:

- The values of the DATE and EXPTEXT attributes.

- The Description and User Data text associated with an installed RELTYPE subject.

- The names specified for LCATNAME and RCATNAME attributes. You can use a different alias name or use the primary name in place of an alias name. You cannot change the categories.

- The directed attribute. You can change DIRECTED=Y to DIRECTED=N. You cannot change DIRECTED=N to DIRECTED=Y.

- RELTYPE primary subject name and alias name.

Changes you cannot make for the RELTYPE subject are:

- The two keywords that identify the relationship-type (FORNAME and INVNAME).

- The sequencing specifications for the relationship-type (SEQOPT). A relationship-type that is sequenced must remain sequenced; a relationship-type that is unsequenced must remain unsequenced. For a sequenced relationship-type, the name of the sequence attribute (SEQATTR) may not be changed.

- The directed attribute. You cannot change DIRECTED=N to DIRECTED=Y.

A summary of the definition operands you can change with the reinstall process are shown in Figure 33 on page 139.
<table>
<thead>
<tr>
<th>Operand</th>
<th>Can Be Modified Any Time</th>
<th>Can Be Modified with Reinstall</th>
<th>Fixed</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias name</td>
<td>X</td>
<td></td>
<td></td>
<td>You can change or delete aliases anytime. You can only add new aliases with REINSTALL.</td>
</tr>
<tr>
<td>Subject name</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESC</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIRECTED=N</td>
<td></td>
<td>X</td>
<td></td>
<td>You cannot change to 'Y' because existing occurrences of this relationship may conflict with this new definition (for example, status rules).</td>
</tr>
<tr>
<td>DIRECTED=Y</td>
<td></td>
<td>X</td>
<td></td>
<td>You can change to 'N' because status rules were enforced when existing occurrences of this relationship were stored, and those rules have been nullified by this change.</td>
</tr>
<tr>
<td>EXPTEXT</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORNAME</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INVNAME</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEYWORD</td>
<td></td>
<td>X</td>
<td></td>
<td>This applies to attributes that can be specified for relationship-type.</td>
</tr>
<tr>
<td>LCATNAME</td>
<td></td>
<td>X</td>
<td></td>
<td>You can specify the primary or alias name of the category; you cannot change the category.</td>
</tr>
<tr>
<td>RCATNAME</td>
<td></td>
<td>X</td>
<td></td>
<td>You can specify the primary or alias name of the category; you cannot change the category.</td>
</tr>
</tbody>
</table>

Figure 33 (Part 1 of 2). Summary of Definition Operands for an Installation-Defined Relationship-Type
<table>
<thead>
<tr>
<th>Operand</th>
<th>Can Be Modified Any Time</th>
<th>Can Be Modified with Reinstall</th>
<th>Fixed</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPEAT</td>
<td></td>
<td>X</td>
<td></td>
<td>This applies to attributes that can be specified for relationship-type.</td>
</tr>
<tr>
<td>SEQ</td>
<td></td>
<td>X</td>
<td></td>
<td>This applies to attributes that can be specified for relationship-type.</td>
</tr>
<tr>
<td>SEQATTR</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQOPT</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USERn</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 33 (Part 2 of 2). Summary of Definition Operands for an Installation-Defined Relationship-Type

Changing Attributes Specified for Instances of the Relationship-Type

The changes that may be made for attribute data are:

- Adding one or more new attributes.
- Deleting one or more existing attributes.
- Changing the data type (DATATYPE), minimum (DATAMIN), or maximum (DATAMAX) lengths for the values of one or more attributes.
- Changing the validation specifications for the values of one or more attributes.
- Increasing or decreasing the number of instances of a repeating attribute.
- Reordering the sequencing of attributes. If the relationship-type is sequenced, the sequence attribute must remain in the first attribute for the relationship-type.

In addition, you may not change the KEYWORD for an attribute.

Procedure for Reinstalling a Category or Relationship-Type

The reinstall procedure for an installation-defined category or relationship-type consists of two phases:

- Creating a replacement definition with the desired characteristics
- Using the INSTALL command to make the revised definition operational
Creating a Replacement Definition

If you want to replace a relationship-type that is included in the definition of an installed structure-type, see "Procedure for Changing an Installed Structure-Type" on page 200.

You create a replacement definition by defining a CATEGORY or RELTYPE subject in a test or production status. The user name of the new subject must be identical to the primary name of the definition being replaced. One way to start your replacement definition is by copying the definition being replaced to a test or production status. A processing report may be requested to describe the results of the COPY command.

You change definition operands by specifying the desired values for the new subject. The ways in which the operands for the new subject can differ from the ones for the definition being replaced are discussed in "Rules and Restrictions" on page 142. Note that, if you do not specify the value of an operand a default value, if there is one, a default will be assigned during reinstallation. No values from the definition being replaced will be automatically retained when the new subject is installed.

You change the attributes that may be specified for subjects in a category or instances of a relationship-type by defining the needed HAS-ATTRTYPE relationships to the new subject. Each attribute you want in the replacement definition must have a corresponding HAS-ATTRTYPE relationship to the new subject. If you have an attribute in the existing definition that you also want in the replacement definition, you must use the same keyword on the HAS-ATTRTYPE relationship for the attribute in the new definition. That is, you cannot change the keyword for an attribute via the reinstall process. An attribute can be added to the new definition by defining a HAS-ATTRTYPE relationship with a keyword different from any in the existing definition. An attribute is dropped by omitting the appropriate HAS-ATTRTYPE relationship from the replacement definition.

The reinstall process compares the attribute keywords defined in the existing definition with those in the replacement definition to determine which attributes are to be added, deleted, or retained. Changes to each retained attribute are determined by comparing its associated HAS-ATTRTYPE relationship in the existing definition with the one in the replacement definition.

You can change some aspects of an attribute (REPEAT and SEQ) by specifying the needed changes as relationship data in its corresponding HAS-ATTRTYPE relationship. You can change the format requirements (DATATYPE, DATAMIN, and DATAMAX) or the validation specifications (VALIDATE, VALRNGH, VALRNGL, VALRTN, and VALLIST) by associating the attribute with a different ATTRTYPE subject that has the desired specifications. Once an ATTRTYPE subject is installed, you cannot change it.

The ways in which the specifications for an attribute in the new definition can differ from those in the existing definition are discussed in "Changing Formatting Requirements for an Attribute" on page 144.
Installing the Replacement Definition

You use the INSTALL command with REPLACE=YES to make the replacement definition operational. During installation, the following processing will occur:

- The definition will be checked to make sure it is complete and correct. Defaults will be assigned, if necessary. See “Validations Performed by the INSTALL Command” on page 115 for a description of the validation performed.

- The replacement definition will be compared with the existing definition to make sure all changes are valid.

- If no errors are found in the above checks and if UPDATE=YES, the replacement definition will overlay the existing definition in installed status, and the data described by the existing definition (such as the subjects in the category or the instances of the relationship-type) will be converted to comply with the replacement definition. See “Data Conversion” on page 146 for more details on how data is converted.

An Installation Processing report may be requested to describe the results of any of the above steps. A description of the report sections is given in “Installation Processing Report for a Reinstalled Category Definition” on page 155 and “Installation Processing Report for a Reinstalled Relationship-Type Definition” on page 162.

Rules and Restrictions

This section describes the rules and restrictions for modifying CATEGORY and RELTYPE definitions via the reinstall process.

Changing a Category’s Alias Names

The alias subject names for the new definition do not have to correspond on a one-to-one basis with the alias names that were established for the existing definition. You may add or omit one or more of the alias names.

If an alias name for an installed category has been omitted from the new category definition, existing installed relationship-type definitions that reference the alias name as either the left- or right-hand category will be invalid. Therefore, these references will be automatically updated to contain the primary category name when the category definition is reinstalled. The “Installation Summary” section of the Installation Processing report will identify RELTYPE subjects that have been updated.
Changing Name Validation Operands for a Category

The operands that may be changed for name validation are:

- **NAMETYPE**—This may be changed to any valid value, such as 'A' for all alphabetic; 'N' for alphameric with the first character alphabetic or, for DBCS users, a uniform string; 'C' for character or, for DBCS users, a uniform string; or 'Z' for all numeric.

- **MINNAME**—This may be changed to permit or require either shorter or longer user names.

- **MAXNAME**—This may be changed to permit or require either shorter or longer user names.

- **VALRTN**—A validation routine may be added, changed, or deleted.

After the category has been reinstalled, validation of user names will be applied only for user names specified in commands which can result in the addition of new subjects to the Dictionary such as ADD, ADD_RELATIONSHIP, COPY, and CHANGE_NAME commands. Therefore, users will still be able to obtain reports for those subject names that do not meet the revised name specifications. Users must change the subject names to conform to the revised name specifications before updating the subject’s data or relationships.

Changing a Relationship-Type’s Alias Names

The alias subject names for the new definition do not have to correspond on a one-to-one basis with the alias names that were established for the existing definition. You may add or omit one or more of the alias names.

Changing Categories Participating in a Relationship-Type

You can change the LCATNAME or RCATNAME operand to specify a different alias name or the primary name of the same category. You cannot change the category. For example, if the installed definition had an LCATNAME category of “SEGMENT”, you may specify “SEG” in the replacement definition but not a different category such as “ELEMENT”.

Changing the Directed Attribute for a Relationship-Type

You can change a definition from DIRECTED=Y to DIRECTED=N. You cannot change the value of the DIRECTED attribute if the existing value is 'N'. If the DIRECTED attribute has been changed in the new definition, existing installed structure-type definitions will be changed to reflect the new DIRECTED value. However, the SUPERCAT, FOLLOWUP, and FOLLOWDN flags associated with the RELTYPE in existing installed structure-types will not be modified. This means that, with the exception of status rule enforcement, processing of structures involving the reinstalled relationship-type will not change. To achieve the complete change desired, you may have to replace the affected structure-types.
Changing Formatting Requirements for an Attribute

When you are making modifications to an installed category or relationship-type, a frequent request might be to change the formatting requirements for one or more of the attributes by respecifying the type of data or its syntax. For example, a maximum length of an attribute needs to be increased, or a numeric attribute needs to allow alphabetic characters to be present. DATAMIN or DATAMAX can be changed to be either larger or smaller. Figure 34 shows all permissible changes of DATATYPE.

<table>
<thead>
<tr>
<th>Change</th>
<th>To Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Z</td>
</tr>
<tr>
<td>From</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>—</td>
</tr>
<tr>
<td>A</td>
<td>No</td>
</tr>
<tr>
<td>C</td>
<td>Yes*</td>
</tr>
<tr>
<td>N</td>
<td>No</td>
</tr>
<tr>
<td>L</td>
<td>No</td>
</tr>
<tr>
<td>Q</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 34. Allowable Data Type Conversions

Notes to Figure 34:

- The hyphen means not applicable.
- The asterisks (*) for some of the values indicate that these values may be "invalidated." For example, a change from DATATYPE=C to DATATYPE=A would "invalidate" any existing values that did not consist of only alphabetic characters.
- For DBCS users, data types 'C', 'L', and 'Q' allow mixed strings. Data type 'N' allows uniform strings.

After the category or relationship-type has been reinstalled, validation of attributes will be applied only to attributes specified in commands that can result in the addition of attributes such as ADD, ADD_RELATIONSHIP, CHANGE_IN, and CHANGE_RELATIONSHIP_DATA commands. Therefore, users will still be able to obtain reports for those subject names that do not meet the revised specifications. Users must change the attribute names to conform to the revised name specifications before updating the attributes.

Changing Validation Specifications for an Attribute

The changes you can make for validation specifications for an attribute are:

- The type of validation (VALIDATE) to be applied to the values of an attribute may be changed, for example from VALIDATE=LIST to VALIDATE=RANGE.
• For “list” (VALIDATE=LIST) validation, the list of valid values (VALLIST(n)) may be changed—by either adding or deleting values from the list.

• For “range” (VALIDATE=RNG) validation, the minimum (VALRNGL) and/or maximum (VALRNGH) values defining the range may be changed; and in either direction (either to higher or lower values).

• For “routine” (VALIDATE=RTN) validation, the name of the validation routine (VALRTN) may be changed.

• The validation specification for an attribute may be deleted if such validation is no longer desired.

As with formatting requirements, validation of attributes will only be applied to commands that can result in the addition of attributes.

The following changes in validation specifications for an attribute may “invalidate” some of the existing values:

• A validation specification has been added for an attribute. For example, a value has been assigned to the VALIDATE operand.

• The validation specification has been changed. For example, the value for the VALIDATE attribute has been changed.

• A valid value for an attribute has been deleted from its corresponding list of valid values. For example, an instance of the VALLIST has been deleted.

• The valid range for an attribute value has been changed. For example, the VALRNGL value for an attribute has been changed to a higher value or its VALRNGH value has been changed to a lower value.

• The validation routine for an attribute has been changed. For example, the value of the VALRTN attribute associated with the corresponding ATTRTYPE subject has been changed or the routine referenced by the attribute has been modified.

The Reinstallation Process

You use the INSTALL command to make the new definition for a category or relationship-type operational. See “The INSTALL Command” on page 112 for the INSTALL command. You will always get validation checking, as described in “Validations Performed by the INSTALL Command” on page 115. These changes will appear in the “Validation Processing” section of the Installation Processing report. You get this section by specifying PROCRPT=VALIDATE with the INSTALL command.

Any category changes such as new alias names, deleted alias names, new attributes, deleted attributes, any attribute changes, and reordering of attributes will also be checked. These changes will appear in the “Category Alterations” or “RELTYPE
Alterations" section of the Installation Processing report. You get this section by specifying PROCRPT=ALTER with the INSTALL command.

If you specify PROCRPT=CHECK, all attribute data described by the definition will be checked against the replacement definition. If a category is being reinstalled, subject names will also be checked against the replacement definition. Attributes and subject names invalidated by the replacement definition will be listed in the "Conversion Processing" section of the Installation Processing report.

If you do not request the report, no validation of attributes or subject names will be performed. The data, however, will be converted to the replacement definition if UPDATE=Y. Details on conversion are described in "Data Conversion."

If you specify PROCRPT=TRUNC, all attributes which will be truncated on conversion to the new definition will be listed in the "Conversion Processing" section of the Installation Processing report. Even if you do not request the report, truncation will be performed when UPDATE=Y. Details on truncation are given in "DATAMAX Operand" on page 147.

Performance Considerations

Because the REPLACE option may result in the reformattting of large amounts of data, you may want to restrict its use to a batch-only environment.

If you specify PROCRPT=CHECK, performance may be significantly impacted if many of the attributes described by the replacement definition require validation by installation-written routines. Therefore, you may want to submit a preliminary INSTALL command (UPDATE=N) requesting the "CATEGORY ALTERATIONS" or "RETYPE Alterations" section of the processing report (PROCRPT=ALTER). If the resulting report indicates that some data may be potentially invalidated, you can submit a subsequent INSTALL command with PROCRPT=CHECK to identify specific values that will be invalidated.

Data Conversion

If UPDATE=Y was specified with the INSTALL command, data described by the replacement definition will be converted to that definition. During conversion, attribute data may be reformatted or truncated to satisfy the new requirements.

You will notice that subject names in the Dictionary will not be reformatted or truncated, although the validation specifications in the replacement definition for a category may have been changed. You must use the CHANGE__NAME command to correct the subject names after the reinstall process is completed. If you specify PROCRPT=CHECK with the INSTALL command, invalidated names will be listed in the "Conversion Processing" section of the Installation Processing report.
DATATYPE Operand

Some things to note about data type conversions are:

- For data types changed to DATATYPE=Q, the existing values for the attribute will be reformatted during installation processing to enclose the current value in single quotation marks (' ') (except for DATATYPE=L).

  For DATATYPE=A, C, or N, the quotation marks will be inserted before the first character, and immediately after the last nonblank character in the value.

  For DATATYPE=Z, any leading zeros on the existing value will be stripped, and the remaining digits enclosed in quotation marks. The DATAMAX specification for the attribute must be increased by at least two bytes for the enclosing quotation marks. Otherwise, the existing values may be truncated.

  For DATATYPE=L, the existing values for the attribute will be reformatted with the enclosing parentheses replaced by the single quotation marks.

- For data types changed to DATATYPE=L, the existing values for the attribute will be remapped during installation processing to enclose the current value in parentheses.

  For DATATYPE=A, C, or N, the parentheses will be inserted before the first character and immediately after the last nonblank character in the value.

  For DATATYPE=Z, any leading zeros on the existing value will be stripped, and the remaining digits enclosed in parentheses. The DATAMAX specification for the attribute must be increased by two bytes for the enclosing parentheses. Otherwise, the existing values may be truncated.

- During conversion of an attribute from DATATYPE=Z to DATATYPE=C, any leading zeros on the existing values for the attribute will be stripped. For example, a numeric value '00354' would be converted to '354'.

- An attribute of DATATYPE=Q may not be changed to any other DATATYPE.

DATAMAX Operand

The DATAMAX operand determines the maximum length in bytes for the values of an attribute and includes the enclosing quotation marks or parentheses. This operand also determines the amount of space that is allocated in the Dictionary data base for storing values of an attribute.
When the DATAMAX operand is changed to a longer value, existing values for an attribute will be "padded" during installation processing and will conform to the following rules:

- DATATYPE=Z attributes will be padded on the left with leading zeros.
- The values for attributes of any other DATATYPE will be padded on the right with blanks.
- Any conversion of the attribute value will be performed before padding.

When the DATAMAX operand is changed to a shorter value, existing values for the attribute will be truncated during installation processing and will conform to the following rules:

- DATATYPE=Z attributes will be truncated by first stripping leading zeros and then trailing digits from the value.
- DATATYPE=A, C, or N will be truncated by stripping trailing characters from the value.
- DATATYPE=L will be truncated by:
  - First stripping any trailing blanks
  - Then, stripping trailing characters preceding the closing parenthesis
- DATATYPE=Q will be truncated by:
  - First stripping any trailing blanks
  - Then stripping trailing characters preceding the closing quotation marks. If the final character to be stripped within the quotation marks is the second of a pair of single quotation marks, both quotation marks will be deleted.

Any changes to the DATAMAX operand will be reflected in the "Conversion Processing" section of the Installation Processing report.

**REPEAT Operand**

The REPEAT operand determines the number of times you may specify an attribute value for a single subject or relationship. If the REPEAT value is greater than one, the attribute is called a "repeating attribute." The keyword for each occurrence of the attribute has the format "keyword(n)", where "n" cannot exceed the REPEAT value.

When the REPEAT operand is changed to a smaller number, occurrences of the attribute with instance numbers greater than the new REPEAT value will not be converted to the replacement definition. After reinstallation, that data will be deleted from the Dictionary. For example, when the REPEAT operand is changed to one, only the attribute with instance number "1" will be converted. Attributes with instance numbers of two or greater will be deleted. When the REPEAT
operand is changed from one to greater than one, the existing value will be assigned to the attribute with instance number one "keyword(1)".

Data Not Converted to Replacement Definition

An attribute value will not be converted to the replacement definition if:

- The attribute keyword is not included in the replacement definition.
- The attribute keyword has an instance number greater than the REPEAT value for the same attribute keyword in the replacement definition.

Values that are not converted will be deleted from the Dictionary after reinstallation.

Summary of Reinstallation Steps

The following procedure summarizes your activities:

1. Make a copy of the existing definition:
   - The user name must be identical to the existing primary name, that is, the name of the category or relationship-type stays the same.
   - The status code can be any test status or status 'P'.
   
   Use the COPY command with the AND RELATE option so that the subject, subject data, and aliases are also copied.

2. Make modifications, within the constraints of the reinstall process:
   - Provide new aliases or delete them, as desired—names must not conflict with any names of any other installed category or RELTYPE.
   - Alter any of the subject name validation specifications.
   - Provide new attribute-type definitions or use an attribute-type definition with the right attributes. Add any new relationships such as HAS-ATTRTYPE to the new definition.
   - Alter the sequence of any of the "old" attributes, as required.
   - Make attribute deletions by deleting the HAS-ATTRTYPE relationships.
   - Increase or decrease the number of times a repeating attribute can occur.
3. Make use of the INSTALL command with the REPLACE option:

- Specify UPDATE=NO. If performance is a consideration, you should not specify the CHECK option. You can check the "Category Alterations" section of the report after the INSTALL command has finished processing for any potential data problems. If there are potential data problems, you can then specify the CHECK option. This should be performed as a batch job, because there may be a large amount of data to check.

- Analyze the effect on the existing data.

- After final checking, use both UPDATE=YES, REPLACE=YES options to complete the reinstall process. At this time, the status code of the subject to be installed must be 'P'.

- Distribute GUIDE reports and make known that the revised definition is fully operational.

While you are doing the procedure, the existing category definition is still operational, as are any relationship-type definitions used by the category definition in installed status.
Example of a Reinstalled Category Definition

The initial installation for this category is shown in Chapter 7, “Installing Categories and Relationship-Types” on page 111.

You may find it helpful to make your own chart of the changes you plan to make. You can then compare what you expect with the details given in the alterations section of the report. For example, here is a summary of the changes made to the PGMR category:

<table>
<thead>
<tr>
<th>Old Operand/Keyword</th>
<th>New Operand/Keyword</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category: PGMR</td>
<td>Category: same</td>
<td>Add SE as an alias</td>
</tr>
<tr>
<td>Alias: PROGMR</td>
<td>Aliases: add SE</td>
<td></td>
</tr>
<tr>
<td>User names: minimum length 1 character</td>
<td>User names: minimum length 3 character</td>
<td>Make minimum length 3 characters</td>
</tr>
<tr>
<td>Validation: no routines</td>
<td>Validation: add PGMRCHK</td>
<td>Names use routine &quot;PGMRCHK&quot;</td>
</tr>
<tr>
<td>TITTLE</td>
<td>—</td>
<td>Moved</td>
</tr>
<tr>
<td>OFFICE</td>
<td>—</td>
<td>Delete</td>
</tr>
<tr>
<td></td>
<td>SERIAL</td>
<td>New (need to add attribute-type)</td>
</tr>
<tr>
<td></td>
<td>DIVISION</td>
<td>New (need to add attribute-type)</td>
</tr>
<tr>
<td>LOCATION</td>
<td>Same</td>
<td>Fixed length 5, numeric (need to add attribute-type)</td>
</tr>
<tr>
<td></td>
<td>BLDG</td>
<td>New, attribute-type already installed</td>
</tr>
<tr>
<td></td>
<td>ROOM</td>
<td>New, attribute-type already installed</td>
</tr>
<tr>
<td>PHONE</td>
<td>Same</td>
<td>Allow 7 digits (need to add attribute-type)</td>
</tr>
<tr>
<td>Moved</td>
<td>TITTLE</td>
<td>T TITLE keyword, no change</td>
</tr>
<tr>
<td>LANGSKIL</td>
<td>Same</td>
<td>Allow length 7, and 7 values. Add APL and FORTRAN to list (need to add attribute-type, occurrence 001)</td>
</tr>
</tbody>
</table>
The procedure to follow would be:

1. Decide how the modified category will appear to the user.

2. Create a new CATEGORY subject with appropriate HAS-ATTRTYPE relationships to define the modified category.

3. Verify that the category definition is as wanted.

4. Install the definition with the REPLACE option.

You can then go through the steps to define and install a modified category definition.


You should make decisions on how the category will appear to the user. Some questions you might want to ask are:

- What aliases will the category subject have?

- What validation will be necessary for the user names of subjects in the category?

- What attributes should subjects in the category have? What validation is wanted for each of these attributes? Which of these attributes are already in the current definition? Which attributes have ATTRTYPE subjects that already specify the validation rules desired?

2. Creating the New Definition.

You can create a new definition by performing the following steps:

a. First you would want to create a production category. You can use the COPY command as follows:

   COPY CATEGORY (*,PGMR,0) TO CATEGORY (P,PGMR,0) +
   AND RELATE UPDATE=YES ;

   The new category will be copied into "production" status because it must be in this status to be reinstalled. The COPY function will use the default structure-type to determine which relationships are to be copied. The COPY function will copy alias names, Description and User Data text, and subject name validation operands. The RELATE option will copy HAS-ATTRTYPE relationships.

b. Next, you can add or delete alias names. The following command will make the production subject conform to the new definition:

   ADD_RELATIONSHIP CATEGORY (P,PGMR,0) +
   HAS_ALIAS CATEGORY (P,SE,0) ;

   The subject will now have two aliases: PROGMR and SE. The first was created by the COPY command. The second was created by the ADD_RELATIONSHIP command above.
c. You can then update validation rules for subject names in the category through the following commands:

```
CHANGE_IN CATEGORY (P,,PGMR,0) FROM NAMETYPE=N + 
   NAMEMIN=1 TO NAMETYPE=A NAMEMIN=3 ;
ADD CATEGORY (P,,PGMR,0) VALRTN=PGMCHK ;
```

d. You can then create new ATTRTYPE subjects through the following commands:

```
ADD ATTRTYPE (P,,DIVISION,0) DATATYPE=A DATAMIN=3 +
   DATAMAX=3 ;
ADD ATTRTYPE (P,,SERIAL,0) DATATYPE=Z DATAMIN=8 +
   DATAMAX=8 ;
```

You can then create ATTRTYPE subjects for modified validation rules for LOCATION, PHONE, and LANGSKIL:

```
ADD ATTRTYPE (P,,LOCATION_CODE,0) DATATYPE=Z +
   DATAMIN=5 DATAMAX=5 ;
ADD ATTRTYPE (P,,PHONE_NUMBER,0) DATATYPE=Z +
   DATAMIN=4 DATAMAX=7 ;
ADD ATTRTYPE (P,,PROG_LANG_SKILL,1) DATATYPE=C +
   DATAMIN=3 DATAMAX=7 VALLIST(1)=COBOL +
   VALLIST(2)=PL/I VALLIST(3)=RPG VALLIST(4)=ASM +
   VALLIST(5)=DL/I VALLIST(6)=APL VALLIST(7)=FORTRAN ;
```

e. You can then add or delete relationships to the new category subject. To delete a relationship, you would specify:

```
DELETE_RELATIONSHIP CATEGORY (P,,PGMR,0) HAS ATTRTYPE+
   (*,,OFFICE_ID,0) SEQ=10 ;
```

Relationships that no longer specify desired validation rules can be deleted with:

```
DELETE_RELATIONSHIP CATEGORY (P,,PGMR,0) HAS +
   ATTRTYPE (*,,INSTALLATION_LOCATION,0) SEQ=15 ;
DELETE_RELATIONSHIP CATEGORY (P,,PGMR,0) HAS +
   ATTRTYPE (*,,INTERNAL_PHONE_EXTENSION,0) SEQ=20 ;
DELETE_RELATIONSHIP CATEGORY (P,,PGMR,0) HAS +
   ATTRTYPE (*,,PROG_LANG_SKILL,0) SEQ=25 ;
```

The relationships for LOCATION, PHONE, and LANGSKIL will be replaced by new relationships to ATTRTYPE subjects that specify the new validation rules. The commands would be:

```
ADD_RELATIONSHIP CATEGORY (P,,PGMR,0) HAS ATTRTYPE +
   (P,,LOCATION_CODE,0) SEQ=15 KEYWORD=LOCATION ;
ADD_RELATIONSHIP CATEGORY (P,,PGMR,0) HAS ATTRTYPE +
   (P,,PHONE_NUMBER,0) SEQ=30 KEYWORD=PHONE ;
ADD_RELATIONSHIP CATEGORY (P,,PGMR,0) HAS ATTRTYPE +
   (P,,PROG_LANG_SKILL,1) SEQ=40 KEYWORD=LANGSKIL +
   REPEAT=7 ;
```
New attributes can be added to the category definition with the following commands:

```
ADD_RELATIONSHIP CATEGORY (P,,PGMR,0) HAS ATTRTYPE +
  (P,,SERIAL,0) SEQ=5 KEYWORD=SERIAL ;
ADD_RELATIONSHIP CATEGORY (P,,PGMR,0) HAS ATTRTYPE +
  (P,,DIVISION,0) SEQ=10 KEYWORD=DIVISION ;
ADD_RELATIONSHIP CATEGORY (P,,PGMR,0) HAS ATTRTYPE +
  (*.BUILDING,0) SEQ=20 KEYWORD=BUILDG ;
ADD_RELATIONSHIP CATEGORY (P,,PGMR,0) HAS ATTRTYPE +
  (*.ROOM_NUMBER,0) SEQ=25 KEYWORD=ROOM ;
```

You can also change the sequence attribute for an existing relationship with the following command:

```
CHANGE_RELATIONSHIP_DATA CATEGORY (P,,PGMR,0) +
  HAS ATTRTYPE (*.JOB_TITLE,0) FROM SEQ=5 TO SEQ=35 ;
```

f. Update text and other specifications. You may want to modify User Data or Description text to reflect changes to the category definition. The command to change the DATE keyword is:

```
CHANGE IN CATEGORY (P,,PGMR,0) FROM DATE=031279 +
  TO DATE=050385 ;
```

3. Verify the new definition. You can study the category definition through the GUIDE, structure, or subject-specific report. The GUIDE report will show how the category definition will appear after it has been installed. You can check this report to determine which subject names in a category definition, if any, will be invalidated by the new category definition. You can also determine if you might have potential data problems that do not conform to changed or new attributes.

The structure report will show the HAS-ATTRTYPE relationships that have been created for the CATEGORY subject (assuming that the CATEGORY-HAS-ATTRTYPE relationship-type is in the default structure-type).

The subject-specific report will contain all the operands that describe validation rules for subject names in the category, and also the relationships to ATTRTYPE subjects (which define the attributes that may be specified for a subject in the category). Individual reports of the related ATTRTYPE subjects must be obtained to determine the validation rules for each attribute.

4. Reinstall the New Definition.

The category definition will be replaced by executing the following command:

```
INSTALL CATEGORY (P,,PGMR,0) UPDATE=YES REPLACE=YES +
  PROCRT=ALL RPT=GUIDE ;
```
Installation Processing Report for a Reinstalled Category Definition

Figure 35 shows a sample Installation Processing report that will be produced when the PGMR category completed the reinstall process.

INSTALLATION PROCESSING REPORT  11/11/85  13:42:03
PAGE: 0001

COMMAND OPTIONS:

INSTALLATION OF CATEGORY - PGMR
UPDATE: YES          DEST: L
REPLACE: YES          REPORT: GUIDE
PROCRPT: ALL

VALIDATION PROCESSING:

ALIAS:   PROGMR
      SE

SUBJECT CODE:  00057
SUBJECT NAMES:  TYPE: A  MIN-LENGTH: 03  MAX-LENGTH: 31
NAME VALIDATION:  PGMRCHK
DATE:  050385  EXPTEXT:  5
DBD5543 I MINNAME VALUE DEFAULTED TO 1

ATTRIBUTES:

ST  USERNAME  OCC  SEQ  KEYWORD  REP
--  -----------  ---  -----  -------  ----
P  SERIAL  000  005  SERIAL
DBD5573 I NO VALIDATION SPECIFIED FOR ATTRIBUTE-TYPE
P  DIVISION  000  010  DIVISION
DBD5573 I NO VALIDATION SPECIFIED FOR ATTRIBUTE-TYPE
P  LOCATION_CODE  000  015  LOCATION
DBD5573 I NO VALIDATION SPECIFIED FOR ATTRIBUTE-TYPE
  * BUILDING  000  020  BLDG
DBD5503 I ATTRTYPE ALREADY INSTALLED
  * ROOM_NUMBER  000  025  ROOM
DBD5503 I ATTRTYPE ALREADY INSTALLED
P  PHONE_NUMBER  000  030  PHONE
DBD5573 I NO VALIDATION SPECIFIED FOR ATTRIBUTE-TYPE
  * JOB_TITLE  000  035  TITLE
DBD5503 I ATTRTYPE ALREADY INSTALLED
P  PROG_LANG_SKILL  001  040  LANGSKIL  007

Figure 35 (Part 1 of 3).  Sample Installation Processing Report for a Reinstalled Category
**CATEGORY ALTERATIONS:**

**MODIFICATIONS TO THE CATEGORY SUBJECT:**

<table>
<thead>
<tr>
<th>NEW ALIAS NAMES</th>
<th>DELETED ALIAS NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KEYWORD</strong></td>
<td><strong>CHANGE</strong></td>
</tr>
<tr>
<td><strong>NAME</strong></td>
<td><strong>OLD:</strong> N</td>
</tr>
<tr>
<td><strong>NEW:</strong></td>
<td><strong>A</strong></td>
</tr>
<tr>
<td><strong>MINNAME</strong></td>
<td><strong>OLD:</strong> 1</td>
</tr>
<tr>
<td><strong>NEW:</strong></td>
<td><strong>3</strong></td>
</tr>
<tr>
<td><strong>VALRRTN</strong></td>
<td><strong>OLD:</strong></td>
</tr>
<tr>
<td><strong>NEW:</strong></td>
<td><strong>PGMRCHK</strong></td>
</tr>
<tr>
<td><strong>DATE</strong></td>
<td><strong>OLD:</strong> 031279</td>
</tr>
<tr>
<td><strong>NEW:</strong></td>
<td><strong>050381</strong></td>
</tr>
</tbody>
</table>

**MODIFICATIONS TO THE ATTRIBUTES:**

<table>
<thead>
<tr>
<th>NEW ATTRIBUTES</th>
<th>DELETED ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SERIAL</strong></td>
<td><strong>OFFICE</strong></td>
</tr>
<tr>
<td><strong>DIVISION</strong></td>
<td></td>
</tr>
<tr>
<td><strong>BLDG</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ROOM</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LOCATION</strong></td>
<td><strong>DATATYPE</strong></td>
</tr>
<tr>
<td></td>
<td><strong>NEW:</strong> Z</td>
</tr>
<tr>
<td><strong>DATAMAX</strong></td>
<td><strong>OLD:</strong> 24</td>
</tr>
<tr>
<td></td>
<td><strong>NEW:</strong> 5</td>
</tr>
<tr>
<td><strong>PHONE</strong></td>
<td><strong>DATAMAX</strong></td>
</tr>
<tr>
<td></td>
<td><strong>NEW:</strong> 7</td>
</tr>
<tr>
<td><strong>LANGSKIL</strong></td>
<td><strong>REPEAT</strong></td>
</tr>
<tr>
<td></td>
<td><strong>NEW:</strong> 7</td>
</tr>
<tr>
<td><strong>VALLIST</strong></td>
<td><strong>OLD:</strong> COBOL,PL/I,RPG,ASM,DL/I</td>
</tr>
<tr>
<td></td>
<td><strong>NEW:</strong> COBOL,PL/I,RPG,ASM,DL/I</td>
</tr>
<tr>
<td></td>
<td><strong>APL</strong>,FORTRAN</td>
</tr>
</tbody>
</table>

**ORDERING OF ATTRIBUTES HAS BEEN CHANGED**

**Figure 35 (Part 2 of 3). Sample Installation Processing Report for a Reinstalled Category**
CONVERSION PROCESSING:

<table>
<thead>
<tr>
<th>SUBJECT NAME</th>
<th>OLD VALUE</th>
<th>TRUNCATED INVALIDATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>P DOUGLAS_JOHNSON</td>
<td>00001SFO</td>
<td>T</td>
</tr>
<tr>
<td>LOCATION:</td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td>P SHARON_SMITH</td>
<td>00002SJO</td>
<td>T</td>
</tr>
<tr>
<td>LOCATION:</td>
<td>0002</td>
<td></td>
</tr>
<tr>
<td>P CHARLES_BROWN</td>
<td>0002SCL</td>
<td>T</td>
</tr>
<tr>
<td>LOCATION:</td>
<td>0002</td>
<td></td>
</tr>
</tbody>
</table>

NO SUBJECT NAMES HAVE BEEN INVALIDATED

NO DATA HAS BEEN INVALIDATED

INSTALLATION SUMMARY:

DBD5504 I INSTALLATION SUCCESSFUL FOR ATTRTYPE SERIAL
DBD5504 I INSTALLATION SUCCESSFUL FOR ATTRTYPE DIVISION
DBD5504 I INSTALLATION SUCCESSFUL FOR ATTRTYPE LOCATION_CODE
DBD5504 I INSTALLATION SUCCESSFUL FOR ATTRTYPE PHONE_NUMBER
DBD5504 I INSTALLATION SUCCESSFUL FOR ATTRTYPE PROG_LANG_SKILL
DBD5701 I REPLACEMENT OF CATEGORY P PGMR 0 COMPLETED

***** END OF INSTALLATION PROCESSING REPORT *****

Figure 35 (Part 3 of 3). Sample Installation Processing Report for a Reinstalled Category
GUIDE Report for a Reinstalled Category Definition

You can compare GUIDE reports for the PGMR category. Figure 36 is the modified definition and Figure 27 on page 124 is the original definition. You will not see any direct indications on the GUIDE report for the reinstalled category about those attributes that have changed. For this reason, when you inform the end users that there has been a change in the definition, you could include a brief summary of those changes, or you could annotate the GUIDE report itself, showing new attributes and altered data requirements. Note that the relationships that the subject category can take part in are not changed and still appear in the second part of the GUIDE report.

* * * * * * * DB/DC DATA DICTIONARY REPORT 11/11/84 16:03:51
CATEGORY: PGMR

CATEGORIES: PROGMR
SE

DESCRIPTION:

SUBJECTS IN THIS CATEGORY REPRESENT PROGRAMMERS

SUBJECT NAME RULES: DATE: 050582
NAME TYPE: A
MIN LENGTH: 03
MAX LENGTH: 31
VAL ROUTINE: PGMCHK

ATTRIBUTES:

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>REPEAT FACTOR</th>
<th>LENGTH MIN</th>
<th>LENGTH MAX</th>
<th>DATA TYPE</th>
<th>VALIDATION TYPE</th>
<th>VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERIAL</td>
<td>00</td>
<td>8</td>
<td>8</td>
<td>Z</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>DIVISION</td>
<td>00</td>
<td>3</td>
<td>3</td>
<td>A</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>LOCATION</td>
<td>00</td>
<td>5</td>
<td>5</td>
<td>Z</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>BLDG</td>
<td>00</td>
<td>1</td>
<td>3</td>
<td>C</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>ROOM</td>
<td>00</td>
<td>3</td>
<td>3</td>
<td>C</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>PHONE</td>
<td>00</td>
<td>4</td>
<td>7</td>
<td>Z</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>TITLE</td>
<td>00</td>
<td>1</td>
<td>24</td>
<td>C</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>LANGSKIL</td>
<td>07</td>
<td>3</td>
<td>7</td>
<td>C</td>
<td>LIST</td>
<td>COBOL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>,PL/I</td>
<td>,RPG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>,ASM</td>
<td>,DL/I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>,APL</td>
<td>,FORTRAN</td>
</tr>
</tbody>
</table>

Figure 36 (Part 1 of 2). Example of a Category GUIDE Report after a Reinstall Process
Example of a Reinstalled Relationship-Type Definition

Your preparation for a reinstall process for an installation-defined relationship-type is similar to that for a category. You perform a trial install using the UPDATE=NO and REPLACE=YES options on the INSTALL command. You need to check the changes to the definition and the attributes used for relationship data. Then examine the effects reported in the “Conversion Processing” section of the Installation Processing report.

For this example, a relationship-type named “PGMR/UPDATED/PROGRAM” has the following modifications:

<table>
<thead>
<tr>
<th>Old Operand/Keyword</th>
<th>New Operand/Keyword</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias: RELTYPE</td>
<td>Alias: add RELTYP040</td>
<td>Add RELTYP040</td>
</tr>
<tr>
<td>DIRECTED</td>
<td></td>
<td>Make not directed</td>
</tr>
<tr>
<td>—</td>
<td>CHNGID</td>
<td>New attribute, put as last one (need to install)</td>
</tr>
</tbody>
</table>

The procedure to follow would be:

1. Decide how the modified relationship-type will appear to the user.
2. Create a new RELTYPE subject with appropriate HAS-ATTRTYPE relationships to define the modified relationship-type.
3. Verify that the relationship-type definition is as wanted.
4. Install the definition with the REPLACE option.
You can then go through the steps to define and install a modified relationship-type definition:


Some of the questions you might want to ask are:

- What aliases will the RELTYPE subject have?
- Should a "directed" relationship-type be modified so that status rules are no longer enforced for instances of this type?
- Should a different name (for the same category) be used to specify the left- or right-hand category in the relationship-type definition?
- What attributes should instances of the relationship-type have? What validation is desired for each of these attributes? Which of these attributes are already in the current definition? Which attributes have ATTRTYPE subjects which already specify the validation rules desired?

2. Create the New Definition.

To create a new definition, you would perform the following steps:

a. Create a production relationship-type.

You can specify the following command to copy an existing RELTYPE:

```
COPY RELTYPE (*,PGMR/UPDATED/PROGRAM,0) +
    TO RELTYPE (P,,PGMR/UPDATED/PROGRAM,0) AND RELATE ;
```

b. Add an alias name.

To add an alias, you can specify the following command:

```
ADD_RELATIONSHIP RELTYPE (P,,PGMR/UPDATED/PROGRAM,0) +
    HAS_ALIAS RELTYPE (P,,RELTYPE040,0) ;
```

c. Update operands in the relationship-type definition.

To update the operands, you can specify the following:

```
CHANGE_IN RELTYPE (P,,PGMR/UPDATED/PROGRAM,0) +
    FROM DIRECTED=Y TO DIRECTED=NO ;
```

d. Create ATTRTYPE subjects.

To create an ATTRTYPE subject, you can specify the following:

```
ADD ATTRTYPE (P,,CHANGE_IDENTIFICATION,0) DATATYPE=C +
    DATAMIN=7 DATAMAX=7 ;
```
e. Add or delete relationships to the new RELTYPE subject.

To add a new attribute to the relationship-type definition, you can specify the following:

```
ADD_RELATIONSHIP (P,,PGMR/UPDATED/PROGRAM,0) HAS +
ATTRTYPE (P,,CHANGE_IDENTIFICATION,0) +
SEQ=25 KEYWORD=CHNGID ;
```

3. Verify the New Definition.

You can get GUIDE, structure, and subject-specific reports to help examine the new definition.

The GUIDE report will show how the relationship-type definition will appear after it has been installed.

The structure report shows the HAS-ATTRTYPE relationships that have been created for the RELTYPE subject (assuming that the REL`YPE-HAS-ATTRTYPE relationship-type is in the default structure-type).

The subject-specific report contains all the operands associated with a relationship-type, and also the relationships to ATTRTYPE subjects (which define the attributes that may be specified for instances of this relationship-type). To determine the validation rules for each attribute, individual reports of the related ATTRTYPE subjects must be obtained.

4. Reinstall the New Definition.

The relationship-type definition will be replaced by the INSTALL command shown below:

```
INSTALL RELTYPE (P,,PGMR/UPDATED/PROGRAM,0) +
UPDATE=YES REPLACE=YES PROCRPT=ALL RPT=ALL;
```
Installation Processing Report for a Reinstalled Relationship-Type Definition

Figure 37 shows a sample Installation Processing report when a relationship-type is reinstalled.

COMMAND OPTIONS:
INSTALLATION OF RELTYPE  -  PGMR/UPDATED/PROGRAM
UPDATE: YES   DEST: L
REPLACE: YES   REPORT: ALL
PROCRPT: ALL
NOTE: "*" MEANS DEFAULT VALUE TAKEN

VALIDATION PROCESSING:

ALIAS: PROGRAM/UPDATED_BY/PGMR
       RELTYP040
LEFT-HAND CATEGORY:  PGMR
RIGHT-HAND CATEGORY:  PROGRAM
FORWARD NAME:  UPDATED
INVERSE NAME:  UPDATED_BY
SEQUENCED:  Y
SEQUENCE-ATTRIBUTE:  MODDATE
DIRECTED:  N
DATE:  102782   EXPTEXT:  5

ATTRIBUTES:
ST USERNAME OCC SEQ KEYWORD REP
-- ------------------------- --- ------ ----- ---
* DATE 000 002 MODDATE
DBD5503 I ATTRTYPE ALREADY INSTALLED
* FAILURE REASON 000 005 REASON
DBD5503 I ATTRTYPE ALREADY INSTALLED
* PROBLEM DESCRIPTION 000 010 PROBLEM
DBD5503 I ATTRTYPE ALREADY INSTALLED
* CORRECTION DESCRIPTION 000 015 CORRECTN
DBD5503 I ATTRTYPE ALREADY INSTALLED
* DATE 000 020 VERDATE
DBD5503 I ATTRTYPE ALREADY INSTALLED
P CHANGE IDENTIFICATION 000 025 CHNGID
DBD5573 I NO VALIDATION SPECIFIED FOR ATTRIBUTE-TYPE

Figure 37 (Part 1 of 2). Sample Installation Processing Report for a Reinstalled Relationship-Type

162  Administration and Customization Guide
**RELTYPE ALTERATIONS:**

<table>
<thead>
<tr>
<th>INSTALLATION</th>
<th>PROCESSING REPORT</th>
<th>01/22/84 11:06:55</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE: 0002</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MODIFICATIONS TO THE RELTYPE SUBJECT**

<table>
<thead>
<tr>
<th>NEW ALIAS NAMES</th>
<th>DELETED ALIAS NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----------------</td>
<td>----------------------</td>
</tr>
</tbody>
</table>

**RELTP040**

**KEYWORD**   **CHANGE**

**DIRECTED**  **OLD: Y**

**NEW: N**

**MODIFICATIONS TO THE ATTRIBUTES:**

<table>
<thead>
<tr>
<th>NEW ATTRIBUTES</th>
<th>DELETED ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CHNGID**

**CONVERSION PROCESSING:**

<table>
<thead>
<tr>
<th>SUBJECT A NAME</th>
<th>RELKW</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBJECT B NAME</td>
<td>*SEQ</td>
</tr>
</tbody>
</table>

**NO DATA HAS BEEN TRUNCATED**

**NO DATA HAS BEEN INVALIDATED**

**NO VALIDATION ROUTINES WERE MISSING**

**INSTALLATION SUMMARY:**

- **DBD5504 I INSTALLATION SUCCESSFUL FOR ATTRTYPE CHANGE_IDENTIFICATION**
- **DBD5500 I INSTALLATION OF RELTYPE P PGMR/UPDATED/PROGRAM 0 COMPLETED**
- **DBD5701 I REPLACEMENT OF RELTYPE P PGMR/UPDATED/PROGRAM 0 COMPLETED**

**** END OF INSTALLATION PROCESSING REPORT ****

---

**Figure 37 (Part 2 of 2). Sample Installation Processing Report for a Reinstalled Relationship-Type**
GUIDE Report for a Reinstalled Relationship-Type Definition

You can compare GUIDE reports for this relationship-type. Figure 38 is the modified definition and Figure 28 on page 127 is the original definition. You will not see any direct indications on the GUIDE report for the reinstalled relationship-type about those attributes that have changed. For this reason, when you inform the end users that there has been a change in the definition, you could include a brief summary of those changes, or you could annotate the GUIDE report itself, showing new attributes and altered data requirements.

* * * * * * *  DB/DC DATA DICTIONARY REPORT  01/22/84  11:07:05
   RELTYPE  GUIDE
   PAGE:   01

RELTYPE:  PGMR/UPDATED/PROGRAM
ALIASES:  PROGRAM/UPDATED_BY/PGMR
          RELTYF040

DESCRIPTION:

PROGRAMMER - UPDATED - PROGRAM RELATIONSHIP

RELATIONSHIP TYPE RULES:  DATE:  110579
   LEFT-HAND CATEGORY:  PGMR
   RIGHT-HAND CATEGORY:  PROGRAM
   FORWARD NAME:  UPDATED
   INVERSE NAME:  UPDATED_BY
   SEQUENCE OPTION:  YES
   SEQUENCE ATTRIBUTE:  MODDATE
   DIRECTED:  NO

ATTRIBUTES:

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>REPEAT</th>
<th>LENGTH</th>
<th>DATA</th>
<th>VALIDATION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FACTOR</td>
<td>MIN</td>
<td>MAX</td>
<td>TYPE</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>--------</td>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>MODDATE</td>
<td>00</td>
<td>6</td>
<td>6</td>
<td>Z</td>
</tr>
<tr>
<td>REASON</td>
<td>00</td>
<td>3</td>
<td>24</td>
<td>Q</td>
</tr>
<tr>
<td>PROBLEM</td>
<td>00</td>
<td>3</td>
<td>120</td>
<td>Q</td>
</tr>
<tr>
<td>CORRECTN</td>
<td>00</td>
<td>3</td>
<td>120</td>
<td>Q</td>
</tr>
<tr>
<td>VERDATE</td>
<td>00</td>
<td>6</td>
<td>6</td>
<td>Z</td>
</tr>
<tr>
<td>CHNGID</td>
<td>00</td>
<td>7</td>
<td>7</td>
<td>C</td>
</tr>
</tbody>
</table>

* * *  END-OF-REPORT  * * *

Figure 38.  Example of a Relationship-Type GUIDE Report after a Reinstallation Process
Chapter 9. Display Forms for Customized Entities

Installation-defined categories and relationship-types, and the attribute-types you define to use with them, can be completely specified with display forms. You establish the definition operands of ATTRTYPE, CATEGORY, and RELTYPE subjects using the EXTENSIBILITY SUBJECT form. To make any of these entities operational for the general user requires the use of the INSTALL command. After the definitions are installed, you can conveniently modify text data to complete the documentation of the entities.

The forms that help you manage relationships for these subjects are:

- Special relationships forms to relate ATTRTYPE subjects to CATEGORY or RELTYPE subjects
- A CATEGORY-RELTYPES form that lists all the installed relationship-types that currently connect an individual installed category to other categories

It is assumed that you are familiar with the general hierarchy of display forms and the online operation of the displays, as described in Chapter 3 of OS/VS DB/DC Data Dictionary Terminal User’s Guide and Command Reference. It is also assumed that you will refer to Chapter 4 of that publication for an explanation of the following forms that deal with installation-defined entities:

- EXTENSIBILITY SUBJECT form
- 'Selected' SUBJECT-RELATIONSHIPS form
- 'Unselected' SUBJECT-RELATIONSHIPS form
- RELATIONSHIP DATA form

The common subordinate forms are:

- EXPLANATION
- DESCRIPTION
- USER DATA
- ALIASES
- EXTENSIBILITY EXPLANATION
The hierarchy of display forms is summarized in Figure 39 on page 167 to help you work with customized entities both during development and after the definitions are installed.

**Extensibility Control Information Categories and the Header Form**

The standard Header display form can be updated to list extensibility categories by changing the DBDXHD module. However, a more flexible approach is to have users at your installation use the Sign-On Header display form instead of the standard Header form. When user access control is active, the Sign-On Header display form can be formatted to display a customized list of categories (including extensibility categories) for each user defined by security administration. For details of how to format the display form, see “Defining Dictionary Users with Display Forms” on page 39.
Figure 39. Hierarchy of Display Forms for Extensibility Subjects
Using Display Forms for CATEGORY Subjects

The EXTENSIBILITY SUBJECT Form for a CATEGORY Subject

If you enter the subject name for a CATEGORY subject on the Header form, and use as category name CATEGORY or CAT (or the category number '20'), the EXTENSIBILITY SUBJECT form is displayed showing the title "CATEGORY" and the definition of the specified category (if one exists). You can use this form to enter and edit the definition for a category, and to obtain access to the ALIASES, DESCRIPTION, and USER DATA forms to edit the alias names and Description and User Data text for the CATEGORY subject.

The form for editing a CATEGORY subject and its attributes is shown in Figure 40. Note that this form is the standard EXTENSIBILITY SUBJECT form. A description of entries and actions on the CATEGORY form is given in Chapter 4 of OS/VS DB/DC Data Dictionary Terminal User’s Guide and Command Reference. Modification of the definition of an installed category is limited to the DATE, EXPTEXT, and VALRTN attributes, and to the Description and User Data text.

---

Figure 40. EXTENSIBILITY SUBJECT Form for a CATEGORY Subject
The CATEGORY-ATTRIBUTES Display Form

This form is used for editing and displaying the list of attribute-types associated with a given subject category. It is a list of the relationships between a CATEGORY subject and the ATTRTYPE subjects defining the attributes for subjects in that category. The CATEGORY-ATTRIBUTES form in edit mode is shown in Figure 41.

The form is accessed by specifying "ATTRTYPE" in the field next to 5-RELATIONSHIPS on the EXTENSIBILITY SUBJECT form and executing the 5-RELATIONSHIP action when displaying the definition for a CATEGORY subject. Note that the list of relationships displayed on the form may not be modified if the CATEGORY is in installed status (because you cannot change the set of attributes for an installed subject category).

![Diagram of CATEGORY-ATTRIBUTES form]

Figure 41. The CATEGORY-ATTRIBUTES Display Form
The CATEGORY-RELTYPES Display Form

This form is used for displaying a list of the installed relationship-types that have been defined for a given category—those installed relationship-types where the category has been specified as either the "left-hand" or the "right-hand" category (or both) for the relationship. This form is fully described in Chapter 4 of OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference and available to general users, but displays information of specific interest to those responsible for developing the definitions of categories and relationship-types. The list displayed on the form cannot be changed or updated.

The form is accessed by specifying "RELTYPE" in the field next to 5-RELATIONSHIP on the EXTENSIBILITY SUBJECT form and executing the 5-RELATIONSHIP action when displaying the definition for an installed CATEGORY subject. This form is shown in Figure 42.

---

**Figure 42. The CATEGORY-RELTYPES Display Form**
Using Display Forms for RELTYPE Subjects

The EXTENSIBILITY SUBJECT Form for a RELTYPE Subject

If you enter the subject name for a RELTYPE subject on the Header form, and use as category name "RELTYPE" (or the category number "21"), the EXTENSIBILITY SUBJECT form is displayed, showing the title "RELTYPE" and the definition of the specified relationship-type (if one exists). You can use this form for entering and editing the definition of a relationship-type, and for obtaining access to the ALIASES, DESCRIPTION, and USER DATA forms to edit the alias names and Description and User Data text for the RELTYPE subject. This form is shown in Figure 43.

This form is the standard EXTENSIBILITY SUBJECT form, and is fully described in Chapter 4 of OS/VS DB/DC Data Dictionary Terminal User’s Guide and Command Reference. Modification of the definition of an installed relationship-type is limited to the DATE and EXPTEXT attributes, and to the Description and User Data text.

Figure 43. EXTENSIBILITY SUBJECT Form for a RELTYPE Subject
The RELTYPE–ATTRIBUTES Display Form

This form is used for editing and displaying the list of attribute-types associated with a given relationship-type. It is a list of the relationships between a RELTYPE subject and the ATTRTYPE subjects defining the attributes ("relationship data") for instances of that type of relationship. This form is shown in edit mode in Figure 44.

The form is accessed by specifying "ATTRTYPE" in the field next to 5-RELATIONSHIPS on the EXTENSIBILITY SUBJECT form and executing the 5-RELATIONSHIP action when displaying the definition for a RELTYPE subject. The list of relationships displayed on the form may not be modified if the RELTYPE is in installed status (because you cannot change the set of attributes for an installed relationship-type).

Figure 44. The RELTYPE–ATTRIBUTES Display Form
Using Display Forms for ATTRTYPE Subjects

The EXTENSIBILITY SUBJECT Form for an ATTRTYPE Subject

If you enter the subject name for an ATTRTYPE subject on the Header form, and use as category name "ATTRTYPE" (or the category number "22"), the EXTENSIBILITY SUBJECT form is displayed showing the title "ATTRTYPE" and the definition for the specified attribute-type (if one exists). You can use this form for entering and editing the definition of an attribute-type, and for obtaining access to the ALIASES, DESCRIPTION, and USER DATA forms to edit the alias names and Description and User Data text for the ATTRTYPE subject. This form is shown in edit mode in Figure 45.

This form is the standard EXTENSIBILITY SUBJECT form, and is fully described in Chapter 4 of OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference. Modification of the definition of an installed attribute-type is limited to the DATE attribute, and to the Description and User Data text.

Figure 45. EXTENSIBILITY SUBJECT Form for an ATTRTYPE Subject
Considerations for SUBJECT-RELATIONSHIPS Forms

The two SUBJECT-RELATIONSHIPS display forms and their use are described in Chapter 4 of OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference. However, there are several points that should be noted about their use with definitions of an attribute-type.

The RELATIONSHIPS action on an EXTENSIBILITY SUBJECT form specifying an ATTRTYPE subject, enables you to access a SUBJECT-RELATIONSHIPS form to locate where the attribute-type is used, both for installed categories and relationship-types, and for those under development. The SUBJECT-RELATIONSHIPS forms can also be accessed from the Header form in the usual manner, by specifying an ATTRTYPE subject and using the RELATIONSHIPS TO field.

With either version of the SUBJECT-RELATIONSHIPS form, the value in the SEQUENCE column displays, for each subject category or relationship-type, the SEQ operand value for the relationship to the attribute-type. Neither the KEYWORD nor the REPEAT attribute values on the relationships are displayed on either display form, but can be viewed with the RELDATA action, because these constitute relationship data for a specific relationship.

Using either version of the form, you can establish a relationship between an individual attribute-type definition and the definition for a CATEGORY subject (or RELTYPE subject), by placing an entry for that subject into an empty row in the table. You can delete a relationship by erasing the relationship keyword (under the RELATIONSHIP column heading) in the appropriate entry. Should you use the above way of associating attribute-type definitions, note that:

- The SEQ operand must be specified in every relationship entry. That is, the SEQ operand is required on every attribute-category or attribute-relationship-type relationship.

- The attribute-type may be related to the same category or relationship-type definition more than once, provided that the attribute-type name and SEQ operand combination is unique.

- You cannot modify or delete any of the relationships between an attribute-type and an installed subject category or relationship-type definition.

If you use the SUBJECT-RELATIONSHIPS forms to establish or examine the relationships that an individual ATTRTYPE subject has with CATEGORY and RELTYPE subjects, remember that you cannot immediately see what other ATTRTYPE subjects are present as attributes of those definitions, or see the relative position that the ATTRTYPE subject holds within the sequence of attributes defined for the CATEGORY or RELTYPE subject.
Chapter 10. Validation Routines

You specify name validation routines as part of a category definition. If you are adding a validation routine to an installation-defined category, you may add it to the category definition either before or after installation. If you are adding a routine to a standard category, you may add it to the corresponding installed category definition that was supplied with the Dictionary.

The validation routines can be written in COBOL, PL/I, or Assembler language. They execute in the same region as the Dictionary. Each validation routine must be link-edited with a Dictionary linkage module that is appropriate for the source language of that routine.

To add a routine to an existing category (installed or standard), you use the command:

\[
\text{ADD CATEGORY (*,category-name,0) VALRTN=routine-name ;}
\]

To change an existing routine's name, you use the command:

\[
\text{CHANGE IN CATEGORY (*,category-name,0) +}
\text{FROM EXISTING TO VALRTN=routine-name ;}
\]

To add a routine to a standard category, you use the command:

\[
\text{ADD CATEGORY (*,SEGMENT,0) VALRTN=CODECHK ;}
\]

To delete a routine from a category, you use the command:

\[
\text{CHANGE IN CATEGORY (*,category-name,0) +}
\text{FROM VALRTN=routine-name TO ;}
\]

Name validation only occurs when the category specified in the command has a corresponding validation routine.

You specify attribute value validation as part of the ATTRTYPE definition. You may add a routine to the definition before it is installed. After installation, you may not change the validation routine. Validation routines cannot be specified for attributes of standard categories or relationship-types.

To add a validation routine to an ATTRTYPE subject definition, you must specify VALIDATE=RTN and specify the name of the validation routine. For example:

\[
\text{ADD ATTRTYPE (P,DATE,0) VALIDATE=RTN VALRTN=DATECHK ;}
\]

Attribute validation occurs when the attribute specified in the command is associated with an ATTRTYPE requiring a validation routine.
Validation Routine Processing

If a validation routine is specified for a subject category, validation will be performed only for those subject names which may be added to the Dictionary. See Figure 11 on page 59 for a summary of commands and functions that result in name validation processing. Consequently, if the validation routine is changed in a way that makes some of the subject names already stored in the category incompatible, the user will still be able to obtain reports on those subjects, and to change their subject names to names that are acceptable to the new version of the routine. The user will, however, be prevented from adding new subjects with "invalid" names, or from establishing new relationships to those subjects.

If a validation routine is specified for an attribute-type, the validation will be performed only for values specified in an ADD or ADD__RELATIONSHIP command, or in the TO operand of the CHANGE__IN or CHANGE__RELATIONSHIP__DATA command. Thus, if the validation routine is changed in a way that makes the values incompatible for some of the subjects already stored, the user will still be able to obtain reports on those subjects, and to change their values to those that are acceptable to the modified routine. The user will, however, be prevented from adding new subject data with invalid values, or from establishing new relationships with invalid relationship data.

Validation routines may not call the Program Access facility. Any attempt to do so will result in a terminating error.

Requirements for Validation Routines

The validation routines can be written in COBOL, PL/I, or Assembler language. They execute in the same region as the Dictionary. This section describes the communication between a validation routine and the Dictionary, and gives the programming requirements and link-editing for use when coding both attribute value and name validation programs. The name validation information in this section applies to routines defined for use with standard categories and installation-defined categories. Attribute value validation is used only with installation-defined attribute-types.

Communication Area for Attribute Value Validation

The Dictionary invokes the routine to validate an attribute value and passes the address of a communication area. This area contains both reference data and the attribute value. On return to the Dictionary, the routine must set a return code in a reserved field in the communication area and, optionally, a message describing the outcome of the validation. The installation-written routine cannot cause a different attribute value to be entered into the data base, that is, no editing or encoding of the value passed to the routine may be reflected in data returned to the Dictionary.

The manner in which the address of the communication area is passed to the validation routine depends on the source language chosen, as described in “Program Entry Points” on page 178.
The format of the communication area is shown in Figure 46.

<table>
<thead>
<tr>
<th>Format</th>
<th>Offset in Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type-code</td>
<td>0</td>
<td>A 1-byte field containing the value 'A' to indicate there is a value of an attribute to be validated.</td>
</tr>
<tr>
<td>attribute-keyword</td>
<td>1</td>
<td>An 8-byte field containing the keyword name that controlled the input of the value. The keyword name is left adjusted in the field.</td>
</tr>
<tr>
<td>msg-string</td>
<td>9</td>
<td>A 69-byte field available for an output message. The field is set to blanks upon entry to the routine.</td>
</tr>
<tr>
<td>attr-status</td>
<td>78</td>
<td>A 1-byte field containing an '*'.</td>
</tr>
<tr>
<td>attr-code</td>
<td>79</td>
<td>A 1-byte field containing the number 22 (X'16').</td>
</tr>
<tr>
<td>attr-name</td>
<td>80</td>
<td>A 31-byte field containing the installed attribute-type user name.</td>
</tr>
<tr>
<td>attr-occ</td>
<td>111</td>
<td>A 1-byte field containing the occurrence number of the attribute-type.</td>
</tr>
<tr>
<td>return-code</td>
<td>112</td>
<td>A binary fullword field, which must be set by the routine before control is passed back to the Dictionary. This field is not initialized by the Dictionary.</td>
</tr>
<tr>
<td>value-length</td>
<td>116</td>
<td>A binary halfword field, containing the maximum length of the value to be validated (the DATAMAX operand value).</td>
</tr>
<tr>
<td>value</td>
<td>118</td>
<td>A 120-byte field containing the value to be validated. The value is in character format and left adjusted, but remains exactly as entered.</td>
</tr>
</tbody>
</table>

Figure 46. Format of Communication Area for Attribute Value Validation

Communication Area for Name Validation

The Dictionary invokes the routine to validate the name of a subject, and passes the address of a communication area. This area contains both the full subject name and the category name. On return to the Dictionary, the routine must set a return code in a reserved field in the communication area, and, optionally, a message describing the outcome of the validation. The installation-written routine cannot cause substitution of any part of the name, and changes made in the communication area fields that specify the subject name and category are ignored.

The manner in which the address of the communication area is passed to the validation routine depends on the source language chosen, as described in "Program Entry Points" on page 178.
The format of the communication area is shown in Figure 47.

<table>
<thead>
<tr>
<th>Format</th>
<th>Offset in Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type-code</td>
<td>0</td>
<td>A 1-byte field containing the value 'N' to indicate that this is a name-validation request.</td>
</tr>
<tr>
<td>category-name</td>
<td>1</td>
<td>An 8-byte field containing the name of the category specified for the subject that is having its name validated. The name of the category is left-adjusted in the field.</td>
</tr>
<tr>
<td>msg-string</td>
<td>9</td>
<td>A 69-byte field available as the text of an output message. The field is set to blanks upon entry to the routine.</td>
</tr>
<tr>
<td>subject-status</td>
<td>78</td>
<td>A 1-byte status code for the subject.</td>
</tr>
<tr>
<td>subject-code</td>
<td>79</td>
<td>A 1-byte field containing either the subject code (for standard categories) or the category number (56 through 255) for installation-defined categories.</td>
</tr>
<tr>
<td>user-name</td>
<td>80</td>
<td>A 31-byte field containing the Dictionary user name that is to be validated; left-adjusted.</td>
</tr>
<tr>
<td>occ-number</td>
<td>111</td>
<td>A 1-byte field containing the occurrence number of the subject.</td>
</tr>
<tr>
<td>return-code</td>
<td>112</td>
<td>A binary fullword field, which must be set by the routine before control is passed back to the Dictionary. This field is not initialized by the Dictionary.</td>
</tr>
<tr>
<td>user-identification</td>
<td>116</td>
<td>A 31-byte field containing the id of the current user, left-adjusted.</td>
</tr>
</tbody>
</table>

**Figure 47. Format of Communication Area for Name Validation**

**Program Entry Points**

The Dictionary uses the entry point name **DBDWUP** when it invokes a validation routine. The corresponding entry statements depend on the chosen source language.

The entry statement for an Assembler language program is:

**ENTRY DBDWUP**

Register 1 points to the address of the **communication area**.
The entry statement for a COBOL program is:

ENTRY 'DBDWUP' USING communication-area

where the operand communication-area is replaced by a data declaration name present in the COBOL program.

The entry statement for a PL/I program is a PROCEDURE statement coded:

DBDWUP: PROC(com_area_ptr) OPTIONS(MAIN);

where com_area_ptr is a pointer to a level-1 structure that defines the communication area. A substitute name could be used for DBDWUP, because the program is entered at the main entry point.

Validation Return Codes and Messages

When control returns to the Dictionary, the validation routine communicates the results of its processing with a return code and an optional message containing up to 69 bytes of text. Three Dictionary message numbers have been reserved for user-provided messages. The messages have an identifying number and severity code; the message text is taken directly from the communication area, as follows:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Message</th>
<th>Severity</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DBD0151</td>
<td>I</td>
<td>Message for information only</td>
</tr>
<tr>
<td>4</td>
<td>DBD0152</td>
<td>W</td>
<td>Message warning the user</td>
</tr>
<tr>
<td>8</td>
<td>DBD0153</td>
<td>E</td>
<td>Message indicating an error</td>
</tr>
</tbody>
</table>

In batch output, one of these messages appears whenever the corresponding text is completed. If no message text is substituted for the blanks passed by the Dictionary, then there is no message output. In the case of return code '8', with or without a message, the Dictionary continues execution as though an 'E' severity error had occurred. Subsequent commands will be flushed unless FLUSH=NO is in effect.

In display forms, information messages are not displayed. When display form input data is processed, only the first message of 'W' or greater severity is displayed. The highlight feature indicates which data has been processed.

The INSTALL, COBOL__IN, and PLI__IN functions will suppress information and warning messages and issue a diagnostic message unique to their respective function.

Because a message does not appear if the optional message text is not specified, the end user may be left in doubt as to the outcome of the validation, or may not know that it occurred. For this reason, you should always issue an appropriate message, especially for conditions of 'W' or 'E' severity.
Installing Validation Routines

You can identify a load module library to hold validation routines instead of placing them in the Dictionary program library. The resulting library must be in the operating system library search order for the Dictionary job step. You may enter and replace members of this library, using standard OS/VS utilities. The member names in the library must correspond to the name declared in the subject category or attribute-type definitions.

Link-Editing Requirements

Each name or value validation routine must be link-edited with a Dictionary linkage module that is appropriate for the source language of that routine. The requirements are identical to those of user programs using the Program Access facility and are given below.

Link-Editing Validation Routine for COBOL

Your validation routine must be link-edited into a library to which the Dictionary has access. This may be the library in which the Dictionary itself resides, or a separate library defined in a STEPLIB or JOBLIB DD statement that is concatenated with the Dictionary execution JCL. It must be link-edited with the DBDWLNC module, to allow the Dictionary to communicate with your program. The primary entry point of the resulting load module is DBDWLNC.

The necessary link-edit control statements are shown below. “valroutinec” is the name of your validation routine (that is, its object module). “userpgm” is the name of the resulting load module; it is the name specified with the VALRTN keyword.

Link-edit control statements are:

//SYSLMOD DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=OLD
//LOADLIB DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=SHR
//OBJLIB DD (user parameters)
//SYSIN DD *
  INCLUDE LOADLIB(DBDWLNC)
  INCLUDE OBJLIB(valroutinec)
  ENTRY DBDWLNC
  NAME userpgm(R)

The link-edit job’s JCL statements should include a DD statement that defines an object library (PDS) that contains your program (that is, its object module). The INCLUDE statement above uses a ddname of OBJLIB.

Note: The ENTRY statement forces the Dictionary linkage module to be the primary entry point of the resulting load module.
Link-Editing Validation Routines for PL/I

Your validation routine must be link-edited into a library to which the Dictionary has access. This may be the library in which the Dictionary itself resides or a separate library defined in a STEPLIB or JOBLIB DD statement that is concatenated with the Dictionary execution JCL. It must be link-edited with the DBDWLNP module, to allow the Dictionary to communicate with your program. The primary entry point of the resulting load module is DBDWLNP. When the Dictionary passes control to your program, the environment is that required for a PL/I MAIN program.

The necessary link-edit control statements are shown below. "valroutinep" is the name of your PL/I program (that is, its object module). "userpgm" is the name of the resulting load module; it is the name specified with the VALRTN keyword.

Link-edit control statements are:

```plaintext
//SYSLMOD DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=OLD
//LOADLIB DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=SHR
//OBJLIB DD (user parameters)
//SYSIN DD *
    INCLUDE LOADLIB(DBDWLNP)
    INCLUDE OBJLIB(valroutinep)
    ENTRY DBDWLNP
    NAME userpgm(R)
```

The link-edit job step's JCL statements should include a DD statement that defines an object library (PDS) for the user-written programs (object modules). The INCLUDE statement above uses a ddname of OBJLIB.

*Note:* The ENTRY statement forces the Dictionary linkage module to be the primary entry point of the resulting load module.

Link-Editing Validation Routines for Assembler

Your validation routine should be link-edited into a library to which the Dictionary has access. This may be the library in which the Dictionary itself resides or a separate library defined in a STEPLIB or JOBLIB DD statement that is concatenated with the Dictionary execution JCL. It must be link-edited with the DBDWNKA module, to allow the Dictionary to communicate with your program. The primary entry point of the resulting load module is DBDWNKA.

The necessary link-edit control statements are given below. "valroutinea" is the name of your program (that is, its object module). "userpgm" is the name of the resulting load module; it is the name specified with the VALRTN keyword.

Link-edit control statements are:

```plaintext
//SYSLMOD DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=OLD
//LOADLIB DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=SHR
//OBJLIB DD (user parameters)
//SYSIN DD *
    INCLUDE LOADLIB(DBDWNKA)
    INCLUDE OBJLIB(valroutinea)
    ENTRY DBDWNKA
    NAME userpgm(R)
```
The link-edit job step's JCL statements should include a DD statement that defines an object library (PDS) for your program (object module). The INCLUDE statement above uses a ddname of OBJLIB.

Note: The ENTRY statement forces the Dictionary linkage module to be the primary entry point of the resulting load module.

Managing Validation Libraries

Subject name and attribute validation routines should be completed and placed in a library before the associated category is in general use. If a Dictionary user attempts to enter subjects into a new category (or relate subjects with a new relationship-type) before the necessary validation routines are in place, a "severe error" response will result.

Take care when modifying and replacing validation routines to ensure that data already residing in the Dictionary will not be made incompatible with the validation criteria. The CHANGE_IN, CHANGE_RELATIONSHIP_DATA, and CHANGE_NAME commands must be used to correct any attribute values or subject names that have been indirectly affected by a change in a validation routine before those values or names may be addressed by any other Dictionary command.

The same attribute validation routine may be used for values of more than one Extensibility attribute-type. The same subject name validation routine may also be used to validate names for subjects in different standard or installation-defined categories. Therefore, if you establish the same naming standards for names of subjects in several different subject categories, you may enforce the standards with a single subject name validation routine.
Chapter 11. Defining Structure-Types

This chapter specifies how to define the name, attributes, and relationships of an installation-defined structure-type. You can define and report on the STRTYPE definition with standard commands or with display forms, as described in this chapter. Refer to "Understanding Dictionary Structures" on page 61 for a description of Dictionary structures and how they are used.

Assuming that a group of resources has been identified as belonging to a structure of interrelated categories and you want to set up a structure-type to control processing, you define a subject that resides in a Dictionary category called STRTYPE. The definition of this subject will control what subject categories are accessed and the manner of subsequent relationships. Processing of COPY, DELETE__STRUCTURE, EXPORT, and STRUCTURE__REPORT commands can be controlled by the content of these definitions.

Defining a STRTYPE Subject Name

Structure-type subjects are defined in the STRTYPE category (category number 23). The Dictionary subject name for a STRTYPE definition has the general form:

(status,,user-name,0)

The status of a structure-type definition may be any of the standard codes while under development, and is '*' when installed.

No subject code is specified in the subject name.

The user-name portion of the subject name is the identifying name. The name can be from 1 to 31 bytes. If the name is EBCDIC, the first character must be alphabetic. For DBCS users, the name must be a uniform string.

All subject names for structure-type definitions must have an occurrence-number of '0'.

Regarding the use of STRTYPE subject names:

- The user-name portion of the subject name becomes the name of the new structure-type. It will be the name known to the general user, and the name used on display forms and in commands. For example, a structure-type named LANGSTR used with a COPY command would be specified as STRUCTURE=LANGSTR.
• You may establish aliases for the name of the structure-type before it is
installed. The aliases become alternative names for the structure-type that the
general user can use in commands or when using display forms.

• If you develop a structure-type in a standard status, you must place it in
production 'P' status before the definition can be installed. The status code is
changed to '**' when the structure-type is installed (that is, made operational
for the general user).

**STRTYPE Definition Operands**

The operands that specify the definition of a STRTYPE subject are of two types:

• Operands that are concerned with documentation text and that can be updated
after the structure-type is installed (DATE, DESC, and USERn).

• Operands that identify relationship-types associated with this structure-type.
Keywords are used to specify relationship data for each relationship. The
relationship-types associated with the structure-type definition become
“permanent” when the structure-type is installed.

The first type of operand is described below. These operands are specified as values for Dictionary keywords in update commands or as fields on display forms. The second type is associated with the new structure-type by relating subjects in the RELTYPE category to it. The definition of these relationships is covered in the section “Relating Relationship-Types to a Structure-Type.”

The DATE, DESC, and USERn operands form the first type of operands. You may enter or update this information at any time during the development of the structure-type definition, or after the definition is installed. DBCS users can use mixed strings to specify the text portion for DESC and USERn operands.

The DATE operand may be used to associate a date with the structure-type
definition. Note that this attribute is not interpreted or maintained by the
Dictionary, it is provided merely for documentation purposes. The value is
specified as six numeric digits. There is no default.

The DESC and USERn (n is a value from '1' to '5') operands are used to define
lines of text documenting the structure-type. The standard Description text and
five sets of User Data are available for a subject specified for STRTYPE.
Alternatively, you may use the DESCRIPTION or USER DATA display forms to enter this text.
Using Commands with Structure-Type Definitions

The operands described in the previous section are specified as keywords with update commands. For your convenience, the keyword names and valid values are summarized in Appendix C of OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference.

Adding and Updating a Definition

Subjects belonging to the STRTYPE subject category may be defined and (with certain restrictions) updated and deleted using update commands.

The ADD and CHANGE__IN commands may be used to define and update the definition of a structure-type in any of the status codes.

For example, to add a STRTYPE named LANGSTR:

```
ADD STRTYPE (E,,LANGSTR,0) DATE=110582 +
   DESC=(1,'STRUCTURE-TYPE FOR LANGUAGE DATA STRUCTURES') ;
```

To change the date:

```
CHANGE__IN STRTYPE (E,,LANGSTR,0) +
   FROM EXISTING TO DATE=111182 ;
```

To establish an alias for the structure-type:

```
ADD_RELATIONSHIP STRTYPE (E,,LANGSTR,0) +
   HAS_ALIAS STRTYPE (E,,LANG,0);
```

After the structure-type has been installed, you cannot add or delete aliases.

The DELETE__DATA command may be used to delete lines within a range or individual lines of the Description or User Data text associated with a structure-type definition (in any status).

Before the structure-type is installed, the CHANGE__NAME command may be used to change the primary or an alias subject name for a structure-type definition. You cannot use the CHANGE__NAME command to change the name of an installed structure-type.

Using the COPY Command

The COPY command can be used to create a copy of a structure-type definition under a new subject name, to copy a structure-type definition from one status to another (but not to the installed status), or to copy an "installed" structure-type definition to create another definition in a different status.
Deleting a Structure-Type Definition

An installation-defined structure-type definition may be deleted using either the DELETE or DELETE_STRUCTURE command for any of the status codes, including '*'. When you use the DELETE command, all established relationships between the STRTYPE subject and the RELTYPE subjects are deleted. There are restrictions for the DELETE command when the structure-type has been installed. If the structure-type has alias names, you must specify DELETE with the AND ALIASES option.

You need to be cautious when using the DELETE_STRUCTURE command. The command may be used to delete the definition of a structure-type. If you use a structure-type with the command that follows relationships, the related relationship-types could be candidates for deletion. However, you will probably not want those relationship-types to be deleted. They would still be of general value to other developers of new structure-types, so that you might delete certain of the relationships before using the DELETE_STRUCTURE command. A further consideration is that relationships may be followed from a deleted RELTYPE subject to select candidate ATTRTYPE subjects that are related to the RELTYPE definition. Again, you will probably not want these definitions to be deleted.

Relating Relationship-Types to a Structure-Type

You specify the categories to be included in the structure-type and the relationships to be traversed by defining the relationship-types in the structure-type definition. To do this, relationship-type definitions have to be related to a structure-type subject. You build a set of individual relationship-types, placing them in a desired order. Four operands used to specify the attributes of the relationship of the individual RELTYPE subjects to the STRTYPE subject are:

- The SEQ operand is required as a sequencing attribute for relationship update commands.
- The SUPERCAT, FOLLOWUP, and FOLLOWDN operands define the rules for processing the relationships.

You can specify a maximum of 255 RELTYPE subjects related to a STRTYPE.

Using Relationship Commands with Structure-Type Subjects

You can use the ADD_RELATIONSHIP, CHANGE_RELATIONSHIP_DATA, and DELETE_RELATIONSHIP commands to create and manipulate the relationships between RELTYPE subjects and STRTYPE subjects. The DELETE_RELATIONSHIP_DATA command cannot be used to delete data associated with these relationships commands. You can also define and alter these relationships, using a special display form, the STRTYPE-RELTYPES form. Altering relationships between the definition of an installed structure-type and the associated relationship-type definitions is prohibited. Refer to “Procedure for Changing an Installed Structure-Type” on page 200 for the procedure for changing an installed STRTYPE.
The standard ADD_RELATIONSHIP command is used to establish a relationship between a structure-type definition and the definition for a relationship-type. The command requires the sequence attribute keyword SEQ and the relationship keyword CONTAINS. For example:

```
ADD_RELATIONSHIP STRTYPE (E,,LANGSTR,0) +
   CONTAINS RELTYPE (*,,user-defined-relationship,0) +
   SEQ=1 SUPERCAT=N ;
```

The “inverse” relationship keyword used for RELTYPE-STRTYPE relationships (that is, the keyword used in relationship commands where the relationship-type is identified first) is CONTAINED_IN. For example, to achieve the same result as the previous command:

```
ADD_RELATIONSHIP RELTYPE (*,,user-defined-relationship,0) +
   CONTAINED_IN STRTYPE (E,,LANGSTR,0) +
   SEQ=1 SUPERCAT=N ;
```

SEQ is a sequencing operand that establishes an order within the list of relationship-types. Although duplicate SEQ values can be specified, by giving each relationship-type a unique number you make the order predictable. Specifying unique values for the relationship-types will cause them to appear in ascending order of SEQ value in all reports and display forms. It will not have any effect on the order in which relationships in the structure are processed.

Before installing, the CHANGE_RELATIONSHIP_DATA command may be used to change any (or all) of the SEQ, SUPERCAT, FOLLOWUP, and FOLLOWDN operand values associated with a RELTYPE subject relationship.

The DELETE_RELATIONSHIP command is used to delete the relationship between a structure-type definition and the definition of one of its associated relationship-types.

Although any of the above relationship commands can specify a RELTYPE subject with any standard status code, you usually work with definitions that have already been installed—by including the * status code as part of the RELTYPE subject name. This is because there is a requirement for all related RELTYPE subjects to be in installed status before the STRTYPE definition can be installed.

It is possible that your development of a structure-type might be done in parallel with relationship-type definitions. If all definitions were developed at production 'P' status, you would be able to use subject-specific reports to record the progress of the development. However, to achieve a successful structure-type install, you would have to:

1. Install each relationship-type definition.
2. Install the structure-type.
Defining Processing Options for Relationship Types in a Structure Type

The previous section indicated that there are four operands that specify relationship data for each relationship-type associated with a structure-type definition. Three of these operands (SUPERCAT, FOLLOWUP, and FOLLOWDN) directly influence the processing performed by commands that specify the use of a structure-type. They provide your installation with great flexibility for the data selected for processing within Dictionary structures. They allow you to:

- Identify the direction of a relationship
- Limit the scope of processing for commands referencing structure-types, such as COPY, DELETE__STRUCTURE, EXPORT, and STRUCTURE__REPORT

This section defines how you specify these operands. OS/VS DB/DC Data Dictionary Applications Guide contains more of the strategy for their use and what advantages they offer. You should be familiar with the way COPY, DELETE__STRUCTURE, EXPORT, and STRUCTURE__REPORTS use the structure-type definition so that you can select the operand values necessary to get the results you want.

Designating a Superior Category

The SUPERCAT operand enables you to declare that a relationship-type should be processed with one of the pair of subject categories in the relationship-type considered superior to the other. The values you can specify are:

L The left-hand category (the category specified for LCATNAME in the RELTYPE definition) should be considered superior.

R The right-hand category (the category specified for RCATNAME in the RELTYPE definition) should be considered superior.

N Neither category is superior.

For example, the relationship-type:

PGMR FIXES MODULE

You can designate the PGMR category as superior by using SUPERCAT=L. By specifying SUPERCAT=R, you can declare the MODULE subject to be superior.
If a relationship-type definition was originally defined with the property of being "DIRECTED", the kind and choice of which category is superior are fixed by the definition. In this case, your specification of the SUPERCAT value would have to agree. For example, the directed relationship:

**DATABASE WITH SEGMENT**

The value would have to be SUPERCAT=L because the DATABASE WITH SEGMENT relationship-type is defined as being directed. However, this specification is not checked until INSTALL command processing occurs.

**Processing Direction Options**

A second operand that controls processing is FOLLOWDN. This indicates that, when positioned at a superior category's subject, relationships should be followed in a downward direction, from superior subject to subordinate subject; that is, the relationships are processed and the subordinate subject is considered for further processing. The values for the FOLLOWDN operand are:

- **Y** Yes (Y), relationships are to be followed in a downward direction, superior to subordinate.
- **N** No (N), relationships are not to be followed in a downward direction.

A similar interpretation is used for the FOLLOWUP operand. The values for the FOLLOWUP operand are:

- **Y** Yes (Y), relationships are to be followed in an upward direction, subordinate to superior.
- **N** No (N), relationships are not to be followed in an upward direction.

The effect of the FOLLOWDN and FOLLOWUP operands is to control the scope of processing when positioned at any Dictionary subject whose category is named in the structure-type definition. At a given subject, the Dictionary is aware (from SUPERCAT) that the subject is in a superior or subordinate category and knows whether processing can follow relationships.

**Required Operand Values**

If you define SUPERCAT as having no processing direction, then the FOLLOWDN and FOLLOWUP operands must both be N. Suppose you had two relationship-types:

**MODULE UPDATED_BY PGMR**
**PGMR MEMBER_OF DEPT**

If you were interested in reporting which programmers had updated a module, you would not necessarily want to follow that relationship to include (in the report) the department of each programmer. By specifying SUPERCAT=N for the PGMR MEMBER_OF DEPT relationship-type, the corresponding FOLLOWDN=N or FOLLOWUP=N operand would cause any relationships from PGMR subjects not to be followed and DEPT subjects would not occur in the report.
If you are working with a symmetrical relationship, for example,

\textit{JOB WITH JOB}

where both of the categories in the relationship-type are the same and the forward and inverse keywords are identical, then there is a restriction: SUPERCAT must be N. In this case, it is not possible for the Dictionary to determine a processing direction, because all distinguishing aspects of the relationship are identical.

When you install the structure-type definition, all the above restrictions pertaining to DIRECTED and symmetrical relationships are validated; any inconsistencies will be detected and successful installation will be prevented.

The INSTALL command supplies appropriate defaults as follows:

<table>
<thead>
<tr>
<th>If Relationship-Type Definition Is:</th>
<th>SUPERCAT Is:</th>
<th>Default for FOLLOWDN</th>
<th>Default for FOLLOWUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECTED</td>
<td>L (left-hand)</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Not DIRECTED</td>
<td>N (neither)</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Summary of Structure-Type Characteristics

- Structure-type definitions specify an ordered list of relationship-types.
- Only subjects in relationships named in the list of relationship-types are processed when under the control of that structure-type.
- Using the SUPERCAT operand, each relationship-type can have one of its pair of categories designated a superior category, or neither category superior, consistent with the relationship-type definition.
- Using the FOLLOWDN and FOLLOWUP operands, each relationship-type that has or is assigned a processing direction has rules for whether to follow relationships in an upward or downward direction.

An Example of Structure-Type Definition

Suppose you are interested in a less extensive structure. You need to obtain reports of individual programmer skills or skills for an entire department. When you are defining new department members, you may want to use the COPY command to copy the relationships to skills of an existing member with similar skills. To do this, you need to eliminate any of the other relationships that the PGMR subjects might have to subjects in other categories, such as modules. You do not want these relationships to be part of the processing.
To achieve this, you define a structure-type, named TASK. The TASK structure-type is shown in Figure 16 on page 67. This structure contains the three relationship-types:

DEPT HAS PGMR
PGMR HAS SKILL
SKILL REQUIRES EDUC

Using commands, you can:

Add the STRTYPE subject and its description:

ADD STRTYPE (P,,TASK,0) DATE=102082 +
DESC=(1,'THIS STRUCTURE-TYPE CONTAINS +
ALL RELATIONSHIP-TYPES') ;
ADD STRTYPE (P,,TASK,0) +
DESC=(2,'TO DESCRIBE PERSONNEL DATA ON PROGRAMMERS') ;

Add an alias for the STRTYPE subject:

AR STRTYPE (P,,TASK,0) +
HAS_ALIAS STRTYPE (P,,PERSONNEL-DATA-ON-PGMRS,0) ;

Define all relationship-types in the structure-type and their processing options:

AR STRTYPE (P,,TASK,0) +
CONTAINS RELTYPE (*,,DEPT/HAS/PGMR,0) +
SEQ=10 SUPERCAT=L FOLLOWUP=Y FOLLOWDN=Y ;

AR STRTYPE (P,,TASK,0) +
CONTAINS RELTYPE (*,,PGMR/HAS/SKILL,0) +
SEQ=20 SUPERCAT=L FOLLOWUP=Y FOLLOWDN=Y ;

AR STRTYPE (P,,TASK,0) +
CONTAINS RELTYPE (*,,SKILL/REQUIRES/EDUC,0) +
SEQ=30 SUPERCAT=L FOLLOWUP=N FOLLOWDN=Y ;

You can declare a processing direction from DEPT to PGMR to SKILL to EDUC, as shown below. In this case, the FOLLOWDN operands are specified as YES.

When you use this structure, all relationships will be processed.

DEPT ---+ PGMR ---+ SKILL ---+ EDUC

In the upward direction, given FOLLOWUP=Y, only the following path can be processed:

SKILL ---+ PGMR ---+ DEPT

You would not use a subject in the EDUC category as the starting point for processing because its upward relationships would not be followed.

An attempt to use a PROGRAM subject as a starting subject with this structure-type, specified by an option STRUCTURE=TASK on a command, would fail because the category is not part of any of the relationship-types in the structure-type.
Reporting Structure-Type Development

The GUIDE report documents the structure-type for the end user and includes the relationship-types associated with structures of this type and the processing rules for each one. Therefore, this report supplements the command and display form information given in *OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference.*

The GUIDE report can be obtained at any point during the development of a structure-type as optional output from an INSTALL command. You use the UPDATE=NO option to prevent the structure-type installation processing from being completed. If the structure-type is installed, the GUIDE report can be obtained directly with the REPORT command.

The subject-specific and GUIDE report are described in the sections that follow. The Installation Processing report follows the section on the use of the INSTALL command.

Structure-Type Definition Reports

Dictionary subject-specific reports are obtained using the REPORT command as described in Chapter 6 of *OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference.* The REPORT command can be used in the usual manner to obtain full or partial information about the definition of a structure-type (or, more precisely, about the STRTYPE subject), either before or after it has been installed. To obtain display form equivalent reports, the DETAIL option can specify ALIAS for STRTYPE subject aliases, DESC for the Description text, or USERn for a set of User Data text. The standard DETAIL options for subjects in installation-defined categories on the REPORT command can be used to restrict the information to be included in the output report.

Example of a Subject-Specific STRTYPE Report

Figure 48 on page 193 shows an example of a subject-specific report for an installed PGMRRELS structure-type. This is the standard Dictionary subject-specific report for an Extensibility subject.

Note the following characteristics of the report:

- The ALIASES section shows an alias name for the STRTYPE subject: (*,RELATIONSHIPS__TO__PGMRS,0)
  - All Description data is included.
  - Value of the DATE operand.
• Under the RELATIONSHIPS section:
  
  – CONTAINS RELTYPE is the only type of relationship that can be defined for a structure-type.
  
  – The user-name of each relationship-type will be the primary name.
  
  – The sequence number of each relationship-type (flagged by *SEQ) is given but not the relationship data (SUPERCAT, FOLLOWUP, FOLLOWDN). This can be in the report if RELDATA=YES is specified.

To obtain this report, the following command should be entered:

```
REPORT STRTYPE (*,,PGMRRELS,0) ;
```

```
** * * * * * * * * DB/DC DATA DICTIONARY REPORT 01/22/84 10:59:23
SUBJECT SPECIFIC PAGE: 01

DICTIONARY DATA BASE: EXT CATEGORY: STRTYPE

NAME: * PGMRRELS 0

ALIASES: * RELATIONSHIPS_TO_PGMRS 0

DESCRIPTION:

001 THIS STRUCTURE-TYPE CONTAINS ALL RELATIONSHIP-TYPES IN WHICH
005 THE CATEGORY 'PGMR' PARTICIPATES

ATTRIBUTES:

DATE: 100182

RELATIONSHIPS:

CONTAINS RELTYPE * PGMR/OWNS/MODULE 0 *SEQ: 002
CONTAINS RELTYPE * PGMR/UPDATES/MODULE 0 *SEQ: 005
CONTAINS RELTYPE * DEPT/HAS/PGMR 0 *SEQ: 010
CONTAINS RELTYPE * PGMR/HAS_BACKUP/PGMR 0 *SEQ: 015

USER DATA 1:

001 LINE ONE OF USER DATA

* * * END-OF-REPORT * * *

Figure 48. Example of a Subject-Specific STRTYPE Subject Report
GUIDE Report for Structure Types

The structure-type GUIDE report contains information about an installed structure-type that users of the Dictionary need when they control processing of the COPY, DELETE__STRUCTURE, EXPORT, and STRUCTURE__REPORT commands. The GUIDE report contains:

- The primary name and any aliases for the structure-type.
- The Description data that has been stored for the definition.
- The value of the DATE attribute.
- A list of relationship-types occurring in the definition, listed in the order determined by the SEQ keyword. For each relationship-type, the following information is provided (report headings are given in parentheses):
  - The names of the "left-hand" and "right-hand" categories (LEFT CATEGORY and RIGHT CATEGORY)
  - The forward and inverse keyword names (FORWARD NAME and INVERSE NAME)
  - An indication (in the DIRECTED column) of whether or not the relationship is "directed"

  N Indicates that the relationship is not directed.

  Y Indicates that the relationship is directed, so status rules are to be enforced. The left-hand category (LEFT CATEGORY) is considered to be the superior category.

  - An indication of a superior category in the relationship-type (SUPERCAT column):

    N Neither category is superior.

    L The left-hand category is superior.

    R The right-hand category is superior.

  - Indications (under the FOLLOW columns) of whether or not to follow relationships when processing both upward and downward relationships (UP and DOWN columns, respectively)
Example of a Structure-Type GUIDE Report

Figure 49 is an example of the format of a GUIDE report for a structure-type, the PGMRRELS subject in the STRTYPE category. It shows how the relationship data for each relationship-type in the structure-type definition is reported. This is the same subject illustrated in the preceding subject-specific report.

* * * * * * * DB/DC DATA DICTIONARY REPORT 11/13/85 10:41:57
STRTYPE GUIDE PAGE: 01

STRUCTURE-TYPE: PGMRRELS
ALIASES: RELATIONSHIPS_TO_PGMRS
DESCRIPTION:

THIS STRUCTURE-TYPE CONTAINS ALL RELATIONSHIP-TYPES IN WHICH
THE CATEGORY 'PGMR' PARTICIPATES

DATE: 100182

RELATIONSHIP-TYPES:

<table>
<thead>
<tr>
<th>LEFT CATEGORY</th>
<th>FORWARD NAME</th>
<th>RIGHT CATEGORY</th>
<th>INVERSE NAME</th>
<th>DIRECTED</th>
<th>SUPERCAT</th>
<th>FOLLOW UP</th>
<th>FOLLOW DOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGMR</td>
<td>OWNS</td>
<td>MODULE</td>
<td>OWNED_BY</td>
<td>Y</td>
<td>L</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>PGMR</td>
<td>UPDATES</td>
<td>MODULE</td>
<td>UPDATED_BY</td>
<td>Y</td>
<td>L</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>DEPT</td>
<td>HAS</td>
<td>PGMR</td>
<td>IN</td>
<td>N</td>
<td>L</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>PGMR</td>
<td>HAS_BACKUP</td>
<td>PGMR</td>
<td>BACKUP_FOR</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

* * * END-OF-REPORT * * *

Figure 49. Example of a Structure-Type GUIDE Report

Installing Structure-Type Definitions

When you have completed the preparation of a structure-type definition and have ascertained that the required relationship-types have been included and that the processing rules are appropriate, you need to perform the installation process for the definition.

For a structure-type, this is done with the INSTALL command, using the STRTYPE category as its first operand value. The INSTALL STRTYPE command does not allow you to specify a REPLACE option and does not produce a FORMAT report. PROCRT does not produce ALTER, TRUNC, or CHECK sections of the processing report. However, all other operands are the same.

To complete the installation, the STRTYPE subject must be in production status 'P'. You also need to specify UPDATE=YES. The default for the command is to perform validation processing without installing the definition (UPDATE=NO).
Although the command performs validation processing, it does have some required conditions before a successful installation can be obtained:

- All relationship-types associated with the structure-type definition must be already installed only when UPDATE=YES.

- No relationship-type can occur more than once in the definition of the structure-type.

- No relationship-type that is installed with the DIRECTED attribute of 'Y' can have a SUPERCAT value of 'N'. In addition, the direction of the processing must match that of the relationship-type definition, that is, the FOLLOWUP and FOLLOWDN values must be consistent with SUPERCAT.
The INSTALL STRTYPE Command

This command is used to install and validate STRTYPE subjects. Successful validation allows the command to establish control information, internal to the Dictionary, for the definition of the structure-type. The INSTALL STRTYPE command is available in both batch and online command processing.

The general format of this command is:

```
INSTALL - STRTYPE - structuretype +
```

Required

Optional

```
> UPDATE= NO | YES | PROCRPT= ALL | VALIDATE | RPT= GUIDE | NONE
> DEST= \__________________________;
```

The options for the command are explained below.

INSTALL

The command verb. There is no abbreviated form.

STRTYPE

A required operand that specifies a STRTYPE subject is to be installed.

structuretype

A required operand that specifies the subject name of the structure-type definition. The status code of the subject must be 'P' when UPDATE=YES is specified. For DBCS users, the user name portion must be a uniform string.

UPDATE= NO | N | YES | Y

An optional operand that specifies whether or not the structure-type definition will be installed in the Dictionary. If UPDATE=YES, the definition will be installed. If UPDATE=NO, the definition will be validated, the “Installation Processing” report (and, optionally, a prototype of the GUIDE report) will be generated, but the definition will not be placed in installed status, and there will be no modification of the current Dictionary data. If omitted, NO is the default.

PROCRPT= ALL | VALIDATE | NONE

An optional operand that specifies the sections of the Installation Processing report to be generated. Only one detail may be specified. The valid values are:

ALL | VALIDATE  Requests the “Validation Processing” section of the report. This section contains information about the validation of the structure-type definition.
NONE Requests that the "Validation Processing" section of the report not be generated.

Regardless of the value specified, the "Installation Summary" section of the report will always be generated, and messages will be issued. If omitted, the default is ALL.

**RPT=GUIDE | NONE**

An optional operand that specifies whether or not the GUIDE report for the structure-type definition is to be generated. If **RPT=GUIDE** is specified, the GUIDE report will be generated. If **UPDATE=NO**, a prototype of the GUIDE report will be produced. If **RPT=NONE** is specified, the GUIDE report will not be generated. If omitted, the default is **GUIDE**.

**DEST=L | T**

An optional operand used to specify the destination for the processing report generated by the INSTALL command. Valid values are:

- L Line Printer
- T Terminal

If omitted, the default is 'L' for batch and 'T' for online.

*Note:* The REPLACE operand is not applicable for this form of the INSTALL command.

**Validations Performed by the INSTALL Command for a Structure-Type**

The structure-type definition is first checked to see if it exists as a STRTYPE subject.

Validation of the structure-type definition involves a number of different checks and tests, including:

- If **UPDATE=YES**, the subject must have an assigned status of 'P'. Failure to do so causes the installation to fail. If **UPDATE=YES** is specified, the definition is valid, and the check is successful; the definition is automatically placed in status '**'. If **UPDATE=NO**, the status code is not changed.

- Checks are performed to ensure that the primary name and any aliases do not conflict with any names of any other installed structure-types.

- If **UPDATE=YES**, each relationship-type is checked to see if it is in installed status.

- A check is made to keep any relationship-type definitions from occurring more than once in the set of relationship-types related to the structure-type.

- The SUPERCAT specification and FOLLOW attributes are checked to see if they conflict with the relationship-type's specification, and appropriate defaults are applied. If the relationship-type is not installed and the required DIRECTED operand has not been specified, no defaults are provided.
Should any aspect of the structure-type definition fail to satisfy any of the validation criteria, Installation Processing will continue for validation purposes only, but the structure-type will not be installed.

Report for Installing a Structure-Type

The Installation Processing report for a structure-type follows the same sequence as that for other installation-defined entities, but has a section entitled "RELTYPES" in place of the "ATTRIBUTES" section. This section contains the status and primary subject names of the RELTYPE subjects related to the STRTYPE subject. The sequence number for each instance of a related RELTYPE subject is given (under the SEQ heading) and, to the right, the relationship data.

Diagnostic and installation summary messages are included where necessary.

Figure 50 is an example of a typical Installation Processing report for a simple structure-type named PGMRELS—the same structure-type illustrated in the GUIDE report in Figure 49 on page 195. You will notice that the FOLLOWUP value for the PGM/UPDATES/MODULE relationship had not been established and a default value of 'Y' was supplied by installation processing.

```
INSTALLATION PROCESSING REPORT  11/13/85  13:45:00
PAGE: 0001

COMMAND OPTIONS:

INSTALLATION OF STRTYPE - PGMRELS

UPDATE: YES     DEST: L
*REPLACE: NO   *REPORT: GUIDE
PROCRPT: ALL

NOTE: "*" MEANS DEFAULT VALUE TAKEN

VALIDATION PROCESSING:

ALIAS: RELATIONSHIPS_TO_PGMRS

RELTYPES:

<table>
<thead>
<tr>
<th>ST</th>
<th>USERNAME</th>
<th>SEQ</th>
<th>SUPERCAT</th>
<th>FOLLOW</th>
<th>UP</th>
<th>DOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PGM/OWN/MODULE</td>
<td>002</td>
<td>L</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PGM/UPDATES/MODULE</td>
<td>005</td>
<td>L</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>DBD5592 I FOLLOWUP VALUE DEFAULTED TO Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEPT/HAS/PGMR</td>
<td>010</td>
<td>L</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PGM/HAS_BACKUP/PGMR</td>
<td>015</td>
<td>N</td>
<td></td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

INSTALLATION SUMMARY:

DBD5500 I INSTALLATION OF STRTYPE P PGMRELS 0 COMPLETED

**** END OF INSTALLATION PROCESSING REPORT ****
```

Figure 50. Example of a Structure-Type Installation Report
Procedure for Changing an Installed Structure-Type

The developer of a structure-type definition may want to clarify the documentation and add usability information to a structure-type that has been installed. Documentation portions of the definition can be modified. The DATE, Description, and User Data text can be added to or altered.

If you need to make any other changes to the definition of an installed structure-type (such as, the set of relationship-types included in the definition), you must delete the installed structure-type and define and install a new one with the characteristics desired. There is no provision for modifying the definition, using a reinstall process. The REPLACE option is not available for the INSTALL STRTYPE command.

To reestablish a structure-type definition requires that:

- A replacement STRTYPE subject be prepared with a standard status code and all required relationship-types
- The existing structure-type be deleted
- The replacement definition be installed

You might consider using the COPY command to obtain a base version of the definition for modification, before deletion of the STRTYPE subject, especially if there is a large number of relationship-types in the structure-type definition.

Display Forms for Defining Structure-Types

Installation-defined structure-types, with their associated relationship-type definitions, can be completely defined with display forms.

- You establish the definition for a STRTYPE subject with an EXTENSIBILITY SUBJECT form.
- You relate the RELTYPE subjects with a special relationships form, the STRTYPE-RELTYPES form.

It is assumed that you are familiar with the general hierarchy of display forms and the online operation of the displays, as described in Chapter 3 of OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference.

You can also perform the installation process by using the Dictionary Command form.
The Extensibility Subject Form for a STRTYPE Subject

You can access an EXTENSIBILITY SUBJECT display form for a STRTYPE subject directly from the Header form. If your installation has user access control active, you can use a Sign-On Header form, which shows the STRTYPE category with its category number of 23. In either case, you may enter the number 23 or STRTYPE and the name of the STRTYPE subject to obtain access to the STRTYPE display form.

You can use this form for entering and editing the definition of a structure-type, and for obtaining access to the ALIASES, DESCRIPTION, and USER DATA forms to make appropriate entries.

The blank form is shown in Figure 51.

---

Figure 51. Extensibility Form for a STRTYPE Subject

For information on the use of this form, see OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference.
The STRTYPE–RELTYPES Display Form

This form is used for editing and displaying the list of relationship-types associated with a given structure-type definition. The list contains those RELTYPE subjects already related to the STRTYPE subject at the top of the form. If the structure-type is not installed, you can add to, delete, or modify the entries in the list.

The form is accessed from the STRTYPE subject form by using the RELTYPE option of the RELATIONSHIPS action on the EXTENSIBILITY SUBJECT form when displaying the definition of a structure-type subject. Alternatively, the form may be accessed from the Header form by specifying a STRTYPE subject and "RELTYPE" in the -RELATIONSHIPS TO line.

The blank form in edit mode is shown in Figure 52.

Figure 52. The STRTYPE–RELTYPES Display Form

For an explanation of the use of this form, see OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference.
The Default Structure-Type

A structure-type must be defined in order to drive structure-dependent functions. This structure-type may be one of those defined by your installation or the one supplied with the Dictionary. If you do not specify a structure-type name on those commands that process structures, the default structure-type will be used.

The default structure-type is named DEFAULT. You may use the one supplied with the Dictionary or create your own (see “Modifying the DEFAULT Structure-Type” on page 206). The default structure-type supplied with the Dictionary contains all of the relationship-types supplied with the Dictionary. It has an alternative name of DICTIONARY__SUPPLIED__STRTYPE.

Figure 53 on page 204 summarizes all these relationships and shows which categories participate in relationships with direction.

Figure 54 on page 205 describes the default structure-type in GUIDE report format. The relationship-types in this report that have an asterisk preceding the category indicate those relationship-types for which status rules will be enforced. These relationship-type definitions are DIRECTED.
<table>
<thead>
<tr>
<th>Category</th>
<th>Has Relationship To Category</th>
<th>Has Subordinate Category</th>
<th>Has Forward Keyword Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSDEF</td>
<td>All standard</td>
<td>TRANSACT</td>
<td>WITH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PSB</td>
<td>WITH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATABASE</td>
<td>WITH</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>All standard</td>
<td></td>
<td>WITH</td>
</tr>
<tr>
<td>JOB</td>
<td>All standard</td>
<td></td>
<td>WITH</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>All standard</td>
<td>MODULE</td>
<td>WITH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANSACT</td>
<td>WITH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PSB</td>
<td>WITH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATABASE</td>
<td>WITH</td>
</tr>
<tr>
<td>MODULE</td>
<td>All standard</td>
<td></td>
<td>WITH</td>
</tr>
<tr>
<td>TRANSACT</td>
<td>All standard</td>
<td></td>
<td>WITH</td>
</tr>
<tr>
<td>PSB</td>
<td>All standard</td>
<td></td>
<td>WITH</td>
</tr>
<tr>
<td>PCB</td>
<td>All standard</td>
<td>DATABASE</td>
<td>WITH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEGMENT</td>
<td>WITH</td>
</tr>
<tr>
<td>DATABASE</td>
<td>All standard</td>
<td></td>
<td>WITH</td>
</tr>
<tr>
<td>SEGMENT</td>
<td>All standard</td>
<td>SEGMENT</td>
<td>HAS_VIEW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ELEMENT</td>
<td>WITH</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>All standard</td>
<td>ELEMENT</td>
<td>CONTAINS</td>
</tr>
<tr>
<td>STRTYPE</td>
<td>RECTYPE</td>
<td></td>
<td>CONTAINS</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>ATTRTYPE</td>
<td></td>
<td>HAS</td>
</tr>
<tr>
<td>RELTYPE</td>
<td>ATTRTYPE</td>
<td></td>
<td>HAS</td>
</tr>
<tr>
<td>ATTRTYPE</td>
<td>CATEGORY</td>
<td></td>
<td>(None)</td>
</tr>
<tr>
<td></td>
<td>RELTYPE</td>
<td></td>
<td>(None)</td>
</tr>
<tr>
<td>DDUSER</td>
<td>(None allowed)</td>
<td></td>
<td>(None)</td>
</tr>
</tbody>
</table>

Figure 53. Dictionary-Supplied Relationships
STRUCTURE-TYPE: DEFAULT

ALIASES: DICTIONARY_SUPPLIED_STRTYPE

DESCRIPTION:

ECI DEFINITION FOR THE STRTYPE SUBJECT - DEFAULT
THIS STRUCTURE-TYPE CONTAINS ALL RELATIONSHIP-TYPES SUPPLIED WITH
THE DICTIONARY.

DATE: 010182

RELATIONSHIP-TYPES:

<table>
<thead>
<tr>
<th>LEFT CATEGORY</th>
<th>FORWARD NAME</th>
<th>RIGHT CATEGORY NAME</th>
<th>INVERSE NAME</th>
<th>DIRECTED</th>
<th>SUPERCAT</th>
<th>FOLLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRTYPE</td>
<td>CONTAINS</td>
<td>RELTYPE</td>
<td>CONTAINED_IN</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>*RELTYPE</td>
<td>HAS</td>
<td>ATTRTYPE</td>
<td>APPLIES_TO</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>*CATEGORY</td>
<td>HAS</td>
<td>ATTRTYPE</td>
<td>APPLIES_TO</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SYSDEF</td>
<td>WITH</td>
<td>SYSDEF</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSDEF</td>
<td>WITH</td>
<td>JOB</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSDEF</td>
<td>WITH</td>
<td>PROGRAM</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSDEF</td>
<td>WITH</td>
<td>MODULE</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>*SYSDEF</td>
<td>WITH</td>
<td>TRANSACT</td>
<td>WITH</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>*SYSDEF</td>
<td>WITH</td>
<td>PSB</td>
<td>WITH</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SYSDEF</td>
<td>WITH</td>
<td>PCB</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>*SYSDEF</td>
<td>WITH</td>
<td>DATABASE</td>
<td>WITH</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SYSDEF</td>
<td>WITH</td>
<td>SEGMENT</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSDEF</td>
<td>WITH</td>
<td>ELEMENT</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>WITH</td>
<td>SYSTEM</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>WITH</td>
<td>SYSDEF</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>*SYSTEM</td>
<td>WITH</td>
<td>JOB</td>
<td>WITH</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>WITH</td>
<td>PROGRAM</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>WITH</td>
<td>MODULE</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>WITH</td>
<td>TRANSACT</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>WITH</td>
<td>PSB</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>WITH</td>
<td>PCB</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>WITH</td>
<td>DATABASE</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>WITH</td>
<td>SEGMENT</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>WITH</td>
<td>ELEMENT</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>JOB</td>
<td>WITH</td>
<td>JOB</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>*JOB</td>
<td>WITH</td>
<td>PROGRAM</td>
<td>WITH</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>JOB</td>
<td>WITH</td>
<td>MODULE</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>JOB</td>
<td>WITH</td>
<td>TRANSACT</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>JOB</td>
<td>WITH</td>
<td>PSB</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>JOB</td>
<td>WITH</td>
<td>PCB</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>JOB</td>
<td>WITH</td>
<td>DATABASE</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>JOB</td>
<td>WITH</td>
<td>SEGMENT</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>JOB</td>
<td>WITH</td>
<td>ELEMENT</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>WITH</td>
<td>PROGRAM</td>
<td>WITH</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>*PROGRAM</td>
<td>WITH</td>
<td>MODULE</td>
<td>WITH</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Figure 54 (Part 1 of 2). GUIDE Report for Dictionary-Supplied Default Structure-Type
Modifying the DEFAULT Structure-Type

You can make modifications to the Dictionary-supplied structure-type. Then whenever an end user uses the default for a STRUCTURE operand on a command or supplies DEFAULT as the name, processing is controlled by the content of that definition. The DEFAULT structure-type definition will then determine how the Dictionary processes the structure dependent functions, such as COPY, DELETE_STRUCTURE, EXPORT, and STRUCTURE_REPORT.

The direction specified in a relationship-type, for example, will affect what subjects of the related categories are copied. To determine what effects modification will have on processing, refer to OS/VS DB/DC Data Dictionary Applications Guide for more information on working with structures and structure-dependent functions.
Procedure for Changing the DEFAULT Structure-Type

You need to follow the procedure outlined earlier in this chapter if you want to modify the Dictionary-supplied structure-type. The steps in that procedure are:

- Copy the installed STRTYPE subject (*,DEFAULT,0) and its relationships to a production status.

- Make any desired changes (additions, deletions, processing direction options) to the 'P' status definition.

- Delete the installed definition.

- Install the structure-type with UPDATE=NO.

- After thoroughly checking the Installation Processing report, complete the installation, using UPDATE=YES.

If you want to restore the default structure-type originally supplied with the Dictionary, you can resubmit the command stream provided during installation of the Dictionary.

Alterations to the DEFAULT STRTYPE Subject

For your evaluation, the following types of changes may be of value when altering your installation’s default structure-type:

- Ability to follow a relationship from a standard category subject to another standard category, for example, SYSTEM to DATABASE.

- Ability to follow relationships for a STRTYPE subject to the definitions for its related RELTYPE subjects.

- Added installation-defined relationships to be followed from standard categories.

- Added relationships involving installation-defined categories.

- Deletion of standard relationships you do not use or that have no direction.

- Added alias names for the structure-type.

Remember, a symmetrical relationship cannot have a superior category.

Structure-Type Considerations

The structure processing functions (EXPORT, COPY, DELETE__STRUCTURE, and STRUCTURE__REPORT) use a structure-type definition to direct their processing. The degree of sensitivity to the structure-type contents differs for each function. There are two reasons why you may want to construct one or more special structure-types:
• DELETE_STRUCTURE uses the structure-type to determine those subject categories that are subordinate to the subject currently being processed. You can therefore improve Dictionary performance in this area by eliminating the checking of subordinate subjects that are not to be deleted.

For example, when a PSB and its associated PCBs use segments and data bases that are not being deleted, then improved performance will be obtained by using a structure-type that does not contain PCB/WITH/SEGMENT or PCB/WITH/DATABASE relationship-types. The improved performance is realized because DELETE_STRUCTURE will not consider the related segment or data base subjects as candidates for deletion.

• All four structure processing functions read the entire structure-type definition into storage during their initialization phase. Because the IBM-supplied structure-type (named DEFAULT) is large, you may want to use a special structure-type for your installation.

The DEFAULT structure-type contains 66 relationship-types and consequently, 66 DL/I calls are required to retrieve and use it. A structure-type to be used for processing a PSB and its PCBs needs to contain only one relationship-type and therefore, only one DL/I call is needed to access the structure-type definition.

Manipulation of the four extensibility control information categories (CATEGORY, RELTYPE, ATTRTYPE, and STRTYPE) may be a task reserved only for the Dictionary administrator at your installation. If so, you may wish to define a special structure-type for that task and remove the three extensibility control information relationship-types from the DEFAULT structure-type.

You could define a STRTYPE subject, (P,ECl,0), as follows:

ADD STRTYPE (P,ECl,0) DATE=101082 + 
DESC=(1,'STRUCTURE-TYPE FOR COPYING AND + 
REPORTING ECI CATEGORIES') ;
AR STRTYPE (P,ECl,0) CONTAINS + 
RELTYPE (*,CATEGORY/HAS/ATTRTYPE,0) SEQ=10
AR STRTYPE (P,ECl,0) CONTAINS + 
RELTYPE (*,RELTYPE/HAS/ATTRTYPE,0) SEQ=20
AR STRTYPE (P,ECl,0) CONTAINS + 
RELTYPE (*,STRTYPE/CONTAINS/RELTYPE,0) SEQ=30 + 
SUPERCAT=L FOLLOWUP=Y FOLLOWDN=Y ;
INSTALL STRTYPE (P,ECl,0) UPDATE=YES DEST=L ;

You would use the name ECI as the value for the STRUCTURE operand when using STRUCTURE_REPORT, COPY, DELETE_STRUCTURE, and EXPORT commands.

If you frequently request the structure-type definition report as part of the output from structure dependent commands, using a smaller structure-type results in fewer lines of output. Accessing the structure-type is done once each time a structure processing command is executed. This one-time initialization step is most significant when processing a structure containing only a few subjects.
Chapter 12. Writing Programs to Access Dictionary Data

The purpose of this chapter is to show you:

- How to invoke a user-written program
- How to write a program that accesses Dictionary data

A Dictionary facility called Program Access allows you to write a program that can be invoked by a Dictionary user, either online or in batch mode. You can pass information to the program. The program can access the Dictionary data bases to retrieve:

- A subject’s attribute data
- Text data associated with a subject
- Data about the relationship between two subjects
- A list of the subject’s aliases
- A list of the subjects in a category
- A list of the subjects to which a subject is related
- A list of the types of relationships in which subjects in a category can participate
- A list of the relationship-types present in an installed structure-type definition

In addition, the program can use the Dictionary output facilities to send data to an online terminal, to the Dictionary printer or punch, and to a Dictionary stack.

How a User-Written Program Operates

Figure 55 on page 210 shows how a user-written program fits into the Dictionary environment. The letters in parentheses in the following discussion refer to encircled letters in Figure 55 on page 210. The user-written program resides in a load library. When a user invokes the program (A), the EXECUTE command is issued:

EXECUTE PGM=userpgm [PARM=‘parameter-string’] ;
The user can issue EXECUTE either as part of a batch input stream or from an online terminal. The EXECUTE command specifies a user-written program (userpgm) and usually includes a parameter string (parameter-string). DBCS users can enter a mixed string for the parameter string.

At this point, the Dictionary:

- Examines its address space and dynamically loads a copy of the user-written program from a load library (B)

- Removes the leading and trailing single quotation marks from the parameter string, compresses any pairs of single quotation marks into one single quotation mark, and passes the string to the user-written program (C)

- Passes control to the user-written program at its entry point

The user-written program is now in control. It can communicate with the Dictionary, via a CALL statement, to retrieve data from the Dictionary data bases and to send output to a terminal, line printer, punch, or Dictionary stack:

Figure 55. How a User-Written Program Operates in the Dictionary Environment
• For a retrieval request (D), the user-written program:
  – Defines the retrieval request with a control area (PACA).
  – Specifies a data area (PADA) to contain the retrieved data.
  – Specifies a format table (PADAFT) to position the data within the data area. When the data is retrieved, the Dictionary examines the format table (PADAFT) to determine where to put each data item in the data area (PADA).
  – Calls the Dictionary to perform the retrieval.

*Note:* PACA, PADAFT, and PADA reside in the user-written program and are accessible to the Dictionary.

• For an output request (E), the user-written program:
  – Defines the output request with a control area (PACA)
  – Places the output line in a data area (PADA)
  – Calls the Dictionary to send the data to its destination: the Dictionary’s printer, punch, Dictionary stack, or an online Dictionary user’s display screen

When the Dictionary is called by a user-written program for retrieval or output, it:

• Performs the requested function:
  – It retrieves the requested data from the Dictionary data bases and places it in the user-written program’s data area (PADA), or
  – It sends the data in the user-written program’s data area (PADA) to the specified output destination.

• Sets a return code that the user-written program can use to determine the status of the request

• Returns control to the user-written program

When the user-written program completes (F), it sets a return code and returns control to the Dictionary. The Dictionary interprets the return code set by the user-written program and takes appropriate action, thus completing the processing of the EXECUTE command.

The issuer of the EXECUTE command has received whatever output the user-written program may have sent. The Dictionary can now accept another command from an online user or can process the next command on a batch input stream.
How to Invoke a User-Written Program

You invoke a user-written program by issuing the Dictionary EXECUTE command, which names the program and may include a parameter string. A parameter string (up to 200 bytes) is used to pass information to the program (for example, the name of a subject and its category). The format of the EXECUTE command is:

EXECUTE PGM=upg [PARM='parameter-string'] ;

userpgm
Names the executable load module that contains the user-written program.

parameter-string
Communicates Dictionary subject names and other information required by the user-written program.

The parameter-string is a character string (up to 200 bytes) in a format required by the user-written program. DBCS users can enter a mixed string. If an EBCDIC single quotation mark is to appear in the string, it must be specified as two EBCDIC single quotation marks. Standard Dictionary command continuation rules apply when specifying a long parameter string.

The parameter string is passed to the user-written program with no modification, except for both removing the leading and trailing single quotation marks and reducing pairs of EBCDIC single quotation marks to a single quotation mark.

How to Write a Program to Access Dictionary Data

Before you write a program that accesses Dictionary data, you should be familiar with the structure and format of Dictionary subjects. In addition, you should know how subjects you are interested in can be related to other subjects. To familiarize yourself with these topics, you should refer to *OS/VS DB/DC Data Dictionary Applications Guide* for an overview, and to *OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference* and (for extensibility subjects) FORMAT and GUIDE reports for detailed reference information.

To avoid duplicating a Dictionary function with your program, you should be familiar with the capabilities of the REPORT, SCAN, PUNCH, DBD_OUT, DDT_OUT, PSB_OUT, STAGE_1_OUT, and STRUCTURES__OUT commands. The function you need may be provided with one or more of these commands.

Your program can be written in COBOL, PL/I, or Assembler language. Each programming language has somewhat different requirements for coding a program's entry point, issuing CALL statements, and link-editing.
When you write a program to access Dictionary data, you must correctly specify:

- The program's entry point.
- A parameter string area, if the program expects input when it is invoked.
- For Dictionary data retrieval:
  - A control area (PACA)\(^2\) to define the retrieval request
  - A data area (PADA)\(^3\) to contain the retrieved data
  - A format table (PADAFT) to locate the data within the data area (PADA)
- For output to the user:
  - A control area (PACA)\(^2\) to define the output request
  - A data area (PADA)\(^3\) to contain the output character string
- A return code before exiting, when the program completes.

When your program is coded and assembled, you must link-edit it with a Dictionary linkage module (as specified in “Link-Editing the COBOL Program” on page 277, “Link-Editing the PL/I Program” on page 281, and “Link-Editing the Assembler Language Program” on page 286).

Each of the tasks of writing a program is detailed in subsequent sections. Several tasks are specific to the type of programming language in which the program will be written (COBOL, PL/I, or Assembler language) and are described in subsequent sections. The sections below contain some facts about the user program's environment and describe the tasks of retrieval and output in detail.

The Program-with-Dictionary Interface

The Dictionary and your program communicate using a control area (PACA), a format table (PADAFT), and a data area (PADA). All three of these control blocks are in your program's virtual storage and can be accessed and updated by both your program and the Dictionary. Retrieval and output requests are initiated by CALL statements written in the source language of your program.

To assist you with correctly specifying the control area (PACA) and format table (PADAFT), a Dictionary library includes source code segments you can copy into your program that define these data areas. More details can be found in “Specifying the Program Access Data Area and Control Blocks” on page 279.

---

\(^2\) The PACA used for retrieval may, with slight modification, also be used for output.

\(^3\) The same PADA can be used for both retrieval and output with no change.
Requesting Dictionary Access

Your program retrieves Dictionary data by issuing a CALL statement with appropriate values set in the control area (PACA) and the format table (PADAFT). The control area (PACA) and format table (PADAFT) are control blocks that are built and reside in your program's virtual storage. The control area (PACA) defines the retrieval request to the Dictionary. The format table (PADAFT) specifies where the retrieved data is to be placed in the data area (PADA). The Dictionary puts the retrieved data in the data area (PADA) (another area in your program's virtual storage).

Your program can provide output by issuing a CALL statement with appropriate values set in the control area (PACA). The output text is put in the data area (PADA) prior to issuing the CALL.

After processing a CALL statement, the Dictionary sets a return code in the control area (PACA) and, following retrieval requests, in the format table (PADAFT). Your program can examine the return codes and take appropriate action.

Program Completion

When your program completes and returns control to the Dictionary, your program can set a return code for the Dictionary to examine in the parameter area (DBDWPAP). The Dictionary interprets the return code set by your program and takes appropriate action.

Access Authorization Checking

If Dictionary user access control is enabled, user programs are subject to access authorization checking. Prior to actually retrieving data for a retrieval request, the Dictionary determines if the user who invoked the program is authorized to access subjects in the category specified (or implied) by the retrieval request. The Dictionary also verifies that the subject named in the PACA for retrieval has a status code which the user of your program is allowed to access.

User-Program Restrictions

A user program should observe the following restrictions:

- It should not issue DL/I CALL statements. The CALL statements described in subsequent sections are sufficient to retrieve Dictionary data and to send output to the Dictionary printer, punch, Dictionary stack, and to the online terminal user that invokes the program.

- It should not initiate any subtasking.

- It should not issue STAE or ESTAE macros.

- It should free all virtual storage obtained by the program itself.
• When written in COBOL, it should not use dynamic calls to the Program Access facility.

Examples of Programs That Access the Dictionary

Three sample programs are included in a Dictionary library. Each shows how various parts of a typical program are coded. You can print the sample program written in COBOL, PL/I, or Assembler language for examples of many of the concepts presented in this chapter. In addition, "Appendix B.5. The COBOL Sample Program" on page 427, includes an annotated listing of the COBOL sample program.

Retrieving Dictionary Data

To retrieve Dictionary data, your program must specify:

• A CALL statement

• A control area (PACA), to define the request and identify the data (or type of data) to be retrieved

• A data area (PADA), to contain the retrieved data

• A format table (PADAFT), to specify the location of individual data items within the data area (PADA)

Prior to issuing the CALL statement, your program sets values in the control area (PACA) and obtains space for the data area (PADA) and format table (PADAFT). If your program doesn’t want all the data fields usually retrieved for the request, it can set values in the format table (PADAFT) to ignore some fields. The other fields will be placed at specified locations within the data area (PADA). For a general description of these data areas, see "Data Areas for Dictionary-User Program Communication" on page 216. For more details about what can be specified in the control area (PACA) and format table (PADAFT), see the description of the desired retrieval request.

Before the Program Access facility returns control to your program, it sets values:

• In the control area (PACA), to indicate the status of your request and (depending on the type of retrieval request) to specify the location of the next data item in a list to be retrieved

• In the format table (PADAFT), to indicate the status, type, and location of each retrieved item

• In the data area (PADA), the retrieved data

For more details about the information returned in the control area (PACA) and format table (PADAFT), see the description of the desired retrieval request.
For more details about the return codes set in the control area (PACA) and format table (PADAFT), see “Appendix B.1. Overall Status of a Program Access CALL Statement” on page 408 and “Appendix B.2. Status of Each Format Table (PADAFT) Entry” on page 412.

Sending Output Messages via the Dictionary

Your program can send an output message (that is, a character string) to the Dictionary printer or punch, or to the online user who invoked the program. To send an output message, your program must specify:

- A CALL statement
- A control area (PACA) that defines the request and specifies an output destination
- A data area (PADA) that contains the character string (that is, the output message)

Note: For a COBOL program, the CALL statement for an output request must specify a format table (PADAFT) to satisfy the CALL statement's format requirements, although this parameter is ignored by the Dictionary.

Prior to issuing the CALL statement, your program should set values in the control area (PACA) and put the output message in the data area (PADA). For more details about what can be specified in the control area (PACA) for an output request, see “User-Program Output” on page 271.

Before the Program Access facility returns control to your program, it sets a return code in the control area (PACA). For more details, see “Appendix B.1. Overall Status of a Program Access CALL Statement” on page 408.

Data Areas for Dictionary-User Program Communication

When your program issues a CALL statement to access Dictionary data, it:

- Describes the request with the control area (PACA)
- Reserves space for the retrieved data in the data area (PADA)
- Specifies, in the format table (PADAFT), information about each item of data to be retrieved into the data area (PADA) (for example, its location and length)

When the Dictionary returns control to your program following a retrieval request, it:

- Sets a return code in the control area (PACA) to describe the overall status of the request
- Sets positioning information in the control area (PACA), when the request is a list-type retrieval, to indicate the current position in the list.

- Places the retrieved data in the data area (PADA).

- Returns information about each item of retrieved data in the format table (PADAFT) (for example, a return code indicating its status).

Figure 56 illustrates the relationship among the three control blocks.

### Figure 56. How Program Access Data Areas Are Related

#### Using the Control Area (PACA)

The control area (PACA) is used by your program to specify information about a request. The PACA is used by the Dictionary to notify your program of the overall status of the request. The PACA:

- Specifies the type of retrieval (or output) request

- Names the Dictionary entities affected by the request (for example, the subject to be retrieved and its category name)

- Specifies arguments pertinent to the request (for example, a relationship keyword)

- Contains the addresses of the data area (PADA) and the format table (PADAFT)

*Note:* In COBOL, you specify the PADA and the PADAFT in the CALL statement.
Using the Data Area (PADA)

The data area (PADA) contains the retrieved Dictionary data after execution of the CALL statement. For output requests, the PADA contains the output character string supplied by the user-written program. The PADA must be large enough to contain all retrieved or output data, and must be accessible by both your program and the Dictionary.

The Dictionary will not write retrieved data outside the data area (PADA) specified by the control area (PACA). Improperly specified PADA length or address fields in the PACA, or offsets in the format table (PADAFT), can cause a protection exception or can cause portions of the Dictionary virtual storage to be overwritten. Even in this case, there is no danger of overwriting the Dictionary data bases. DBCS truncation as described in *OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference* does not occur if the length of the PADA is less than the length of the field. You should make sure when you retrieve data that you retrieve the entire length of the field.

Using the Format Table (PADAFT)

The format table (PADAFT) is composed of entries, each of which uniquely describes an item of data in the data area (PADA):

- Its attribute name (that is, keyword), left-justified.
- Its instance number, if the attribute can repeat.
- Its data type (for example, alphabetic).
- Whether it is to be retrieved or ignored. A retrieval request can specify a subset of the attributes of a subject, either by coding a format table (PADAFT) containing only a subset of the available attributes, or by flagging entries that identify unwanted attributes.
- Its location in the data area (PADA): an offset value from the start of the data area (PADA) to the attribute-value field.
- Its length.
- A return code indicating whether the data item's value has been retrieved completely, partially, or not at all.

Coding Options for the PADAFT

When you code a PADAFT for your program, you can specify it in one of three ways. You specify the type of PADAFT by coding a value in the PACA's PADAFT usage field (PAFTU) with:

- The Verify and Execute option (PAFTU=E). This option is used when your program currently contains the PADAFT ready for use with a retrieval request.
• The Build and Execute option (PAFTU=B). This option is used when your program wants the Dictionary to put a PADAFT appropriate for the type of data being retrieved into a space delegated to the PADAFT, and then retrieves the data. Your program cannot examine or modify the PADAFT until the retrieval request completes.

• The Prebuild option (PAFTU=P). This option is similar to the Build and Execute option, except that no data is retrieved. The Dictionary puts a PADAFT into the space designated for it, and your program can modify the PADAFT before using it with a retrieval request.

During a program's execution, the Build and Execute and Prebuild options are used infrequently, and establish a PADAFT appropriate to retrieve a specific type of subject or type of Dictionary data. After a PADAFT is established, the Verify and Execute option is usually used for subsequent retrieval requests.

Verify and Execute Option: Coding Your Own PADAFT

If you code your own format table (PADAFT), you need to include only entries for data items of interest to your program. The entries you code can be in any order. You can specify offset values to put the retrieved data anywhere within the data area (PADA). In this way, you can build a report or a line of output.

Build and Execute Option: Using a Master PADAFT with Retrieval Requests

You do not need to code your own format table (PADAFT) for each request. Instead, data for a subject or relationship can be retrieved based on a default or "master" PADAFT. In addition to returning the data in the data area (PADA), the Dictionary will return the corresponding master PADAFT into a data area reserved in your program for the PADAFT.

When a master PADAFT is used, the retrieved data is put in contiguous fields in the data area (PADA). To build a report or a line of output, you must first locate the desired data in the PADA, then move it into the desired position in an output PADA.

The Build and Execute option can be used, for example, with a general-purpose report program to retrieve the attribute data for a specified subject or relationship, interpret the format table (PADAFT) to determine what data has been retrieved (and how it is formatted in the data area (PADA)), and then format the data into an output report.

A master PADAFT is available (and is different) for each type of retrieval request. Each master PADAFT is a complete table with an entry describing each item of data to be retrieved.
Prebuild Option: Modifying a Master PADAFT before Retrieval

Alternatively, your program can request a copy of the appropriate master PADAFT without retrieving any data. Your program can then examine entries in the format table (PADAFT). Each entry can be modified to indicate:

- Whether the data item is to be retrieved or ignored.
- Where, in the data area (PADA), the retrieved attribute value is to reside. In this way, the master PADAFT can also be used to "scatter" data within a data area (PADA).

When the master PADAFT has been modified, it can be used with a retrieval request with the verify and execute option.
An Example of Program Access Control Block Usage

In this example, a retrieval request is issued to retrieve the relationship data for the relationship between segment PART-RECORD and element PART-NUMBER. The PACA and user-built PADAFT are initialized as shown in Figure 57 and Figure 58.

After the Dictionary processes the request, the PACA and PADAFT contain new or modified values in some fields, also shown in Figure 57 and Figure 58 on page 222.

<table>
<thead>
<tr>
<th>PACA Field</th>
<th>Contents Before CALL</th>
<th>Contents After CALL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>&lt;no change&gt;</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
<td>&lt;no change&gt;</td>
</tr>
<tr>
<td>PATYPE</td>
<td>RRA</td>
<td>&lt;no change&gt;</td>
<td>Request type: attributes of a relationship</td>
</tr>
<tr>
<td>PADAOP</td>
<td>B</td>
<td>&lt;no change&gt;</td>
<td>PADA blanking: put blanks in the PADA when the field cannot be retrieved</td>
</tr>
<tr>
<td>PAFTU</td>
<td>E</td>
<td>&lt;no change&gt;</td>
<td>PADAFT usage: Verify and Execute</td>
</tr>
<tr>
<td>PADAP</td>
<td>&lt;user-specified&gt;</td>
<td>&lt;no change&gt;</td>
<td>Address of the PADA</td>
</tr>
<tr>
<td>PADAL</td>
<td>41</td>
<td>&lt;no change&gt;</td>
<td>Length of the PADA</td>
</tr>
<tr>
<td>PAFTP</td>
<td>&lt;user-specified&gt;</td>
<td>&lt;no change&gt;</td>
<td>Address of the PADAFT</td>
</tr>
<tr>
<td>PAFTL</td>
<td>108</td>
<td>&lt;no change&gt;</td>
<td>Length of the PADAFT area</td>
</tr>
<tr>
<td>PARC</td>
<td>&lt;not specified&gt;</td>
<td>0</td>
<td>Return code for the request</td>
</tr>
<tr>
<td>PART</td>
<td>&lt;not specified&gt;</td>
<td>12</td>
<td>Highest return code in a PADAFT entry</td>
</tr>
<tr>
<td>PACAT</td>
<td>SEGMENT</td>
<td>&lt;no change&gt;</td>
<td>Principal subject category (8 bytes)</td>
</tr>
<tr>
<td>PANAME</td>
<td>TCPART-RECORD</td>
<td>&lt;no change&gt;</td>
<td>Principal subject name (36-byte name)</td>
</tr>
<tr>
<td>PARKW</td>
<td>WITH</td>
<td>&lt;no change&gt;</td>
<td>Relationship keyword (12 bytes)</td>
</tr>
<tr>
<td>PARCAT</td>
<td>FIELD</td>
<td>&lt;no change&gt;</td>
<td>Related subject category name (8 bytes)</td>
</tr>
<tr>
<td>PARNAME</td>
<td>TCPART-NUMBER</td>
<td>&lt;no change&gt;</td>
<td>Name of related subject (36-byte name)</td>
</tr>
<tr>
<td>PARSEQ</td>
<td>1</td>
<td>&lt;no change&gt;</td>
<td>Value of the sequence attribute (8 bytes)</td>
</tr>
</tbody>
</table>

Figure 57. Example of PACA before and after a Retrieval Request
<table>
<thead>
<tr>
<th>PADAFT Field</th>
<th>Contents Before CALL</th>
<th>Contents After CALL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAFTID</td>
<td>PAFT</td>
<td>&lt;no change&gt;</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PAFTFM</td>
<td>A</td>
<td>B</td>
<td>&lt;no change&gt;</td>
</tr>
<tr>
<td>PAFTENTS</td>
<td>4</td>
<td>&lt;no change&gt;</td>
<td>Number of entries in the PADAFT</td>
</tr>
<tr>
<td>PAFTENTL</td>
<td>24</td>
<td>&lt;no change&gt;</td>
<td>Length of each entry</td>
</tr>
</tbody>
</table>

**Entry 1:**

| PAFNAME      | START                | <no change>         | Keyword identifying the START attribute |
| PAFINST      | 0                    | <no change>         | Occurrence number (zero for nonrepeating attributes) |
| PAFDTYP      | <not specified>      | Z                   | Data type: numeric characters |
| PAFFUNC      | R                    | <no change>         | Retrieval function: retrieve this attribute |
| PAFDATA      | 6                    | <no change>         | Where to put the data in the PADA (offset from start of PADA) |
| PAFLENG      | 5                    | <no change>         | Length of the retrieved data |
| PAFRC        | <not specified>      | 0                   | Status of the retrieval request for the item: OK retrieval |

**Entry 2:**

| PAFNAME      | BITSTART             | <no change>         | Keyword identifying the BITSTART attribute |
| PAFINST      | 0                    | <no change>         | Occurrence number (zero for nonrepeating attributes) |
| PAFDTYP      | <not specified>      | Z                   | Data type: numeric characters |
| PAFFUNC      | R                    | <no change>         | Retrieval function: retrieve this attribute |
| PAFDATA      | 21                   | <no change>         | Where to put the data in the PADA (offset from start of PADA) |
| PAFLENG      | 1                    | <no change>         | Length of the retrieved data |

Figure 58 (Part 1 of 2). Example of PADAFT before and after a Retrieval Request
<table>
<thead>
<tr>
<th><strong>PADAFT Field</strong></th>
<th><strong>Contents Before CALL</strong></th>
<th><strong>Contents After CALL</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>PAFRC</td>
<td>&lt;not specified&gt;</td>
<td>12</td>
<td>Status of the retrieval request for the item: data item unavailable</td>
</tr>
<tr>
<td><strong>Entry 3:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAFNAME</td>
<td>GENFLD</td>
<td>&lt;no change&gt;</td>
<td>Keyword identifying the GENFLD attribute</td>
</tr>
<tr>
<td>PAFINST</td>
<td>0</td>
<td>&lt;no change&gt;</td>
<td>Occurrence number (zero for nonrepeating attributes)</td>
</tr>
<tr>
<td>PAFDTYP</td>
<td>&lt;not specified&gt;</td>
<td>C</td>
<td>Data type: numeric characters</td>
</tr>
<tr>
<td>PAFFUNC</td>
<td>R</td>
<td>&lt;no change&gt;</td>
<td>Retrieval function: retrieve this attribute</td>
</tr>
<tr>
<td>PAFDATA</td>
<td>30</td>
<td>&lt;no change&gt;</td>
<td>Where to put the data in the PADA (offset from start of PADA):¹</td>
</tr>
<tr>
<td>PAFLENG</td>
<td>1</td>
<td>&lt;no change&gt;</td>
<td>Length of the retrieved data</td>
</tr>
<tr>
<td>PAFRC</td>
<td>&lt;not specified&gt;</td>
<td>0</td>
<td>Status of the retrieval request for the item: OK retrieval</td>
</tr>
<tr>
<td><strong>Entry 4:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAFNAME</td>
<td>FIRSTLEV</td>
<td>&lt;no change&gt;</td>
<td>Keyword identifying the FIRSTLEV attribute</td>
</tr>
<tr>
<td>PAFINST</td>
<td>0</td>
<td>&lt;no change&gt;</td>
<td>Occurrence number (blank for nonrepeating attributes)</td>
</tr>
<tr>
<td>PAFDTYP</td>
<td>&lt;not specified&gt;</td>
<td>C</td>
<td>Data type: numeric characters</td>
</tr>
<tr>
<td>PAFFUNC</td>
<td>R</td>
<td>&lt;no change&gt;</td>
<td>Retrieval function: retrieve this attribute</td>
</tr>
<tr>
<td>PAFDATA</td>
<td>41</td>
<td>&lt;no change&gt;</td>
<td>Where to put the data in the PADA (offset from start of PADA):¹</td>
</tr>
<tr>
<td>PAFLENG</td>
<td>1</td>
<td>&lt;no change&gt;</td>
<td>Length of the retrieved data</td>
</tr>
<tr>
<td>PAFRC</td>
<td>&lt;not specified&gt;</td>
<td>12</td>
<td>Status of the retrieval request for the item: data item unavailable</td>
</tr>
</tbody>
</table>

Figure 58 (Part 2 of 2). Example of PADAFT before and after a Retrieval Request

*Note to Figure 58:*

¹ For list type retrievals, this value indicates an offset from the start of the retrieved instance in the PADA.
In this example, the PADA was initialized with the text of an output message (in this case, a line of a report), with space left to insert the retrieved data:

PADA before the retrieval request:
6  21  30  41
|   |   |   |   |
V  V  V  V
'START=  BITSTART=  SEQIND=  FIRSTLEV= '

When the data is retrieved, the PADA contains the data in positions specified by the PADAFT:

PADA after the retrieval request:
6  21  30  41
|   |   |   |   |
V  V  V  V
'START=1  BITSTART=0  SEQIND=U  FIRSTLEV= '

Retrieving Data

Data retrieved by the Dictionary is returned to your program as a character string, and is in one of several formats. The Dictionary identifies the format of the corresponding data with a code in the PAFDTYP field of each PADAFT entry. “Types of Attributes” in “Appendix B.8. Special Topics for Attribute Retrieval” on page 462 lists the format codes and describes each format.

The Dictionary provides two types of data retrieval: attribute data retrieval and list-type data retrieval. Each type is described in the following sections.

Retrieving Attributes of Subjects and Relationships

You can retrieve data about the attributes of a specific subject or relationship. This type of retrieval is called “attribute data retrieval.” The attribute data retrieval request types are:

- Retrieving a subject’s attribute data
- Retrieving a relationship’s attribute data

Figure 57 on page 221 and Figure 58 on page 222 show examples of attribute retrieval. In these examples, the START, BITSTART, GENFLD, and FIRSTLEV attributes of a Segment-WITH-Element relationship are retrieved.

The values of the PACA and PADAFT fields differ from request to request, depending on the type of request and the subject or relationship to be retrieved. Details on what should be in the PACA are provided in “Retrieving a Subject’s Attribute Data (RSA Call)” on page 232 and “Retrieving a Relationship’s Attribute Data (RRA Call)” on page 235.

The PADAFT used in a retrieval request contains an entry for each attribute to be retrieved. “Appendix B.6. PADAFT Entries for Dictionary Categories” on
page 437 describes the master PADAFT for each type of attribute retrieval request for standard categories and relationship-types.

During attribute retrieval, no more than one data item is retrieved for each entry in the PADAFT. This feature, along with the type of data that can be retrieved, differentiates attribute data retrieval from list-type data retrieval.

Some categories and relationship-types have attributes that require special handling. To assist you, “Appendix B.8. Special Topics for Attribute Retrieval” on page 462 contains:

- A description of repeating attributes, groups of repeating attributes, and the PADAFT's used to retrieve them

- Information on how to format a sequence value in the PACA when retrieving the attributes of a Segment-WITH-Element or Element-CONTAINS-Element relationship

### Retrieving Lists of Data

Some types of retrieval requests obtain multiple occurrences of the same type of data from different subjects or relationships. These are called "list-type" retrieval requests. For example, a request to retrieve lines of text data is a list-type retrieval request.

Another example of a list-type retrieval request is the retrieval of a list of the names of all subjects related to a specified subject. Each element of the list completely describes one relationship, where the relationship is represented by a set of data items: the related category name, related subject name, relationship keyword, and other attributes of the relationship.

The list-type retrieval requests are:

- Retrieving text data

- Retrieving a subject's aliases

- Retrieving subject names in a category

- Retrieving relationships for a subject

- Retrieving the names of relationship-types allowed for a category

- Retrieving the relationship-types which are included as an installed structure-type definition

If you are operating under Release 6 of the Dictionary (or any subsequent release), you can selectively retrieve a list of subject names based on their status by specifying a valid status code (A-T, 0-9, *) for the PACA field PSTATUS. Use of the PSTATUS field is valid only when PACAFM=B.
The Format Table (PADAFT) for a List-Type Request

The PADAFT for a list-type request describes a single instance of the list to be retrieved. Each PADAFT entry describes one item in the set of data items that comprise the instance. The set of PADAFT entries (that is, the PADAFT used to retrieve all data items of the instance) is called the master PADAFT. For example, a single line of user data text is described with a master PADAFT consisting of just two PADAFT entries: one for its line number and the other for its text contents.

Your program tells the Dictionary (with a value set in the PACA’s PANREQ field) how many instances to retrieve from the list. The number of instances actually retrieved is set in the PACA’s PANRETR field. If the end of the list is reached before the number of instances you requested has been retrieved, the Dictionary sets an “end-of-data” return code. The master PADAFT is reused as many times as necessary in a single request to retrieve the number of instances requested in the PANREQ field.

Master PADAFTs for list-type retrieval requests are sometimes called special master PADAFTs, because the “attributes” they describe are not unique for a Dictionary subject. The format of each special master PADAFT is included in the description of the list-type retrieval request in this chapter.

The Control Area (PACA) for a List-Type Request

When you request a list-type retrieval, you may want retrieval to begin at some position within the list other than the first instance. The PACA contains positioning parameters that you can set to control this. If the first instance is wanted, these parameters must be set to blanks. If other than the first instance is to be retrieved, then one or more of the following fields in the PACA can be set by your program to identify the instance at which retrieval should start:

- **PAPLINE** - Line number
- **PAPNAME** - Subject name
- **PAPRKW** - Relationship keyword
- **PAPRCAT** - Related category
- **PAPRNAME** - Related subject name
- **PAPRSEQ** - Relationship sequence value

For example, if your program wants to retrieve a list of subjects in a category starting with a particular subject, your program puts the name of the subject into the PAPNAME field. If your program retrieves text data, it can specify a starting line number and retrieve all lines of text following it.

The PAACCESS field, another positioning parameter, can be used with the above parameters to further identify where retrieval should start. The PAACCESS value can indicate:
• FIRST or LAST: Retrieval is to start (or resume) with the first or last instance of the list; other positioning data is ignored.

• HERE: Retrieval is to start or resume with the instance specified in the positioning data.

• NEXT: Retrieval is to start or resume with the next instance after the one specified in the positioning data.

• DOWN: Retrieval is to start a certain number of items “down” the list from the positioning instance. The “down-count” is specified in the PADOWN field. However, if the down-count exceeds the number of instances remaining in the list, the last instance is returned to your program in the PADA, its location is set in the PACA positioning fields, and the return code (PARC) is set to PACLAST or POSITIONED-ON-LAST.

Positioning within a List

Positioning parameters can be specified by your program on an initial request, as stated above. However, the same set of positioning parameters (set by the Dictionary) is usually used on the second and subsequent retrieval requests, after you have retrieved several lines and want to resume retrieval with the line following the last one retrieved. Consequently, a small data area (PADA) can be used with repeated retrieval requests to obtain the complete set of available data.

When a part of a list of data is retrieved and more remains to be retrieved, the Dictionary sets values in positioning fields. The positioning fields contain values that locate the latest retrieved data and PAACCESS contains the “NEXT” option: A subsequent retrieval request can begin with the next available data.

In most cases, the instance located by the positioning information must exist for the retrieval to complete successfully. However, when the “NEXT” option of the PAACCESS field is used with a text-data retrieval request, retrieval begins with the first available text line (that is, instance) that follows the line previously retrieved (located with the positioning data). For example, suppose your program just retrieved text line 99 and it now requests retrieval of the next text line. Because (in this example) lines 100 through 106 are not defined, the Dictionary returns line 107 to your program.

Stepping Through a List: Your program can “step through” a list using only the “NEXT” option. The first retrieval request (you specify blanks in the positioning fields) retrieves the first instance of the list. Because the positioning data (including the PAACCESS field) is set by the Dictionary, a subsequent request using the same control area (PACA) will retrieve the next instance of the list (that is, the one immediately following the last instance retrieved). Retrieval can continue in this manner until all instances of the list are retrieved (indicated by either an end-of-data return code or a data-not-found return code, if the previously retrieved instance was the last instance of the list).

Note: When an end-of-data condition occurs, the positioning fields are set to blanks.
Locating Instances of the List in the Data Area (PADA)

Your program's control area (PACA) should specify an increment value for each list-type retrieval request. That is, the number of bytes from the start of one instance in the data area (PADA) to the start of the next should be included. When several instances are retrieved with one request, the items of each instance are returned in the PADA at a position determined by adding the increment value to the position of the corresponding item of the previous instance.

The Dictionary sets the PACA field PANRETR to indicate the number of instances actually retrieved. This value can be used as a loop control value when processing all instances in the list.

Retrieval Requests

This section describes each type of retrieval request:

- Retrieving a subject's attribute data (RSA)
- Retrieving a relationship's attribute data (RRA)
- Retrieving text data:
  - Description text (RDSC)
  - PL/I data text (RPLI)
  - User data text (RUD1, RUD2, RUD3, RUD4, and RUD5)
- Retrieving a subject's aliases (RA)
- Retrieving subject names in a category (RSN and RSAN)
- Retrieving relationships for a subject:
  - All subjects related to a specified subject (RSR)
  - All subjects within a specified category related to a specified subject (RSRC)
  - All subjects within a specified category and related to the specified subject with a specified keyword (RSRK)
- Retrieving a list of relationship-types allowed for a category (RCR)
- Retrieving a list of relationship-types defined for an installed structure-type definition (RSTR)

Each request type is identified to the Dictionary by placing a 4-byte code (for example, 'RCR ') in the PATYPE field of the PACA. DBCS users can use each of these retrieval requests to retrieve data containing DBCS strings.
If a PADAFT is supplied with the retrieval request, each entry in that PADAFT must contain the attribute keyword, an instance number (if the attribute repeats), and the position and length of the data to be put in the PADA. Appropriate entries are available in a master PADAFT, which identifies all the attribute keyword names and associated specifications that apply to a particular request. The position and length specifications in the master PADAFTs reflect the minimum spacing requirements for retrieving the data without overlapping fields and with no extraneous space between fields.

Some of the retrieval request types reuse the data descriptions in the PADAFT to obtain multiple occurrences of the same collection of data. These are termed "list-type" retrieval requests and are described in "Retrieving Lists of Data" on page 225.

The attribute keyword names, data types, PADA offset, and the length of the value can be obtained for any retrieval request from the following sources:

- For standard categories, control information categories, and standard and control information relationship-types, refer to "Appendix B.6. PADAFT Entries for Dictionary Categories" on page 437.

- For extensibility categories, relationship-types, and structure-types, the information should be supplied by the developer of that definition, preferably as data contained in a GUIDE report.

- For any category or relationship-type, you can also refer to Dictionary GUIDE and FORMAT reports.

- For list-type retrieval requests, details about the master PADAFT are with the retrieval description.

- For any retrieval request, your program can refer to a returned master PADAFT using the "build and execute" or "prebuild" option.

Retrieving Information about Fast Path Subjects

New PACA and PADAFT format codes for Dictionary Release 6 indicate that the format of the PAFINST field is binary. This allows for the retrieval of all 240 possible DATASET groups defined in the master PADAFT. The format code for the new binary format is 'B'. Any PA program compiled or recompiled after update to Release 6 must use a numeric PAFINST field rather than a character PAFINST field.

You may recompile existing PA programs to take advantage of the increased limit for DATASET instance numbers; however, when doing so, make sure that the PACAFM and PAFTFM fields are changed to 'B'. You do not have to recompile existing PA programs to use them with Release 6. However, the Dictionary will support the 'A' format for only an interim period. The Dictionary performs the necessary conversions from the 'A' to the 'B' formats.
Combining Several Types of Retrieval

Your program might issue several types of retrieval requests to achieve a desired result. For example:

- The program can generate reports on a subset of a category's subjects. First, the program retrieves a list of all subjects in the category (with an RSN request). Next, the program selects subject names from the list and retrieves each subject's attribute data (with an RSA request).

- The program can "navigate" through a network of interrelated subjects with the following procedure:
  1. List the related subjects for a specified subject (with an RSR request).
  2. Select a subject from the related-subjects list.
  3. List the subjects related to the related subject (with an RSR request).

The program can obtain further information for each subject selected from the related-subject list:
- Obtain the attributes of the relationship between the specified subject and its related subject (with an RRA request).
- Obtain the attributes of the related subject (with an RSA request).

- Using the Dictionary description of a DL/I data base and its segments, the program can find a SEGMENT subject's parent (with an RRA request) to get the attributes of the SEGMENT-WITH-DATABASE relationship, convert the retrieved parent name from a 13-byte (D-format) name to a 36-byte (I-format) name, and use the I-format name to obtain attributes of the parent segment (with an RSA request) or relationship data of the relationship between the segment and its data base (with an RSR request).
How Each Retrieval Request Is Described in This Section

Each retrieval request description has the following format:

- **Type of Retrieval:**
  
  A description of what is being retrieved.

- **Required Control-Area Fields:**
  
  A table showing the control area (PACA) fields required for the retrieval and, when appropriate, values for required fields.
  
  - Possible values for a field are shown separated by an “or” sign (|).
  
  - When a nonspecific value must be supplied, it is indicated with “---.”

- **Master Format Table:**
  
  For *list-type* retrievals, a table showing the master PADAFT contents for the retrieval\(^4\).

- **Positioning Fields:**
  
  For *list-type* retrievals, a table showing those fields that can be used to describe a starting position in the list at which the retrieval is to begin.

- **Returned Control-Area Data:**
  
  A table showing the control area (PACA) fields that contain information set by the Dictionary after executing the CALL statement.

- **Return Code Table:**
  
  A table showing, for the type of retrieval, what codes can be returned by the Dictionary in the PARC field of the control area (PACA).

More detailed information about the PACA and PADAFT control blocks can be found in:

- “Appendix B.3. The Control Area (PACA)” on page 414 and
- “Appendix B.4. The Format Table (PADAFT)” on page 424.

---

\(^4\) For attribute-type retrieval requests, the master PADAFT contents are listed in “Appendix B.6. PADAFT Entries for Dictionary Categories” on page 437.
Retrieving a Subject's Attribute Data (RSA Call)

In a retrieval operation to obtain attribute data, all (or some subset of) the attribute data associated with a single specified subject is returned to your program in the data area (PADA). Your program must supply the following control area (PACA) fields shown in Figure 59.

**Required Control-Area Fields**

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (if any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>PATYPE</td>
<td>RSA</td>
<td>Request type: attributes of a subject</td>
</tr>
<tr>
<td>PADAOP</td>
<td>B</td>
<td>N</td>
</tr>
<tr>
<td>PAFTU</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>PADAP</td>
<td>---</td>
<td>Address of the PADA</td>
</tr>
<tr>
<td>PADAL</td>
<td>---</td>
<td>Length of the PADA</td>
</tr>
<tr>
<td>PAFTP</td>
<td>---</td>
<td>Address of the PADAFT</td>
</tr>
<tr>
<td>PAFTL</td>
<td>---</td>
<td>Length of the PADAFT</td>
</tr>
<tr>
<td>PANAME</td>
<td>---</td>
<td>Name of the Dictionary subject</td>
</tr>
<tr>
<td>PACAT</td>
<td>---</td>
<td>Name of the subject's category</td>
</tr>
</tbody>
</table>

*Figure 59. Required PACA Fields for an RSA Request*
Returned Control-Area Data

When the Dictionary returns control to your program after processing the RSA request, the control area (PACA) fields shown in Figure 60 contain information set by the Dictionary.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAFTLM</td>
<td>Length of the master PADAFT</td>
</tr>
<tr>
<td>PARC</td>
<td>Overall return code for the request</td>
</tr>
<tr>
<td>PART</td>
<td>Highest of the return codes set in the PADAFT entries (PAFRC)</td>
</tr>
</tbody>
</table>

Figure 60. PACA Fields Set by the Dictionary on Return from an RSA Request
Return Code Table

One of the return codes in Figure 61, is set by the Dictionary. "Appendix B.1. Overall Status of a Program Access CALL Statement" on page 408 describes each return code in detail.

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PACOK</td>
<td>Successful retrieval</td>
</tr>
<tr>
<td>16</td>
<td>PACDATA</td>
<td>No data specified</td>
</tr>
<tr>
<td>20</td>
<td>PACAUTH</td>
<td>User not authorized</td>
</tr>
<tr>
<td>32</td>
<td>PACNAME</td>
<td>No such subject</td>
</tr>
<tr>
<td>60</td>
<td>PACCAT</td>
<td>Bad primary category</td>
</tr>
<tr>
<td>64</td>
<td>PACFTU</td>
<td>Bad primary usage value</td>
</tr>
<tr>
<td>68</td>
<td>PACFTO</td>
<td>PADAFT area overflow</td>
</tr>
<tr>
<td>72</td>
<td>PACFT</td>
<td>Bad PADAFT specification</td>
</tr>
<tr>
<td>76</td>
<td>PACDAO</td>
<td>Bad PADA operation</td>
</tr>
<tr>
<td>80</td>
<td>PACDAO</td>
<td>Data area overflow</td>
</tr>
<tr>
<td>84</td>
<td>PACDA</td>
<td>Bad PADA specification</td>
</tr>
<tr>
<td>92</td>
<td>PACTYPE</td>
<td>Bad request type</td>
</tr>
<tr>
<td>96</td>
<td>PACDLI</td>
<td>DL/I error</td>
</tr>
<tr>
<td>100</td>
<td>PACAINV</td>
<td>Bad PACA specification</td>
</tr>
<tr>
<td>104</td>
<td>PACERR</td>
<td>Unexpected condition</td>
</tr>
</tbody>
</table>

Figure 61. Summary of Return Codes for an RSA Retrieval Request
Retrieving a Relationship's Attribute Data (RRA Call)

When retrieving relationship data, all (or some subset of) the attribute data associated with a single instance of a Dictionary relationship is returned to your program in the data area (PADA). Your program must supply the control area (PACA) fields shown in Figure 62.

Required Control-Area Fields

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (if any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>PATYPE</td>
<td>RRA</td>
<td>Request type: attributes of a relationship</td>
</tr>
<tr>
<td>PADAOP</td>
<td>B</td>
<td>N</td>
</tr>
<tr>
<td>PAFTU</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>PADAP</td>
<td>---</td>
<td>Address of the PADA</td>
</tr>
<tr>
<td>PADAL</td>
<td>---</td>
<td>Length of the PADA</td>
</tr>
<tr>
<td>PAFTP</td>
<td>---</td>
<td>Address of the PADAFT</td>
</tr>
<tr>
<td>PAFTL</td>
<td>---</td>
<td>Address of the PADAFT</td>
</tr>
<tr>
<td>PANAME</td>
<td>---</td>
<td>Name of the Dictionary subject</td>
</tr>
<tr>
<td>PACAT</td>
<td>---</td>
<td>Name of the subject's category</td>
</tr>
<tr>
<td>PARNAME</td>
<td>---</td>
<td>Name of the related Dictionary subject</td>
</tr>
<tr>
<td>PARCAT</td>
<td>---</td>
<td>Name of the related subject's category</td>
</tr>
<tr>
<td>PARKW</td>
<td>---</td>
<td>Keyword identifying the relationship-type</td>
</tr>
<tr>
<td>PARSEQ</td>
<td>---</td>
<td>The value of the sequence attribute identifying the relationship instance, or a blank if instances of the relationship are not sequenced. See “Appendix B.8. Special Topics for Attribute Retrieval” on page 462 for additional format considerations.</td>
</tr>
</tbody>
</table>

Figure 62. Required PACA Fields for an RRA Request
Returned Control-Area Data

When the Dictionary returns control to your program after processing the RRA request, the control area (PACA) fields shown in Figure 63 contain information set by the Dictionary.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAFTLM</td>
<td>Length of the master PADAFT</td>
</tr>
<tr>
<td>PARC</td>
<td>Overall return code for the request</td>
</tr>
<tr>
<td>PART</td>
<td>Highest of the return codes set in the PADAFT entries (PAFRC)</td>
</tr>
</tbody>
</table>

Figure 63. PACA Fields Set by the Dictionary on Return from an RRA Request
Return Code Table

One of the return codes shown in Figure 64, is set by the Dictionary. "Appendix B.1. Overall Status of a Program Access CALL Statement" on page 408 describes each return code in detail.

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PACOK</td>
<td>Successful retrieval</td>
</tr>
<tr>
<td>16</td>
<td>PACDATA</td>
<td>No data specified</td>
</tr>
<tr>
<td>20</td>
<td>PACAUTH</td>
<td>User not authorized</td>
</tr>
<tr>
<td>28</td>
<td>PACREL</td>
<td>No such relationship</td>
</tr>
<tr>
<td>32</td>
<td>PACNAME</td>
<td>No such subject</td>
</tr>
<tr>
<td>52</td>
<td>PACRKW</td>
<td>Bad relationship keyword</td>
</tr>
<tr>
<td>54</td>
<td>PACRNAME</td>
<td>No such related subject</td>
</tr>
<tr>
<td>56</td>
<td>PACRCAT</td>
<td>Bad related category</td>
</tr>
<tr>
<td>60</td>
<td>PACCAT</td>
<td>Bad primary category</td>
</tr>
<tr>
<td>64</td>
<td>PACFTU</td>
<td>Bad primary usage value</td>
</tr>
<tr>
<td>68</td>
<td>PACFTO</td>
<td>PADAFT area overflow</td>
</tr>
<tr>
<td>72</td>
<td>PACFT</td>
<td>Bad PADAFT specification</td>
</tr>
<tr>
<td>76</td>
<td>PACDAO</td>
<td>Bad PADA operation</td>
</tr>
<tr>
<td>80</td>
<td>PACDAO</td>
<td>Data area overflow</td>
</tr>
<tr>
<td>92</td>
<td>PACTYPE</td>
<td>Bad request type</td>
</tr>
<tr>
<td>96</td>
<td>PACDLI</td>
<td>DL/I error</td>
</tr>
<tr>
<td>100</td>
<td>PACAINV</td>
<td>Bad PACA specification</td>
</tr>
<tr>
<td>104</td>
<td>PACERR</td>
<td>Unexpected condition</td>
</tr>
</tbody>
</table>

Figure 64. Summary of Return Codes for an RRA Retrieval Request
Retrieving Text Data (RDSC, RUDn, RPLI Calls)

Several retrieval request types allow your program to retrieve the different sets of "text" data (that is, Description, User Data, or PL/I Data) associated with an individual Dictionary subject. Up to 999 lines of Description or User Data text can be retrieved with a single request, provided there is space in the data area (PADA). Additionally, SEGMENT and ELEMENT subjects can have up to 255 lines of PL/I data text associated with them.

When requesting a retrieval of text data, your program must supply the PACA fields shown in Figure 65.

**Required Control-Area Fields**

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (if any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>PATYPE</td>
<td>RDSC</td>
<td>RUD1</td>
</tr>
<tr>
<td>PADAOP</td>
<td>B</td>
<td>N</td>
</tr>
<tr>
<td>PAFTU</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>PADAP</td>
<td>---</td>
<td>Address of the PADA</td>
</tr>
<tr>
<td>PADAL</td>
<td>---</td>
<td>Length of the PADA</td>
</tr>
<tr>
<td>PAFTP</td>
<td>---</td>
<td>Address of the PADAFT</td>
</tr>
<tr>
<td>PAFTL</td>
<td>---</td>
<td>Address of the PADAFT</td>
</tr>
<tr>
<td>PANAME</td>
<td>---</td>
<td>Name of the Dictionary subject whose text data is to be retrieved</td>
</tr>
<tr>
<td>PACAT</td>
<td>---</td>
<td>Name of the subject's category</td>
</tr>
<tr>
<td>PAACCESS</td>
<td>F</td>
<td>L</td>
</tr>
</tbody>
</table>

*Figure 65 (Part 1 of 2). Required PACA Fields for a Text Data Request*
<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (if any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPLINE</td>
<td>---</td>
<td>The positioning line number. Default: the number of the first available line.</td>
</tr>
<tr>
<td>PANREQ</td>
<td>---</td>
<td>The number of text lines to be retrieved.</td>
</tr>
<tr>
<td>PAINCR</td>
<td>---</td>
<td>The PADA position increment for multiple lines of text data: the number of bytes from the start of one text line to the start of the next.</td>
</tr>
</tbody>
</table>

Figure 65 (Part 2 of 2). Required PACA Fields for a Text Data Request

Master Format Table

All text data retrieval requests are list-type retrievals. The formats in which the different types of text data are written into the data area (PADA) under the control of their respective master format tables (PADAFTs) are outlined below.

The format table (PADAFT) can be modified to return only the text portion of the text line by omitting the other items in the format table (PADAFT) or by setting the other item's PAFFUNC field to "N" (that is, no retrieval).

Description Data

Each line of description data occupies 75 bytes in the PADA. Two PADAFT entries are required to retrieve a line of description data. However, the pair of entries is reused for each line retrieved. The master PADAFT entries for attributes of a line of Description Data are:

<table>
<thead>
<tr>
<th>Keyword (PAFNAME)</th>
<th>Instance (PAFINST)</th>
<th>Type (PAFDTYP)</th>
<th>Offset (PAFDATA)</th>
<th>Length (PAFLENG)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEXTSEQ</td>
<td>0</td>
<td>Z</td>
<td>0</td>
<td>3</td>
<td>Text line number</td>
</tr>
<tr>
<td>TEXT</td>
<td>0</td>
<td>C</td>
<td>3</td>
<td>72</td>
<td>Description text</td>
</tr>
</tbody>
</table>
User Data Text

Each line of User Data text occupies 83 bytes in the PADA. Two PADAFT entries are required to retrieve a line of description data. However, the pair of entries is reused for each line retrieved. The master PADAFT entries for attributes of a line of User Data text are:

<table>
<thead>
<tr>
<th>Keyword (PAFNAME)</th>
<th>Instance (PAFINST)</th>
<th>Type (PAFDTYPE)</th>
<th>Offset (PAFDATA)</th>
<th>Length (PAFLENG)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINENO</td>
<td>0</td>
<td>Z</td>
<td>0</td>
<td>3</td>
<td>Text line number</td>
</tr>
<tr>
<td>TEXT</td>
<td>0</td>
<td>C</td>
<td>3</td>
<td>80</td>
<td>User data text</td>
</tr>
</tbody>
</table>

PL/I Data Text

Each line of PL/I data text occupies 76 bytes in the PADA. Three PADAFT entries are required to retrieve a line of description data. However, the set of entries is reused for each line retrieved. The master PADAFT entries for attributes of a line of PL/I Data text are:

<table>
<thead>
<tr>
<th>Keyword (PAFNAME)</th>
<th>Instance (PAFINST)</th>
<th>Type (PAFDTYPE)</th>
<th>Offset (PAFDATA)</th>
<th>Length (PAFLENG)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLISEQ</td>
<td>0</td>
<td>Z</td>
<td>0</td>
<td>3</td>
<td>Text line number</td>
</tr>
<tr>
<td>PLITYPE</td>
<td>0</td>
<td>C</td>
<td>3</td>
<td>1</td>
<td>Type of PL/I data</td>
</tr>
<tr>
<td>PLIDATA</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>72</td>
<td>PL/I data text</td>
</tr>
</tbody>
</table>

Positioning Fields

The last line of text to be retrieved can be controlled by specifying the PAPLINE field described in Figure 66.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPLINE</td>
<td>Positioning: line number of the last text line retrieved</td>
</tr>
</tbody>
</table>

Figure 66. Optional Field to Control Last Text Line Retrieved
Returned Control-Area Data

When the Dictionary returns control to your program after processing the text data retrieval request, the control area (PACA) fields shown in Figure 67 contain information set by the Dictionary.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PANRETR</td>
<td>Number of text lines retrieved</td>
</tr>
<tr>
<td>PAFTLM</td>
<td>Length of the master PADAFT</td>
</tr>
<tr>
<td>PARC</td>
<td>Overall return code for the request</td>
</tr>
<tr>
<td>PART</td>
<td>Highest of the return codes set in the PADAFT entries (PAFRC)</td>
</tr>
</tbody>
</table>

Figure 67. PACA Fields Set by the Dictionary on Return from a Text Data Retrieval
### Return Code Table

One of the return codes shown in Figure 68 is set by the Dictionary. “Appendix B.1. Overall Status of a Program Access CALL Statement” on page 408, describes each return code in detail.

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PACOK</td>
<td>Successful retrieval</td>
</tr>
<tr>
<td>4</td>
<td>PACEOD</td>
<td>End of data</td>
</tr>
<tr>
<td>6</td>
<td>PACLAST</td>
<td>Positioned on last</td>
</tr>
<tr>
<td>8</td>
<td>PACNF</td>
<td>No data found</td>
</tr>
<tr>
<td>12</td>
<td>PACPNF</td>
<td>Position not found</td>
</tr>
<tr>
<td>16</td>
<td>PACDATA</td>
<td>No data specified</td>
</tr>
<tr>
<td>20</td>
<td>PACAUTH</td>
<td>User not authorized</td>
</tr>
<tr>
<td>32</td>
<td>PACNAME</td>
<td>No such subject</td>
</tr>
<tr>
<td>36</td>
<td>PACDOWN</td>
<td>Bad down count</td>
</tr>
<tr>
<td>40</td>
<td>PACNREQ</td>
<td>Bad number of items</td>
</tr>
<tr>
<td>44</td>
<td>PACINCR</td>
<td>Bad increment</td>
</tr>
<tr>
<td>50</td>
<td>PACPLINE</td>
<td>Bad line number</td>
</tr>
<tr>
<td>60</td>
<td>PACCAT</td>
<td>Bad primary category</td>
</tr>
<tr>
<td>64</td>
<td>PACFTU</td>
<td>Bad primary usage value</td>
</tr>
<tr>
<td>68</td>
<td>PACFTO</td>
<td>PADAFT area overflow</td>
</tr>
<tr>
<td>72</td>
<td>PACFT</td>
<td>Bad PADAFT specification</td>
</tr>
<tr>
<td>76</td>
<td>PACDAOP</td>
<td>Bad PADA operation</td>
</tr>
<tr>
<td>80</td>
<td>PACDAO</td>
<td>Data area overflow</td>
</tr>
<tr>
<td>84</td>
<td>PACDA</td>
<td>Bad PADA specification</td>
</tr>
<tr>
<td>88</td>
<td>PACACC</td>
<td>Bad access code</td>
</tr>
<tr>
<td>92</td>
<td>PACTYPE</td>
<td>Bad request type</td>
</tr>
<tr>
<td>96</td>
<td>PACDLI</td>
<td>DL/I error</td>
</tr>
<tr>
<td>100</td>
<td>PACAINV</td>
<td>Bad PACA specification</td>
</tr>
<tr>
<td>104</td>
<td>PACERR</td>
<td>Unexpected condition</td>
</tr>
</tbody>
</table>

Figure 68. Summary of Return Codes for a Text Retrieval Request
Retrieving a Subject's Aliases (RA Call)

When retrieving a subject's aliases, your program obtains a list of the alias names for a particular Dictionary subject.

The subject name specified by your program need not be the subject's primary name. If the subject name specified in the control area (PACA) is itself an alias, it is listed with the retrieved aliases; the primary name for that subject is written into the PANAME field.

When retrieving a subject's aliases, your program must specify the fields shown in Figure 69 in the control area (PACA).

**Required Control-Area Fields**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Required Contents (If Any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>PADAOP</td>
<td>B</td>
<td>N</td>
</tr>
<tr>
<td>PATYPE</td>
<td>RA</td>
<td></td>
</tr>
<tr>
<td>PAFTU</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>PADAP</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PADAL</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PAFTP</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PAFTL</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PANAME</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PACAT</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PAACCESS</td>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>PAPNAME</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PANREQ</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PAINCR</td>
<td>---</td>
<td></td>
</tr>
</tbody>
</table>

Figure 69. Required PACA Fields for an RA Request
Master Format Table

The master format table (PADAFT) for an alias retrieval contains a single entry for each alias in the data area (PADA). The alias name is 36 bytes long. The master format table (PADAFT) entry for each alias is:

<table>
<thead>
<tr>
<th>Keyword (PAFNAME)</th>
<th>Instance (PAFINST)</th>
<th>Type (PAFDTYPE)</th>
<th>Offset (PAFDATA)</th>
<th>Length (PAFLENG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIAS</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>36</td>
</tr>
</tbody>
</table>

Positioning Field

The last alias name to be retrieved can be controlled by specifying the PAPNAME field described in Figure 70.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPNAME</td>
<td>Positioning: the last alias name retrieved</td>
</tr>
</tbody>
</table>

Figure 70. Optional Field to Control Last Alias Name Retrieved

Returned Control-Area Data

When the Dictionary returns control to your program after processing an alias retrieval request, the control area (PACA) fields shown in Figure 71 contain information set by the Dictionary.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PANRETR</td>
<td>Number of aliases retrieved</td>
</tr>
<tr>
<td>PAFTLM</td>
<td>Length of the master PADAFT</td>
</tr>
<tr>
<td>PARC</td>
<td>Overall return code for the request</td>
</tr>
<tr>
<td>PART</td>
<td>Highest of the return codes set in the PADAFT entries (PAFRC)</td>
</tr>
</tbody>
</table>

Figure 71. PACA Fields Set by the Dictionary on Return from an RA Request
Return Code Table

One of the return codes shown in Figure 72 is set by the Dictionary. "Appendix B.1. Overall Status of a Program Access CALL Statement" on page 408, describes each return code in detail.

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PACOK</td>
<td>Successful retrieval</td>
</tr>
<tr>
<td>4</td>
<td>PACEOD</td>
<td>End of data</td>
</tr>
<tr>
<td>6</td>
<td>PACLAST</td>
<td>Positioned on last</td>
</tr>
<tr>
<td>8</td>
<td>PACNF</td>
<td>No data found</td>
</tr>
<tr>
<td>12</td>
<td>PACPNF</td>
<td>Position not found</td>
</tr>
<tr>
<td>16</td>
<td>PACDATA</td>
<td>No data specified</td>
</tr>
<tr>
<td>20</td>
<td>PACAUTH</td>
<td>User not authorized</td>
</tr>
<tr>
<td>32</td>
<td>PACNAME</td>
<td>No such subject</td>
</tr>
<tr>
<td>36</td>
<td>PACDOWN</td>
<td>Bad down count</td>
</tr>
<tr>
<td>40</td>
<td>PACNREQ</td>
<td>Bad number of items</td>
</tr>
<tr>
<td>44</td>
<td>PACINCR</td>
<td>Bad increment</td>
</tr>
<tr>
<td>46</td>
<td>PACPNAME</td>
<td>No such positioning subject</td>
</tr>
<tr>
<td>60</td>
<td>PACCAT</td>
<td>Bad primary category</td>
</tr>
<tr>
<td>64</td>
<td>PACFTU</td>
<td>Bad primary usage value</td>
</tr>
<tr>
<td>68</td>
<td>PACFTO</td>
<td>PADAFT area overflow</td>
</tr>
<tr>
<td>72</td>
<td>PACFT</td>
<td>Bad PADAFT specification</td>
</tr>
<tr>
<td>76</td>
<td>PACDAOP</td>
<td>Bad PADA operation</td>
</tr>
<tr>
<td>80</td>
<td>PACDAO</td>
<td>Data area overflow</td>
</tr>
<tr>
<td>84</td>
<td>PACDA</td>
<td>Bad PADA specification</td>
</tr>
<tr>
<td>88</td>
<td>PACACC</td>
<td>Bad access code</td>
</tr>
<tr>
<td>92</td>
<td>PACTYPE</td>
<td>Bad request type</td>
</tr>
<tr>
<td>96</td>
<td>PACDLI</td>
<td>DL/I error</td>
</tr>
<tr>
<td>100</td>
<td>PACAINV</td>
<td>Bad PACA specification</td>
</tr>
<tr>
<td>104</td>
<td>PACERR</td>
<td>Unexpected condition</td>
</tr>
</tbody>
</table>

Figure 72. Summary of Return Codes for an RA Retrieval Request
Retrieving Subject Names in a Category (RSN and RSAN Calls)

Either of two types of retrieval requests can provide your program with a list of the subjects in a specified category:

- RSN lists the primary names of subjects in a category.
- RSAN lists the primary names and aliases of subjects in a category.

When retrieving a list of subject names, your program must specify the fields shown in Figure 73 in the PACA.

**Required Control-Area Fields**

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (if any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>PATYPE</td>
<td>RSN</td>
<td>RSAN</td>
</tr>
<tr>
<td>PADAOP</td>
<td>B</td>
<td>N</td>
</tr>
<tr>
<td>PAFTU</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>PASTATUS</td>
<td>A-T, 0-9, *, blank</td>
<td>Retrieve subject names by status. Blank indicates all subject names to be retrieved regardless of status. Valid only when PACAFM=B.</td>
</tr>
<tr>
<td>PADAP</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PADAL</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PAFTP</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PAFTL</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PACAT</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PAACCESS</td>
<td>F</td>
<td>N</td>
</tr>
</tbody>
</table>

Figure 73 (Part 1 of 2). Required PACA Fields for an RSN or RSAN Request
<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (if any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPNAME</td>
<td>---</td>
<td>A &quot;starting name&quot; for the retrieval, or blank if the returned list is to start with the first subject name</td>
</tr>
<tr>
<td>PANREQ</td>
<td>---</td>
<td>The number of subject names to be retrieved</td>
</tr>
<tr>
<td>PAINCR</td>
<td>---</td>
<td>The PADA position increment: the number of bytes from the start of one subject name to the start of the next in the PADA</td>
</tr>
</tbody>
</table>

Figure 73 (Part 2 of 2). Required PACA Fields for an RSN or RSAN Request

Note: The retrieved subject names (in the data area (PADA)) are not sorted in any particular order.

Master Format Table (RSN)

The master format table (PADAFT) for an RSN retrieval defines a 36-byte data area (PADA) field for the primary name of a subject:

<table>
<thead>
<tr>
<th>Keyword (PAFNAME)</th>
<th>Instance (PAFINST)</th>
<th>Type (PAFDTYP)</th>
<th>Offset (PAFDATA)</th>
<th>Length (PAFLENG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBJECT</td>
<td>0</td>
<td>I</td>
<td>0</td>
<td>36</td>
</tr>
</tbody>
</table>

Master Format Table (RSAN)

The master PADAFT for an RSAN retrieval contains two entries:

- The first entry defines a 36-byte data area (PADA) field for a subject’s alias.
- The second defines a 36-byte data area (PADA) field for the subject’s primary name.

<table>
<thead>
<tr>
<th>Keyword (PAFNAME)</th>
<th>Instance (PAFINST)</th>
<th>Type (PAFDTYP)</th>
<th>Offset (PAFDATA)</th>
<th>Length (PAFLENG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBJECT</td>
<td>0</td>
<td>I</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>PRIMARY</td>
<td>0</td>
<td>I</td>
<td>36</td>
<td>36</td>
</tr>
</tbody>
</table>

Each time an alias is retrieved, a pair of names is put into the data area (PADA). The alias name is put into the first field; its corresponding subject’s primary name is put into the second field. If a primary name (not an alias) is retrieved, the primary name is put into the first of the two data area (PADA) fields; the second field is set to blanks.
Positioning Field

The last subject name (primary or alias) to be retrieved may be specified by the PAPNAME field described in Figure 74.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPNAME</td>
<td>Positioning: The last subject name (primary or alias) retrieved</td>
</tr>
</tbody>
</table>

Figure 74. Optional Field to Control Last Subject Name Retrieved

Returned Control-Area Data

When the Dictionary returns control to your program after processing an RSN or RSAN request, the control area (PACA) fields shown in Figure 75 contain information set by the Dictionary.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PANRETR</td>
<td>Number of names (or primary-alias pairs) retrieved</td>
</tr>
<tr>
<td>PAFILM</td>
<td>Length of the master PADAFT</td>
</tr>
<tr>
<td>PARC</td>
<td>Overall return code for the request</td>
</tr>
<tr>
<td>PART</td>
<td>Highest of the return codes set in the PADAFT entries (PAFRC)</td>
</tr>
</tbody>
</table>

Figure 75. PACA Fields Set by the Dictionary on Return from an RSN or RSAN Request
Return Code Table

One of the return codes shown in Figure 76, is set by the Dictionary upon return from an RSN or RSAN retrieval request. “Appendix B.1. Overall Status of a Program Access CALL Statement” on page 408, describes each return code in detail.

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PACOK</td>
<td>Successful retrieval</td>
</tr>
<tr>
<td>4</td>
<td>PACEOD</td>
<td>End of data</td>
</tr>
<tr>
<td>6</td>
<td>PACLAST</td>
<td>Positioned on last</td>
</tr>
<tr>
<td>8</td>
<td>PACNF</td>
<td>No data found</td>
</tr>
<tr>
<td>12</td>
<td>PACPNF</td>
<td>Position not found</td>
</tr>
<tr>
<td>16</td>
<td>PACDATA</td>
<td>No data specified</td>
</tr>
<tr>
<td>20</td>
<td>PACAUTH</td>
<td>User not authorized</td>
</tr>
<tr>
<td>36</td>
<td>PACDOWN</td>
<td>Bad down count</td>
</tr>
<tr>
<td>40</td>
<td>PACNREQ</td>
<td>Bad number of items</td>
</tr>
<tr>
<td>44</td>
<td>PACINCR</td>
<td>Bad increment</td>
</tr>
<tr>
<td>46</td>
<td>PACPNAME</td>
<td>No such positioning subject</td>
</tr>
<tr>
<td>60</td>
<td>PACCAT</td>
<td>Bad primary category</td>
</tr>
<tr>
<td>64</td>
<td>PACFTU</td>
<td>Bad primary usage value</td>
</tr>
<tr>
<td>68</td>
<td>PACFTO</td>
<td>PADAFT area overflow</td>
</tr>
<tr>
<td>72</td>
<td>PACFT</td>
<td>Bad PADAFT specification</td>
</tr>
<tr>
<td>76</td>
<td>PACDAOP</td>
<td>Bad PADA operation</td>
</tr>
<tr>
<td>80</td>
<td>PACDAO</td>
<td>Data area overflow</td>
</tr>
<tr>
<td>84</td>
<td>PACDA</td>
<td>Bad PADA specification</td>
</tr>
<tr>
<td>88</td>
<td>PACACC</td>
<td>Bad access code</td>
</tr>
<tr>
<td>92</td>
<td>PACTYPE</td>
<td>Bad request type</td>
</tr>
<tr>
<td>96</td>
<td>PACDLI</td>
<td>DL/I error</td>
</tr>
<tr>
<td>100</td>
<td>PACAINV</td>
<td>Bad PACA specification</td>
</tr>
<tr>
<td>104</td>
<td>PACERR</td>
<td>Unexpected condition</td>
</tr>
</tbody>
</table>

Figure 76. Summary of Return Codes for an RSN or RSAN Retrieval Request
Retrieving Relationships for a Subject (RSR, RSRC, RSRK Calls)

Your program can request three types of retrieval for a list of the relationships in which a subject participates:

- RSR to retrieve all subjects related to the specified subject
- RSRC to retrieve all subjects in a specified category related to the specified subject
- RSRK to retrieve all subjects in a specified category related to the specified subject with a specified relationship keyword

The RSRC and RSRK types of relationship retrieval allow your program to more selectively list the subjects related to a specified subject.

*Note:* Your program can specify either the subject's primary name or its alias. Names of related subjects returned by the Dictionary, however, are primary names.
Retrieve Subject Relationships (RSR)

When retrieving a list of all subjects related to the specified subject, your program must specify the control area (PACA) fields shown in Figure 77.

**Required Control-Area Fields**

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (if any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>PATYPE</td>
<td>RSR</td>
<td>Request type: retrieve all subjects related to a subject.</td>
</tr>
<tr>
<td>PAFTU</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>PADAOP</td>
<td>B</td>
<td>N</td>
</tr>
<tr>
<td>PADAP</td>
<td>---</td>
<td>Address of the PADA</td>
</tr>
<tr>
<td>PADAL</td>
<td>---</td>
<td>Length of the PADA</td>
</tr>
<tr>
<td>PAFTP</td>
<td>---</td>
<td>Address of the PADAFT</td>
</tr>
<tr>
<td>PAFTL</td>
<td>---</td>
<td>Length of the PADAFT</td>
</tr>
<tr>
<td>PANAME</td>
<td>---</td>
<td>Name of the Dictionary subject (or one of its alias names)</td>
</tr>
<tr>
<td>PACAT</td>
<td>---</td>
<td>Name of the subject's category</td>
</tr>
<tr>
<td>PAACCESS</td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>PANREQ</td>
<td>---</td>
<td>The number of relationships to be retrieved</td>
</tr>
<tr>
<td>PAINCR</td>
<td>---</td>
<td>The PADA position increment: the number of bytes from the start of one block of relationship data to the start of the next in the PADA</td>
</tr>
</tbody>
</table>

Figure 77. Required PACA Fields for an RSR Request
Positioning Fields

The starting position of the relationships to be retrieved can be controlled by specifying the control area (PACA) fields shown in Figure 78. These fields are usually set to all blanks for an initial RSR request.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (if any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPRKW</td>
<td>---</td>
<td>Positioning:¹ relationship keyword</td>
</tr>
<tr>
<td>PAPRCAT</td>
<td>---</td>
<td>Positioning:¹ name of a related subject category</td>
</tr>
<tr>
<td>PAPRNAME</td>
<td>---</td>
<td>Positioning:¹ name of a related subject</td>
</tr>
<tr>
<td>PAPRSEQ</td>
<td>---</td>
<td>Positioning:¹ value of the sequencing attribute of a relationship between the specified subject (named in PANAME) and its related subject</td>
</tr>
</tbody>
</table>

Figure 78. Optional Fields to Control Starting Position for RSR Requests

Note to Figure 78:

¹ The positioning parameters cannot be specified independently. You must specify both the keyword and its associated category. There is a positioning dependency between PAPRKW and PAPRCAT, and between PAPRNAME and PAPRSEQ.
Returned Control-Area Data

When the Dictionary returns control to your program after processing an RSR request, the control area (PACA) fields shown in Figure 79 contain information set by the Dictionary.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPRKW</td>
<td>Positioning: relationship keyword</td>
</tr>
<tr>
<td>PAPRCAT</td>
<td>Positioning: category of related subject</td>
</tr>
<tr>
<td>PAPRNAME</td>
<td>Positioning: subject name of related subject</td>
</tr>
<tr>
<td>PAPRSEQ</td>
<td>Positioning: relationship sequence value</td>
</tr>
<tr>
<td>PANRETR</td>
<td>Number of instances of related-subject information retrieved</td>
</tr>
<tr>
<td>PAFTLM</td>
<td>Length of the master PADAFT</td>
</tr>
<tr>
<td>PARC</td>
<td>Overall return code for the request</td>
</tr>
<tr>
<td>PART</td>
<td>Highest of the return codes set in the PADAFT entries (PAFRC)</td>
</tr>
</tbody>
</table>

Figure 79. PACA Fields Set by the Dictionary on Return from an RSR Request
Return Code Table

One of the return codes shown in Figure 80, is set by the Dictionary upon return from an RSR retrieval request. "Appendix B.1. Overall Status of a Program Access CALL Statement" on page 408, describes each return code in detail.

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PACOK</td>
<td>Successful retrieval</td>
</tr>
<tr>
<td>4</td>
<td>PACEOD</td>
<td>End of data</td>
</tr>
<tr>
<td>6</td>
<td>PACLAST</td>
<td>Positioned on last</td>
</tr>
<tr>
<td>8</td>
<td>PACNF</td>
<td>No data found</td>
</tr>
<tr>
<td>12</td>
<td>PACPNF</td>
<td>Position not found</td>
</tr>
<tr>
<td>16</td>
<td>PACDATA</td>
<td>No data specified</td>
</tr>
<tr>
<td>20</td>
<td>PACAUTH</td>
<td>User not authorized</td>
</tr>
<tr>
<td>32</td>
<td>PACNAME</td>
<td>No such subject</td>
</tr>
<tr>
<td>36</td>
<td>PACDOWN</td>
<td>Bad down count</td>
</tr>
<tr>
<td>40</td>
<td>PACNREQ</td>
<td>Bad number of items</td>
</tr>
<tr>
<td>44</td>
<td>PACINCR</td>
<td>Bad increment</td>
</tr>
<tr>
<td>46</td>
<td>PACPNAME</td>
<td>No such positioning subject</td>
</tr>
<tr>
<td>48</td>
<td>PACPCAT</td>
<td>No such positioning category</td>
</tr>
<tr>
<td>52</td>
<td>PACRKW</td>
<td>No relationship-type exists</td>
</tr>
<tr>
<td>60</td>
<td>PACCAT</td>
<td>Bad primary category</td>
</tr>
<tr>
<td>64</td>
<td>PACFTU</td>
<td>Bad primary usage value</td>
</tr>
<tr>
<td>68</td>
<td>PACFTO</td>
<td>PADAFT area overflow</td>
</tr>
<tr>
<td>72</td>
<td>PACFT</td>
<td>Bad PADAFT specification</td>
</tr>
<tr>
<td>76</td>
<td>PACDAOP</td>
<td>Bad PADA operation</td>
</tr>
<tr>
<td>80</td>
<td>PACDAO</td>
<td>Data area overflow</td>
</tr>
<tr>
<td>84</td>
<td>PACDA</td>
<td>Bad PADA specification</td>
</tr>
<tr>
<td>88</td>
<td>PACACC</td>
<td>Bad access code</td>
</tr>
<tr>
<td>92</td>
<td>PACTYPE</td>
<td>Bad request type</td>
</tr>
<tr>
<td>96</td>
<td>PACDLI</td>
<td>DL/I error</td>
</tr>
<tr>
<td>100</td>
<td>PACAINV</td>
<td>Bad PACA specification</td>
</tr>
<tr>
<td>104</td>
<td>PACERR</td>
<td>Unexpected condition</td>
</tr>
</tbody>
</table>

Figure 80. Summary of Return Codes for an RSR Retrieval Request
Retrieve Subject Relationships by Related Category (RSRC)

When retrieving a list of all relationships between a specified subject and subjects in a specified category, your program must specify the control area (PACA) fields shown in Figure 81.

Required Control-Area Fields

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (If Any)*</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>PATYPE</td>
<td>RSRC</td>
<td>Request type: List all subjects in a category related to a subject.</td>
</tr>
<tr>
<td>PAFTU</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>PADAP</td>
<td>---</td>
<td>Address of the PADA</td>
</tr>
<tr>
<td>PADAL</td>
<td>---</td>
<td>Length of the PADA</td>
</tr>
<tr>
<td>PAFTP</td>
<td>---</td>
<td>Address of the PADAFT</td>
</tr>
<tr>
<td>PAFTL</td>
<td>---</td>
<td>Length of the PADAFT</td>
</tr>
<tr>
<td>PANAME</td>
<td>---</td>
<td>Name of the Dictionary subject (or one of its alias names)</td>
</tr>
<tr>
<td>PACAT</td>
<td>---</td>
<td>Name of the subject’s category</td>
</tr>
<tr>
<td>PARCAT</td>
<td>---</td>
<td>Name of the category containing related subjects</td>
</tr>
<tr>
<td>PAACCESS</td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>PANREQ</td>
<td>---</td>
<td>The number of relationships to be retrieved</td>
</tr>
<tr>
<td>PAINCR</td>
<td>---</td>
<td>The PADA position increment: the number of bytes from the start of one block of relationship data to the start of the next in the PADA</td>
</tr>
</tbody>
</table>

Figure 81. Required PACA Fields for an RSRC Request
Positioning Fields

The starting position of the relationships to be retrieved with the RSRC request can be controlled by specifying the control area (PACA) fields shown in Figure 82.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (If Any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPRKW</td>
<td>---</td>
<td>Positioning: relationship keyword</td>
</tr>
<tr>
<td>PAPRNAME</td>
<td>---</td>
<td>Positioning: name of a related subject</td>
</tr>
<tr>
<td>PAPRSEQ</td>
<td>---</td>
<td>Positioning: value of the sequencing attribute of a relationship between the specified subject (named in PANAME) and its related subject</td>
</tr>
</tbody>
</table>

Figure 82. Optional Fields to Control Starting Position for RSRC Requests

Note to Figure 82:

1. The positioning parameters cannot be specified independently. There is a positioning dependency between PAPRNAME and PAPRSEQ.

Returned Control-Area Data

When the Dictionary returns control to your program after processing an RSRC request, the control area (PACA) fields shown in Figure 83 contain information set by the Dictionary.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPRKW</td>
<td>Positioning: relationship keyword</td>
</tr>
<tr>
<td>PAPRCAT</td>
<td>Positioning: category of related subject</td>
</tr>
<tr>
<td>PAPRNAME</td>
<td>Positioning: subject name of related subject</td>
</tr>
<tr>
<td>PAPRSEQ</td>
<td>Positioning: relationship sequence value</td>
</tr>
<tr>
<td>PANRETR</td>
<td>Number of sets of related-subject information retrieved</td>
</tr>
<tr>
<td>PAFTLM</td>
<td>Length of the master PADAFT</td>
</tr>
<tr>
<td>PARC</td>
<td>Overall return code for the request</td>
</tr>
<tr>
<td>PART</td>
<td>Highest return code set in the PADAFT</td>
</tr>
</tbody>
</table>

Figure 83. PACA Fields Set by the Dictionary on Return from an RSRC Request
Return Code Table

One of the return codes shown in Figure 84 is set by the Dictionary upon return from an RSRC retrieval request. "Appendix B.1. Overall Status of a Program Access CALL Statement" on page 408, describes each return code in detail.

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PACOK</td>
<td>Successful retrieval</td>
</tr>
<tr>
<td>4</td>
<td>PACEOD</td>
<td>End of data</td>
</tr>
<tr>
<td>6</td>
<td>PACLAST</td>
<td>Positioned on last</td>
</tr>
<tr>
<td>8</td>
<td>PACNF</td>
<td>No data found</td>
</tr>
<tr>
<td>12</td>
<td>PACPNF</td>
<td>Position not found</td>
</tr>
<tr>
<td>16</td>
<td>PACDATA</td>
<td>No data specified</td>
</tr>
<tr>
<td>20</td>
<td>PACAUTH</td>
<td>User not authorized</td>
</tr>
<tr>
<td>32</td>
<td>PACNAME</td>
<td>No such subject</td>
</tr>
<tr>
<td>36</td>
<td>PACDOWN</td>
<td>Bad down count</td>
</tr>
<tr>
<td>40</td>
<td>PACNREQ</td>
<td>Bad number of items</td>
</tr>
<tr>
<td>44</td>
<td>PACINCR</td>
<td>Bad increment</td>
</tr>
<tr>
<td>46</td>
<td>PACPNAME</td>
<td>No such positioning subject</td>
</tr>
<tr>
<td>52</td>
<td>PACRKW</td>
<td>No relationship-type exists</td>
</tr>
<tr>
<td>56</td>
<td>PACRCAT</td>
<td>Bad related category</td>
</tr>
<tr>
<td>60</td>
<td>PACCAT</td>
<td>Bad primary category</td>
</tr>
<tr>
<td>64</td>
<td>PACFTU</td>
<td>Bad primary usage value</td>
</tr>
<tr>
<td>68</td>
<td>PACFTO</td>
<td>PADAFT area overflow</td>
</tr>
<tr>
<td>72</td>
<td>PACFT</td>
<td>Bad PADAFT specification</td>
</tr>
<tr>
<td>76</td>
<td>PACDAOP</td>
<td>Bad PADA operation</td>
</tr>
<tr>
<td>80</td>
<td>PACDAO</td>
<td>Data area overflow</td>
</tr>
<tr>
<td>84</td>
<td>PACDA</td>
<td>Bad PADA specification</td>
</tr>
<tr>
<td>88</td>
<td>PACACC</td>
<td>Bad access code</td>
</tr>
<tr>
<td>92</td>
<td>PACTYPE</td>
<td>Bad request type</td>
</tr>
<tr>
<td>96</td>
<td>PACDLI</td>
<td>DL/I error</td>
</tr>
<tr>
<td>100</td>
<td>PACAINV</td>
<td>Bad PACA specification</td>
</tr>
<tr>
<td>104</td>
<td>PACERR</td>
<td>Unexpected condition</td>
</tr>
</tbody>
</table>

Figure 84. Summary of Return Codes for an RSRC Retrieval Request
Retrieve Subject Relationships, by Related Category and Keyword (RSRK)

When retrieving a list of all relationships between a specified subject and subjects in a specified category with a specified keyword, your program must specify the control area (PACA) fields shown in Figure 85.

**Required Control-Area Fields**

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (If Any)**</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>PATYPE</td>
<td>RSRK</td>
<td>Request type: List all subjects in a category related to a subject with specified relationship keyword.</td>
</tr>
<tr>
<td>PAFTU</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>PADAOP</td>
<td>B</td>
<td>N</td>
</tr>
<tr>
<td>PADAP</td>
<td>---</td>
<td>Address of the PADA</td>
</tr>
<tr>
<td>PADAL</td>
<td>---</td>
<td>Length of the PADA</td>
</tr>
<tr>
<td>PAFTP</td>
<td>---</td>
<td>Address of the PADAFT</td>
</tr>
<tr>
<td>PAFTL</td>
<td>---</td>
<td>Length of the PADAFT</td>
</tr>
<tr>
<td>PANAME</td>
<td>---</td>
<td>Name of the Dictionary subject (or one of its alias names)</td>
</tr>
<tr>
<td>PACAT</td>
<td>---</td>
<td>Name of the subject's category</td>
</tr>
<tr>
<td>PARCAT</td>
<td>---</td>
<td>Name of the category containing related subjects</td>
</tr>
<tr>
<td>PARKW</td>
<td>---</td>
<td>Name of the relationship keyword</td>
</tr>
<tr>
<td>PAACCESS</td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>PANREQ</td>
<td>---</td>
<td>The number of relationships to be retrieved</td>
</tr>
<tr>
<td>PAINCR</td>
<td>---</td>
<td>The PADA position increment: the number of bytes from the start of one block of relationship data to the start of the next in the PADA</td>
</tr>
</tbody>
</table>

Figure 85. Required PACA Fields for an RSRK Request
Positioning Fields

The starting position of the relationships to be retrieved can be controlled by specifying the control area (PACA) fields shown in Figure 86.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (If Any)*</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPRNAME</td>
<td>---</td>
<td>Positioning: Name of a related subject</td>
</tr>
<tr>
<td>PAPRSEQ</td>
<td>---</td>
<td>Positioning: Value of the sequencing attribute of a relationship between the specified subject (named in PANAME) and its related subject</td>
</tr>
</tbody>
</table>

Figure 86. Optional Fields to Control Starting Position for RSRK Requests

Note to Figure 86:

1 The positioning parameters cannot be specified independently. There is a positioning dependency between PAPRNAME and PAPRSEQ.

Returned Control-Area Data

When the Dictionary returns control to your program after processing an RSRK request, the control area (PACA) fields shown in Figure 87 contain information set by the Dictionary.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPRNAME</td>
<td>Positioning: subject name of related subject</td>
</tr>
<tr>
<td>PAPRSEQ</td>
<td>Positioning: relationship sequence value</td>
</tr>
<tr>
<td>PANRETR</td>
<td>Number of sets of related-subject information retrieved</td>
</tr>
<tr>
<td>PACTLM</td>
<td>Length of the master PADAFT</td>
</tr>
<tr>
<td>PARC</td>
<td>Overall return code for the request</td>
</tr>
<tr>
<td>PART</td>
<td>Highest return code set in the PADAFT</td>
</tr>
</tbody>
</table>

Figure 87. PACA Fields Set by the Dictionary on Return from an RSRK Request
Return Code Table

One of the return codes shown in Figure 88 is set by the Dictionary upon return from an RSRK retrieval request. "Appendix B.1. Overall Status of a Program Access CALL Statement" on page 408, describes each return code in detail.

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PACOK</td>
<td>Successful retrieval</td>
</tr>
<tr>
<td>6</td>
<td>PACLAST</td>
<td>Positioned on last</td>
</tr>
<tr>
<td>8</td>
<td>PACNF</td>
<td>No data found</td>
</tr>
<tr>
<td>12</td>
<td>PACPNF</td>
<td>Position not found</td>
</tr>
<tr>
<td>16</td>
<td>PACDATA</td>
<td>No data specified</td>
</tr>
<tr>
<td>20</td>
<td>PACAUTH</td>
<td>User not authorized</td>
</tr>
<tr>
<td>32</td>
<td>PACNAME</td>
<td>No such subject</td>
</tr>
<tr>
<td>36</td>
<td>PACDOWM</td>
<td>Bad down count</td>
</tr>
<tr>
<td>40</td>
<td>PACNREQ</td>
<td>Bad number of items</td>
</tr>
<tr>
<td>44</td>
<td>PACINCR</td>
<td>Bad increment</td>
</tr>
<tr>
<td>46</td>
<td>PACPNAM</td>
<td>No such positioning subject</td>
</tr>
<tr>
<td>52</td>
<td>PACRKW</td>
<td>Bad relationship keyword</td>
</tr>
<tr>
<td>56</td>
<td>PACRCAT</td>
<td>Bad related category</td>
</tr>
<tr>
<td>60</td>
<td>PACCAT</td>
<td>Bad primary category</td>
</tr>
<tr>
<td>64</td>
<td>PACFTU</td>
<td>Bad primary usage value</td>
</tr>
<tr>
<td>68</td>
<td>PACFTO</td>
<td>PADAFT area overflow</td>
</tr>
<tr>
<td>72</td>
<td>PACFT</td>
<td>Bad PADAFT specification</td>
</tr>
<tr>
<td>76</td>
<td>PACDAOP</td>
<td>Bad PADA operation</td>
</tr>
<tr>
<td>80</td>
<td>PACDAO</td>
<td>Data area overflow</td>
</tr>
<tr>
<td>84</td>
<td>PACDA</td>
<td>Bad PADA specification</td>
</tr>
<tr>
<td>88</td>
<td>PACACC</td>
<td>Bad access code</td>
</tr>
<tr>
<td>92</td>
<td>PACTYPE</td>
<td>Bad request type</td>
</tr>
<tr>
<td>96</td>
<td>PACDLI</td>
<td>DL/I error</td>
</tr>
<tr>
<td>100</td>
<td>PACAINV</td>
<td>Bad PACA specification</td>
</tr>
<tr>
<td>104</td>
<td>PACERR</td>
<td>Unexpected condition</td>
</tr>
</tbody>
</table>

Figure 88. Summary of Return Codes for an RSRK Retrieval Request
Master Format Table (PADAFT) for Relationship Retrieval Requests (RSR, RSRC, RSRK)

The master PADAFT for the three types of relationship retrieval requests consists of entries for the following attributes. The length of each block of relationship data retrieved by the Dictionary is 77 bytes. The entries that comprise the block are:

**RELKWORD** The forward relationship keyword.

**RELCAT** The name of the related subject category.

**RELSUBJ** The name of the related subject. The name returned in the PADA is always the primary name of the subject.

**SEQVAL** The sequence attribute value.

For sequenced standard relationships, the START relationship attribute may involve a BITSTART parameter. In SEQVAL, the format is 's6ssss/B', where ssss is the START value, and B is the BITSTART value. See "Appendix B.8. Special Topics for Attribute Retrieval" on page 462 for details of sequence values.

**INVKWORD** The inverse relationship keyword.

**DIRECT** A code indicating the structural direction of the relationship the subject participates in.

- **D** The relationship is "downward" to the related subject.
- **U** The relationship is "upward" to the related subject.
- **N** The relationship is not directed.

Master Format Table

The master PADAFT (that is, the set of entries) for a retrieved relationship instance is:

<table>
<thead>
<tr>
<th>Keyword (PAFNAME)</th>
<th>Instance (PAFINST)</th>
<th>Type (PAFDTYPE)</th>
<th>Offset (PAFDATA)</th>
<th>Length (PAFLENG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELKWORD</td>
<td>0</td>
<td>C</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>RELCAT</td>
<td>0</td>
<td>C</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>RELSUBJ</td>
<td>0</td>
<td>I</td>
<td>20</td>
<td>36</td>
</tr>
<tr>
<td>SEQVAL</td>
<td>0</td>
<td>C</td>
<td>56</td>
<td>8</td>
</tr>
<tr>
<td>INVKWORD</td>
<td>0</td>
<td>C</td>
<td>64</td>
<td>12</td>
</tr>
<tr>
<td>DIRECT</td>
<td>0</td>
<td>C</td>
<td>76</td>
<td>1</td>
</tr>
</tbody>
</table>

For RSRC and RSRK requests, the Dictionary returns values (in the data area (PADA)) for the related category and keyword, although these values repeat the values provided in the control area (PACA) with the retrieval request.
Retrieving a List of Relationship-Types Allowed for a Category (RCR Call)

A "category-relationships" retrieval request provides your program with a list of the types of relationships in which any subject in the specified Dictionary category may participate. When requesting this type of retrieval, your program must specify the fields shown in Figure 89 in the control area (PACA).

Required Control-Area Fields

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (If Any)* *</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>PATYPE</td>
<td>RCR</td>
<td>Request type: List all types of relationships subjects in a category can participate in.</td>
</tr>
<tr>
<td>PAFTU</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>PADAOP</td>
<td>B</td>
<td>N</td>
</tr>
<tr>
<td>PADAP</td>
<td>---</td>
<td>Address of the PADA</td>
</tr>
<tr>
<td>PADAL</td>
<td>---</td>
<td>Length of the PADA</td>
</tr>
<tr>
<td>PAFTP</td>
<td>---</td>
<td>Address of the PADAFT</td>
</tr>
<tr>
<td>PAFTL</td>
<td>---</td>
<td>Length of the PADAFT</td>
</tr>
<tr>
<td>PACAT</td>
<td>---</td>
<td>Name of the subject category</td>
</tr>
<tr>
<td>PAACCESS</td>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>PANREQ</td>
<td>---</td>
<td>The number of relationship types to be retrieved</td>
</tr>
<tr>
<td>PAINCR</td>
<td>---</td>
<td>The PADA position increment: the number of bytes from the start of one block of relationship-type data to the start of the next in the PADA</td>
</tr>
</tbody>
</table>

Figure 89. Required PACA Fields for an RCR Request
Format Table (PADAFT) for Relationship-Type Retrieval Requests

The master PADAFT for a relationship-type retrieval request consists of entries for the following attributes. The length of each block of relationship-type data retrieved by the Dictionary is 78 bytes. The entries that comprise the block are:

**RELKW**ORD 
The relationship keyword identifying the relationship-type from the principal category to the related category (that is, the forward keyword)

**RELCAT** 
The name of the related category

**RELTYPID** 
The (primary) dictionary subject name for the relationship-type definition

**SEQOPT** 
An indication of whether the relationship-type occurrences are sequenced or not (indicated by values Y (sequenced) and N (not sequenced))

**SEQATTR** 
The name of the sequence attribute for the relationship-type, if SEQOPT=Y

**INVKWORD** 
The keyword identifying the relationship-type from the related category to the principal category (that is, the inverse keyword)

**DIRECT** 
An indication of the structural direction of the relationship a subject in the category participates in:

- D The relationship is “downward” to a related subject.
- U The relationship is “upward” to a related subject.
- N The relationship is not directed.

Master Format Table

The master PADAFT (that is, the set of entries) for a retrieved relationship-type instance is:

<table>
<thead>
<tr>
<th>Keyword (PAFNAME)</th>
<th>Instance (PAFINST)</th>
<th>Type (PAFDTYP)</th>
<th>Offset (PAFDATA)</th>
<th>Length (PAFLENG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELKW<strong>ORD</strong></td>
<td>0</td>
<td>C</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>RELCAT</td>
<td>0</td>
<td>C</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>RELTYPID</td>
<td>0</td>
<td>I</td>
<td>20</td>
<td>36</td>
</tr>
<tr>
<td>SEQOPT</td>
<td>0</td>
<td>C</td>
<td>56</td>
<td>1</td>
</tr>
<tr>
<td>SEQATTR</td>
<td>0</td>
<td>C</td>
<td>57</td>
<td>8</td>
</tr>
<tr>
<td>INVKWORD</td>
<td>0</td>
<td>C</td>
<td>65</td>
<td>12</td>
</tr>
<tr>
<td>DIRECT</td>
<td>0</td>
<td>C</td>
<td>77</td>
<td>1</td>
</tr>
</tbody>
</table>
Returned Control-Area Data

When the Dictionary returns control to your program after processing an RCR request, the control area (PACA) fields shown in Figure 90 contain information set by the Dictionary.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPRKW</td>
<td>Positioning: Relationship keyword</td>
</tr>
<tr>
<td>PAPRCAT</td>
<td>Positioning: Category of subjects that can participate in this type of relationship</td>
</tr>
<tr>
<td>PANRETR</td>
<td>Number of sets of relationship-type information retrieved</td>
</tr>
<tr>
<td>PAFTLM</td>
<td>Length of the master PADAFT</td>
</tr>
<tr>
<td>PARC</td>
<td>Overall return code for the request</td>
</tr>
<tr>
<td>PART</td>
<td>Highest return code set in the PADAFT</td>
</tr>
</tbody>
</table>

Figure 90. PACA Fields Set by the Dictionary on Return from an RCR Request

Positioning Fields

The starting position of the relationship-types to be retrieved can be controlled by specifying the control area (PACA) fields shown in Figure 91.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (If Any)*</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPRKW</td>
<td>---</td>
<td>Positioning:1 Relationship keyword</td>
</tr>
<tr>
<td>PAPRCAT</td>
<td>---</td>
<td>Positioning:1 Name of a related subject category</td>
</tr>
</tbody>
</table>

Figure 91. Optional Fields to Control Starting Position for RCR Requests

Note to Figure 91:

1 The positioning parameters cannot be specified independently. You must specify both the keyword and its associated category.
Return Code Table

One of the return codes shown in Figure 92 is set by the Dictionary upon return from an RCR retrieval request. "Appendix B.1. Overall Status of a Program Access CALL Statement" on page 408 describes each return code in detail.

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PACOK</td>
<td>Successful retrieval</td>
</tr>
<tr>
<td>4</td>
<td>PACEOD</td>
<td>End of data</td>
</tr>
<tr>
<td>6</td>
<td>PACLAST</td>
<td>Positioned on last</td>
</tr>
<tr>
<td>8</td>
<td>PACNF</td>
<td>No data found</td>
</tr>
<tr>
<td>12</td>
<td>PACPNF</td>
<td>Position not found</td>
</tr>
<tr>
<td>16</td>
<td>PACDATA</td>
<td>No data specified</td>
</tr>
<tr>
<td>20</td>
<td>PACAUTH</td>
<td>User not authorized</td>
</tr>
<tr>
<td>36</td>
<td>PACDOWN</td>
<td>Bad down count</td>
</tr>
<tr>
<td>40</td>
<td>PACNREQ</td>
<td>Bad number of items</td>
</tr>
<tr>
<td>44</td>
<td>PACINCR</td>
<td>Bad increment</td>
</tr>
<tr>
<td>48</td>
<td>PACPCAT</td>
<td>No such positioning category</td>
</tr>
<tr>
<td>60</td>
<td>PACCAT</td>
<td>Bad primary category</td>
</tr>
<tr>
<td>64</td>
<td>PACFTU</td>
<td>Bad primary usage value</td>
</tr>
<tr>
<td>68</td>
<td>PACFTO</td>
<td>PADAFT area overflow</td>
</tr>
<tr>
<td>72</td>
<td>PACFT</td>
<td>Bad PADAFT specification</td>
</tr>
<tr>
<td>76</td>
<td>PACDAOP</td>
<td>Bad PADA operation</td>
</tr>
<tr>
<td>80</td>
<td>PACDAO</td>
<td>Data area overflow</td>
</tr>
<tr>
<td>84</td>
<td>PACDA</td>
<td>Bad PADA specification</td>
</tr>
<tr>
<td>88</td>
<td>PACACC</td>
<td>Bad access code</td>
</tr>
<tr>
<td>92</td>
<td>PACTYPE</td>
<td>Bad request type</td>
</tr>
<tr>
<td>96</td>
<td>PACDLI</td>
<td>DL/I error</td>
</tr>
<tr>
<td>100</td>
<td>PACAINV</td>
<td>Bad PACA specification</td>
</tr>
<tr>
<td>104</td>
<td>PACERR</td>
<td>Unexpected condition</td>
</tr>
</tbody>
</table>

Figure 92. Summary of Return Codes for an RCR Retrieval Request
Retrieving a List of Relationship-Types Defined for an Installed Structure-Type (RSTR Call)

A "structure-type" retrieval request provides your program with a list of the types of relationships that were defined as composing a structure-type. A restriction requires that the structure-type must be in "installed" status. When requesting this type of retrieval, your program must specify the fields shown in Figure 93 in the control area (PACA).

Required Control-Area Fields

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (If Any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>PATYPE</td>
<td>RSTR</td>
<td>Request type: List all types of relationships in a structure-type</td>
</tr>
<tr>
<td>PADAOP</td>
<td>B</td>
<td>N</td>
</tr>
<tr>
<td>PAFTU</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>PADAP</td>
<td>---</td>
<td>Address of the PADA</td>
</tr>
<tr>
<td>PADAL</td>
<td>---</td>
<td>Length of the PADA</td>
</tr>
<tr>
<td>PAFTP</td>
<td>---</td>
<td>Address of the PADAFT</td>
</tr>
<tr>
<td>PAFTL</td>
<td>---</td>
<td>Length of the PADAFT</td>
</tr>
<tr>
<td>PANAME</td>
<td>---</td>
<td>Name of the structure-type subject</td>
</tr>
<tr>
<td>PAACCESS</td>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>PANREQ</td>
<td>---</td>
<td>The number of relationship types to be retrieved</td>
</tr>
<tr>
<td>PAINCR</td>
<td>---</td>
<td>The PADA position increment: the number of bytes from the start of one block of relationship-type data to the start of the next in the PADA</td>
</tr>
</tbody>
</table>

Figure 93. Required PACA Fields for an RSTR Request
Positioning Fields

The starting position in the (sequenced) list of the relationship-types to be retrieved can be controlled by specifying the PAPRNAME field shown in Figure 94.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (If Any)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPRNAME</td>
<td>--</td>
<td>Positioning: Name of a relationship-type that occurs in the structure-type definition</td>
</tr>
</tbody>
</table>

Figure 94. Optional Field to Control Starting Position for RSTR Requests

Format Table (PADAFT) for Structure-Type Retrieval Requests

The master PADAFT for a relationship-type retrieval request consists of entries for the following attributes. The length of each block of relationship-type data retrieved by the Dictionary is 89 bytes. The entries that comprise the block are:

- RELTYPE: the (primary) dictionary subject name for the relationship-type definition.
- LCATNAME: the name of the left-hand category of the relationship-type.
- FORNAME: the relationship keyword identifying the relationship-type from the left-hand category to the right-hand category.
- RCATNAME: the name of the right-hand category of the relationship-type.
- INVNAME: the keyword identifying the relationship-type from the right-hand category to the left-hand category.
- DIRECT: an indication of whether or not the relationship-type is directed.
  - Y: The relationship-type is directed.
  - N: The relationship-type is not directed.
- SEQOPT: an indication of whether the relationship-type occurrences are sequenced or not (this value will be Y if sequenced).
- SEQATTR: the name of the sequence attribute that orders the instances of relationship-types.
SUPERCAT  the designated or defined superior category in the relationship-type.

L  Left-hand category.

R  Right-hand category.

N  None.

FOLLOWUP  An indication of whether (Y) or not (N) relationships are to be followed when processing in an upward direction.

FOLLOWDN  An indication of whether (Y) or not (N) relationships are to be followed when processing in a downward direction.

Master Format Table

The master PADAFT (that is, the set of entries) for a retrieved relationship-type in the list is:

<table>
<thead>
<tr>
<th>Keyword (PAFNAME)</th>
<th>Instance (PAFINST)</th>
<th>Type (PAFDTYP)</th>
<th>Offset (PAFDATA)</th>
<th>Length (PAFLENG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELTYPE</td>
<td>0</td>
<td>I</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>LCATNAME</td>
<td>0</td>
<td>C</td>
<td>36</td>
<td>8</td>
</tr>
<tr>
<td>FORNAME</td>
<td>0</td>
<td>C</td>
<td>44</td>
<td>8</td>
</tr>
<tr>
<td>RCATNAME</td>
<td>0</td>
<td>C</td>
<td>56</td>
<td>36</td>
</tr>
<tr>
<td>INVNAME</td>
<td>0</td>
<td>C</td>
<td>64</td>
<td>12</td>
</tr>
<tr>
<td>DIRECT</td>
<td>0</td>
<td>C</td>
<td>76</td>
<td>1</td>
</tr>
<tr>
<td>SEOOPT</td>
<td>0</td>
<td>C</td>
<td>77</td>
<td>1</td>
</tr>
<tr>
<td>SEQATTR</td>
<td>0</td>
<td>C</td>
<td>78</td>
<td>8</td>
</tr>
<tr>
<td>SUPERCAT</td>
<td>0</td>
<td>C</td>
<td>86</td>
<td>1</td>
</tr>
<tr>
<td>FOLLOWUP</td>
<td>0</td>
<td>C</td>
<td>87</td>
<td>1</td>
</tr>
<tr>
<td>FOLLOWDN</td>
<td>0</td>
<td>C</td>
<td>88</td>
<td>1</td>
</tr>
</tbody>
</table>
Positioning Field

The name of the last relationship-type retrieved from the list defined for the structure-type can be controlled by specifying the field in Figure 95.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPRNAME</td>
<td>Positioning: Name of last relationship-type retrieved from the list defined for the structure-type</td>
</tr>
</tbody>
</table>

**Figure 95. Optional Field to Control Name of Last Relationship-Type Retrieved**

Returned Control-Area Data

When the Dictionary returns control to your program after processing an RSTR request, the control area (PACA) fields shown in Figure 96 contain information set by the Dictionary.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PANRETR</td>
<td>Number of sets of relationship-type information retrieved</td>
</tr>
<tr>
<td>PAFTLM</td>
<td>Length of the master PADAFT</td>
</tr>
<tr>
<td>PARC</td>
<td>Overall return code for the request</td>
</tr>
<tr>
<td>PART</td>
<td>Highest return code set in the PADAFT</td>
</tr>
</tbody>
</table>

**Figure 96. PACA Fields Set by the Dictionary on Return from an RSTR Request**
### Return Code Table

One of the return codes shown in Figure 97 is set by the Dictionary upon return from an RSTR retrieval request. “Appendix B.1. Overall Status of a Program Access CALL Statement” on page 408, describes each return code in detail.

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PACOK</td>
<td>Successful retrieval</td>
</tr>
<tr>
<td>4</td>
<td>PACEOD</td>
<td>End of data</td>
</tr>
<tr>
<td>6</td>
<td>PACLAST</td>
<td>Positioned on last</td>
</tr>
<tr>
<td>8</td>
<td>PACNF</td>
<td>No data found</td>
</tr>
<tr>
<td>12</td>
<td>PACPNF</td>
<td>Position not found</td>
</tr>
<tr>
<td>16</td>
<td>PACDATA</td>
<td>No data specified</td>
</tr>
<tr>
<td>20</td>
<td>PACAUTH</td>
<td>User not authorized</td>
</tr>
<tr>
<td>36</td>
<td>PACDOWN</td>
<td>Bad down count</td>
</tr>
<tr>
<td>40</td>
<td>PACNREQ</td>
<td>Bad number of items</td>
</tr>
<tr>
<td>44</td>
<td>PACINCR</td>
<td>Bad increment</td>
</tr>
<tr>
<td>60</td>
<td>PACCAT</td>
<td>Bad primary category</td>
</tr>
<tr>
<td>64</td>
<td>PACFTU</td>
<td>Bad primary usage value</td>
</tr>
<tr>
<td>68</td>
<td>PACFTO</td>
<td>PADAFT area overflow</td>
</tr>
<tr>
<td>72</td>
<td>PACFT</td>
<td>Bad PADAFT specification</td>
</tr>
<tr>
<td>76</td>
<td>PACDAOP</td>
<td>Bad PADA operation</td>
</tr>
<tr>
<td>80</td>
<td>PACDAO</td>
<td>Data area overflow</td>
</tr>
<tr>
<td>84</td>
<td>PACDA</td>
<td>Bad PADA specification</td>
</tr>
<tr>
<td>88</td>
<td>PACACC</td>
<td>Bad access code</td>
</tr>
<tr>
<td>92</td>
<td>PACTYPE</td>
<td>Bad request type</td>
</tr>
<tr>
<td>96</td>
<td>PACDLI</td>
<td>DL/I error</td>
</tr>
<tr>
<td>100</td>
<td>PACAINV</td>
<td>Bad PACA specification</td>
</tr>
<tr>
<td>104</td>
<td>PACERR</td>
<td>Unexpected condition</td>
</tr>
<tr>
<td>108</td>
<td>PACSNAME</td>
<td>Structure-type not installed</td>
</tr>
</tbody>
</table>

**Figure 97. Summary of Return Codes for an RSTR Retrieval Request**
User-Program Output

Your program can send character strings to a user's terminal (that is, the user who invokes your program), or to the printer or punch allocated to the Dictionary. A Dictionary stack can also be generated as output.

The maximum length of a line of output data depends on the type of device or destination the data is sent to:

- Online terminal: 79 bytes
- Punch: 80 bytes
- Printer: 120 bytes
- Stack output, each entry: 80 bytes

For an output request, your program must supply the control area (PACA) fields shown in Figure 98.

**Required Control-Area Fields**

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Required Contents (If Any)* *</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACAID</td>
<td>PACA</td>
<td>Control block identifier</td>
</tr>
<tr>
<td>PACAFM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>PATYPE</td>
<td>WTRM WPRT WPCH WSTK</td>
<td>Request type: Send output to the terminal, printer, punch, or stack</td>
</tr>
<tr>
<td>PADAP</td>
<td>---</td>
<td>Address of the PADA</td>
</tr>
<tr>
<td>PADAL</td>
<td>---</td>
<td>Length of the PADA</td>
</tr>
<tr>
<td>PACCTRL</td>
<td>blank</td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>blank - Single space</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0       - Double space</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1       - Begin new page</td>
<td></td>
</tr>
</tbody>
</table>

Figure 98. Required PACA Fields for an Output Request
Returned Control-Area Data

When the Dictionary returns control to your program after processing the output request, the control area (PACA) field shown in Figure 99 contains information set by the Dictionary.

<table>
<thead>
<tr>
<th>PACA Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARC</td>
<td>Overall return code for the request</td>
</tr>
</tbody>
</table>

Figure 99. PACA Field Set by the Dictionary on Return from an Output Request

Because the output data contained in the data area (PADA) is assumed to be correctly formatted, the Dictionary ignores the format table (PADAFT) when it processes an output request.

Note: The COBOL CALL statement format for an output request requires three arguments, as does a retrieval request. However, the second argument (COBOL does not allow back-to-back commas in its CALL statements) is ignored by the Dictionary because the PADAFT is not used for output requests.

When you are using the WSTK call to add an entry to a Dictionary stack, you should consider the following rules for creating stack entries which can be used by other Dictionary functions:

- Each entry should contain a category name, followed by a subject name, separated by one or more blanks.

- Blanks may precede the category name, and the subject name can be followed by one or more blanks and, optionally, a comment.

- An '*' in the first entry position (first byte of the PADA) indicates a comment line in the Dictionary stack.

When you are generating a Dictionary stack, remember that the location of the stack is determined by the user before the program is invoked. Stack locations are declared, using the SETSTACK command, prior to invoking the program that issues WSTK calls. If it is necessary to generate several stacks, a program would have to be invoked once for each stack, with appropriate use of the SETSTACK command before each EXECUTE command.

The generated stack output is routed to a data set or to one set of User Data belonging to a particular subject. Part of the SETSTACK control enables the starting line and line number increment to be set for User Data. Your program will not be able to override these controls. Given a large increment, or entries added to an existing stack, the User Data limit of 999 lines may be reached. You can check for this condition (field PARC has a value of 120), so the program design can give the user a warning of a full stack condition.
When a stack is generated by a SCAN or STRUCTURE__REPORT command, a comment is added as the first entry in the stack. For example:

** GENERATED date time BY USER userid

You could use a similar comment within your first WSTK call, giving the program name (and version) instead of USER. If you want to add the userid, you could ask for this information to be included in the input parameter string.

** Return Code Table

One of the return codes shown in Figure 100 is set by the Dictionary upon return from an output request. “Appendix B.1. Overall Status of a Program Access CALL Statement” on page 408, describes each return code in detail.

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PACOK</td>
<td>Successful output operation</td>
</tr>
<tr>
<td>20</td>
<td>PACAUTH</td>
<td>Improper authorization for stack</td>
</tr>
<tr>
<td>24</td>
<td>PACCCTRL</td>
<td>Bad carriage control</td>
</tr>
<tr>
<td>84</td>
<td>PACDA</td>
<td>Bad PADA specification</td>
</tr>
<tr>
<td>92</td>
<td>PACTYPE</td>
<td>Bad request type</td>
</tr>
<tr>
<td>100</td>
<td>PACAINV</td>
<td>Bad PACA specification</td>
</tr>
<tr>
<td>104</td>
<td>PACERR</td>
<td>Unexpected condition</td>
</tr>
<tr>
<td>112</td>
<td>PACNOSTK</td>
<td>Stack location not predefined</td>
</tr>
<tr>
<td>116</td>
<td>PACIOSTK</td>
<td>I/O error for stack data set write</td>
</tr>
<tr>
<td>120</td>
<td>PACOVSSTK</td>
<td>Stack is full</td>
</tr>
<tr>
<td>124</td>
<td>PACNOSBJ</td>
<td>Stack location not found</td>
</tr>
<tr>
<td>128</td>
<td>PACCONFL</td>
<td>Request invalid in RUNSTACK processing</td>
</tr>
<tr>
<td>132</td>
<td>PACNORTE</td>
<td>Error during output routing</td>
</tr>
</tbody>
</table>

** Figure 100. Summary of Return Codes for an Output Request**
Coding a COBOL Program to Access Dictionary Data

The tasks involved in writing a COBOL program, summarized in "How to Write a Program to Access Dictionary Data" on page 212, are described in detail in this section.

Coding an Entry Point

DBDWUP is the name of the entry point the Dictionary uses to call user-written programs. The initial coding for a COBOL program PROCEDURE DIVISION can be:

PROCEDURE DIVISION.
SET-UP.
   ENTER LINKAGE.
   ENTRY "DBDWUP" USING DBDPAP.
   ENTER COBOL.

The name DBDPAP can be replaced by the name of the data declaration statement that defines the input parameter-string area.

Specifying a Parameter String Area

When the Dictionary passes control to your program, the parameter string is passed in a parameter area called DBDWPAP.

To make the DBDWPAP accessible to your program, you may code:

DATA DIVISION.
   LINKAGE SECTION.
   01 DBDWPAP COPY DBDWDUDC.

where DBDWDUDC is a copy file, provided by the Dictionary, that defines the parameter string.

The format of the DBDWPAP area is shown in Figure 101.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2 bytes</td>
<td>USER-RET-CODE</td>
<td>A return code to be set by your program. The initial value passed by the Dictionary is 0.</td>
</tr>
<tr>
<td>2</td>
<td>2 bytes</td>
<td>RESERVED</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>200 bytes</td>
<td>USER-STRING</td>
<td>The parameter string from the EXECUTE command. If the parameter string is less than 200 bytes, USER-STRING is padded on the right with blanks.</td>
</tr>
</tbody>
</table>

Figure 101. Parameter Area for Program Access (COBOL)
Specifying the Program Access Data Area and Control Blocks

To help you write your program, several segments of source code are supplied with the Dictionary. Each segment defines symbolic names for the control block fields and constant values needed in your program.

The source code segments for COBOL are available in the Dictionary macro library. Each segment contains COBOL data declarations for inclusion in COBOL programs as COPY code:

DBDWDCAC  Definition of the control area (PACA).
DBDWDFTC  Definition of the format table (PADAFT).
DBDWDUDC  Definition of the parameter-string area (DBDWPAP) passed to your program.
DBDWDVCAC Definitions of necessary initialized constants for COBOL.

To include these segments with your program, you can code:

DATA DIVISION.
  WORKING- STORAGE SECTION.
    01 PA-FIELD-VALUE-DEFNS COPY DBDWDVCAC.
    01 PADAFT COPY DBDWDFTC
      REPLACING DBDWFPTPA BY PADAFT.5
    01 PROCESS-PACA COPY DBDWDCAC
      REPLACING DBDWCAPA BY PROCESS-PACA.6
    01 DBDWPAP COPY DBDWDUDC.

Coding a CALL for Data Retrieval or Output

A request for retrieval or output of data is made with a CALL statement:

CALL 'CBLTDBD' USING pgm-paca, pgm-padaft, pgm-pada.

where "pgm-paca," "pgm-padaft," and "pgm-pada" are the names of a control area (PACA), format table (PADAFT), and data area (PADA) used with the request.

Note: The CALL statement for an output request must specify a format table (PADAFT) to satisfy the CALL statement's format requirements, although this parameter is ignored by the Dictionary.

5 The copy of the format table (PADAFT) in the Dictionary macro library is named DBDWFPTPA. This line renames it "PADAFT."

6 The copy of the control area (PACA) in the Dictionary macro library is named DBDWCAPA. This line renames it "PROCESS-PACA."
Returning to the Dictionary

When your program completes, it should set a return code to notify the Dictionary of either normal completion or an abnormal condition. The return code is set in the DBDWPAP parameter area, in field USER-RET-CODE. The code is interpreted by the Dictionary upon exit from your program.

If the return code set by your program is the highest that occurs during Dictionary execution, it will become the condition code for the job or job step. Your program is responsible for issuing an appropriate message, by using the output CALL statement described in “User-Program Output” on page 271.

For example, to set a return code and return to the Dictionary you can code:

MOVE URC-OK TO USER-RET-CODE.
GOBACK.

The return codes your program can set are shown in Figure 102.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>URC-OK</td>
<td>No errors detected which would affect further Dictionary execution of user input.</td>
</tr>
<tr>
<td>4</td>
<td>URC-WARNING</td>
<td>The user program wants the Dictionary to issue a warning message.</td>
</tr>
<tr>
<td>8</td>
<td>URC-ERROR</td>
<td>The user program has detected an error such that subsequent Dictionary commands should be flushed, if FLUSH=YES.</td>
</tr>
<tr>
<td>12</td>
<td>URC-SEVERE</td>
<td>The user program has detected a severe error situation. Subsequent Dictionary commands should be flushed, if FLUSH=YES.</td>
</tr>
<tr>
<td>16</td>
<td>URC-TERMINATING</td>
<td>The user program has detected an error situation that requires termination of the current execution of the Dictionary.</td>
</tr>
</tbody>
</table>

Figure 102. Return Codes Available for COBOL Program Access
Link-Editing the COBOL Program

Your COBOL user-written program must be link-edited into a library to which the Dictionary has access. This may be the library in which the Dictionary itself resides or a separate library that is defined in a STEPLIB or JOBLIB DD statement that is concatenated with the Dictionary execution JCL. It must be link-edited with the DDBWLNKC module, to allow the Dictionary to communicate with your program. The primary entry point of the resulting load module is DDBWLNKC.

The necessary link-edit control statements are shown below. “userpgmc” is the name of your COBOL program (that is, its object module). “userpgm” is the name of the resulting load module; it is the name specified with an EXECUTE command.

Link-edit control statements are:

//SYSLMOD DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=OLD
//LOADLIB DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=SHR
//OBJLIB DD (user parameters)
//SYSTR DD *
  INCLUDE LOADLIB(DBDBWLNKC)
  INCLUDE OBJLIB(userpgmc)
  ENTRY DDBWLNKC
  NAME userpgm(R)

The link-edit job’s JCL statements should include a DD statement that defines an object library (PDS) that contains your program (that is, its object module). The INCLUDE statement above uses a ddname of OBJLIB.

Note: The ENTRY statement forces the Dictionary linkage module to be the primary entry point of the resulting load module.

An Example of a COBOL Program

The Dictionary sample library includes a sample program, written in COBOL, that uses the Program Access facility. The sample program illustrates coding for the linkage, communication areas, and CALL statement format. Refer to “Appendix B.5. The COBOL Sample Program” on page 427 for a description of the DBDWSUPC program.

The COBOL sample program accesses the Dictionary to produce a list of the subjects related (in an upward direction) to a specified subject.

The sample program’s prolog describes its function, the EXECUTE command parameters, and the required compilation options.

You can obtain a listing of the sample program by executing the IEBPTPCH utility for the DBDWSUPC member of the sample library (a partitioned data set). Before you can invoke the COBOL sample program, you need to compile and link-edit it.
Coding a PL/I Program to Access Dictionary Data

The tasks involved in writing a PL/I program, summarized in “How to Write a Program to Access Dictionary Data” on page 212, are detailed in this section.

Coding an Entry Point

**DBDWUP** is the name of the entry point the Dictionary uses to call your program. The PL/I **PROCEDURE** statement can be coded:

```pli
DBDWUP: PROC(PAPPTR) OPTIONS(MAIN);
```

**PAPPTR** is the address of the input parameter-string area. **PAPPTR** is the required name if you use the source segments provided with the Dictionary. In PL/I, a substitute name could be used for **DBDWUP**, because the program is entered at the main entry point.

Specifying a Parameter String Area

When the Dictionary passes control to your program, the parameter string is passed in a parameter area called **DBDWPAP**.

To make the **DBDWPAP** accessible to your program, code:

```
%INCLUDE DBDWDUDP;
```

where **DBDWDUDP** is the name of the %**INCLUDE** code, provided by the Dictionary, that defines the parameter string.

The format of the **DBDWPAP** area is shown in Figure 103.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2 bytes</td>
<td>PAURC</td>
<td>A return code set by the user program. The initial value passed by the Dictionary is 0.</td>
</tr>
<tr>
<td>2</td>
<td>2 bytes</td>
<td>PARSV</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>200 bytes</td>
<td>PAUSTR</td>
<td>The parameter string from the EXECUTE command. If the character string is less than 200 bytes, <strong>PAUSTR</strong> is padded on the right with blanks.</td>
</tr>
</tbody>
</table>

**Figure 103.  Parameter Area for Program Access (PL/I)**
Specifying the Program Access Data Area and Control Blocks

To help you write your program, several segments of source code are supplied with the Dictionary. Each segment defines symbolic names for the control block fields and constant values needed in your program.

The source code segments for PL/I are available in the Dictionary macro library. Each segment contains PL/I DECLARE statements for inclusion in PL/I programs as %INCLUDE code:

- **DBDWDCAP**: Definition of the control area (PACA). DBDWDCAP bases the control area (PACA) on a pointer variable.
- **DBDWDFTP**: Definition of the format table (PADAFT).
- **DBDWDUDP**: Definition of the parameter-string area (DBDWPAP) passed to your program.
- **DBDWDVAP**: Definition of necessary initialized constants for PL/I.

To include these segments with your program, you can code:

```
%INCLUDE DBDWDCAP;
%INCLUDE DBDWDFTP;
%INCLUDE DBDWDUDP;
%INCLUDE DBDWDVAP;
```

In the %INCLUDE code the Dictionary provides, the control area (PACA) and the format table (PADAFT) are declared as based variables. Your program must define space for the PACAs and PADAFTs it needs and base them on that space. For example, if your program will use two PACAs (one for retrieval requests and one for output requests) and one PADAFT with 10 entries, you might code:

```
ALLOCATE DBWCAPA SET (PACAPTR);
ALLOCATE DBWCAPA SET (PRT);
DECLARE PADAFT1 CHAR (252);
PAPTR=ADDR (PADAFT1);
```

Because the format of a data area (PADA) is entirely up to the user, the Dictionary doesn't provide %INCLUDE code for it. To define a PADA 1000 bytes long, you could code:

```
DECLARE PADA1 CHAR (1000);
```
Coding a CALL for Data Retrieval or Output

A request for retrieval or output of data is made with the CALL statement:

```
CALL PLITDBD(pgm_paca);
```

or

```
CALL PLITDBD(pgm_paca_pointer);
```

"pgm_paca" names the control area (PACA) for this request, and is a based variable. Alternatively, the CALL can specify a pointer variable ("pgm_paca_pointer").

Returning to the Dictionary

When your program is completed, it should set a return code to notify the Dictionary of either normal completion or an abnormal condition. The return code is set in the DBDWPAP parameter area, in PAURC. The code is interpreted by the Dictionary upon exit from your program.

If the return code set by your program is the highest that occurs during Dictionary execution, it will become the condition code for the job or job step. Your program is also responsible for issuing an appropriate message, by using the output CALL statement described in “User-Program Output” on page 271.

The following code is an example of setting a return code and returning to the Dictionary. It assumes that you have established your control blocks as specified in the examples shown in “Specifying the Program Access Data Area and Control Blocks” on page 279.

```
/* SET OK RETURN CODE */
PAURC = URCOK;
/* FREE ALLOCATED SPACE */
FREE PACAPTR->DBDWCAPA;
FREE PTR->DBDWCAPA;
/* RETURN TO DICTIONARY */
RETURN;
```
The return codes your program can set are shown in Figure 104.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>URCOK</td>
<td>No errors detected which would affect further Dictionary execution of user input.</td>
</tr>
<tr>
<td>4</td>
<td>URCWARN</td>
<td>The user program wants the Dictionary to issue a warning message.</td>
</tr>
<tr>
<td>8</td>
<td>URCEERROR</td>
<td>The user program has detected an error requiring that subsequent Dictionary commands should be flushed, if FLUSH=YES.</td>
</tr>
<tr>
<td>12</td>
<td>URCSVERE</td>
<td>The user program has detected a severe error situation. Subsequent Dictionary commands should be flushed, if FLUSH=YES.</td>
</tr>
<tr>
<td>16</td>
<td>URCTERM</td>
<td>The user program has detected an error situation that requires termination of the current execution of the Dictionary.</td>
</tr>
</tbody>
</table>

Figure 104. Return Codes Available for PL/I Program Access

Link-Editing the PL/I Program

Your PL/I user-written program must be link-edited into a library to which the Dictionary has access. This may be the library in which the Dictionary itself resides or a separate library that is defined in a STEPLIB or JOBLIB DD statement that is concatenated with the Dictionary execution JCL. It must be link-edited with the DBDWLNKPK module, to allow the Dictionary to communicate with your program. The primary entry point of the resulting load module is DBDWLNKPK. When the Dictionary passes control to your program, the environment is that required for a PL/I program.

The necessary link-edit control statements are shown below. “userpgmp” is the name of your PL/I program (that is, its object module). “userpgm” is the name of the resulting load module; it is the name specified in an EXECUTE command.
Link-edit control statements are:

```plaintext
//SYSLMOD DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=OLD
//LOADLIB DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=SHR
//OBJLIB DD (user parameters)
//SYSIN DD *
   INCLUDE LOADLIB(DBDWSLNK)
   INCLUDE OBJLIB(userpgm)
   ENTRY DBDWSLNK
   NAME userpgm(R)
```

The link-edit job step's JCL statements should include a DD statement that defines an object library (PDS) for the user-written programs (object modules). The INCLUDE statement above uses a ddname of OBJLIB.

*Note:* The ENTRY statement forces the Dictionary linkage module to be the primary entry point of the resulting load module.

### An Example of a PL/I Program

The Dictionary sample library includes a sample program, written in PL/I, that uses the Program Access facility. The sample program illustrates coding for the linkage, communication areas, and CALL statement format.

The PL/I sample program accesses the Dictionary to produce one or more reports about a specified category (either standard or extensibility):

- All aliases of the category name
- Description text
- The category's definition
- A list of valid attributes
- Explanation text (from user data segments)
- Relationship-types and, optionally, relationship attributes
- A list of subjects in the category and, optionally, their attributes

The sample program's prolog describes its function, the `EXECUTE` command parameters, and the required compilation options.

You can obtain a listing of the PL/I sample program by executing the `IEBPTPCH` utility for the DBDWSUPP member of the sample library (a partitioned data set). Before you can invoke the PL/I sample program, you need to compile and link-edit it.
Coding an Assembler Language Program to Access Dictionary Data

The tasks involved in writing an Assembler language program, summarized in "How to Write a Program to Access Dictionary Data" on page 212, are described in detail in this section.

Coding an Entry Point

DBDWUP is the name of the entry point the Dictionary uses to call your program. The ENTRY statement for an Assembler language program is coded:

ENTRY DBDWUP

The initial coding for an Assembler language program is:

```
DBDWUP CSECT
STM R14,R12,12(R13)  ; Save registers
< establish base registers >
```

Specifying a Parameter String Area

When the Dictionary passes control to your program, the parameter string is passed in a parameter area called DBDWPAP. Register 1 contains the address of the input parameter-string area.

To make the DBDWPAP available to your program, code:

```
L REG3,0(REG1)  ; REG3 contains address of DBDWPAP
USING DBDWPAP,REG3
```

The format of the DBDWPAP area is shown in Figure 105.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2 bytes</td>
<td>PAURC</td>
<td>A return code set by the user program. The initial value passed by the Dictionary is 0.</td>
</tr>
<tr>
<td>2</td>
<td>2 bytes</td>
<td>PARSV</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>200 bytes</td>
<td>PAUSTR</td>
<td>The parameter string from the EXECUTE command. If the parameter string is less than 200 bytes, PAUSTR is padded on the right with blanks.</td>
</tr>
</tbody>
</table>

Figure 105. Parameter Area for Program Access (Assembler)

---

7 REG1 is general register 1. It contains the address of the DBDWPAP area.

8 REG3 is the base register, general register 3. You can code any register (except register 0 or 1) instead of register 3.
Specifying the Program Access Data Area and Control Blocks

To help you write your program, several segments of source code are supplied with the Dictionary. Each segment defines symbolic names for the control block fields and constant values needed in your program.

The source code segments for Assembler language are available in the Dictionary macro library. Each segment contains source code whose invocation can be coded to generate a DSECT or an actual initialized control block. (A summary of the macro parameters is in “Appendix B.7. Program Access Macro Support” on page 458.)

**DBDWDCA** Definition of the control area (PACA).

**DBDWDFT** Definition of the format table (PADAFT).

**DBDWDUDA** Definition of the parameter-string area (DBDWPAP) passed to your program. (This is a DSECT for inclusion in an Assembler language program as COPY code.)

Definitions of necessary initialized constants can be generated for Assembler programs by using the DBDWDCA and DBDWDFT macros, as described in “Appendix B.7. Program Access Macro Support” on page 458.

To include these segments with your program, you can code:

- For the control area (PACA):
  
  ```
  DBDWDCA TYPE=WTRM, PREFIX=00, ...
  DBDWDCA TYPE=RSA, PREFIX=PP, FTP=CATPADFT, ...
  ```

  **Note:** The name of a control area (PACA) is specified with the PREFIX parameter (which results in a suffix appended to the PACA name and a prefix appended to all PACA field names). In this example, the PACA names are DBDWCAD00 and DBDWCAPP and all PACA field names are prefixed, respectively, with “00” and “PP.” If the PREFIX parameter is not specified, the PACA name defaults to DBDWC.

  The DBDWDCA macro is described in “Appendix B.7. Program Access Macro Support.”

- For the format table (PADAFT):
  
  ```
  DBDWDFT TYPE=DSECT, PREFIX=DD, ...
  ```

  The DBDWDFT macro is described in Appendix B.7. “Program Access Macro Support.”

- For the parameter string area (DBDWPAP):
  
  ```
  COPY DBDWDUDA
  ```
Coding a CALL for Data Retrieval or Output

A request for retrieval or output of data is issued with the CALL statement:

CALL ASMTDBD,pgm=paca

“pgm__paca” names the control area (PACA) for this request.

Returning to the Dictionary

When your program completes, it should set a return code to notify the Dictionary of either normal completion or an abnormal condition. The return code is set in the DBDWPAP parameter area, in field PAURC. The code is interpreted by the Dictionary upon exit from your program.

If the return code set by your program is the highest that occurs during Dictionary execution, it will become the condition code for the job or job step. Your program is responsible for issuing an appropriate message, by using the output CALL statement described in “User-Program Output” on page 271.

To set an OK return code, code:

LA R14,URCOK
STH R14,PAURC

To return to the Dictionary, code:

SLR R15,R15
L R13,4(R13) R13 is save area address
L R14,12(R13) R14 is return address
LM R0,R12,20(R13) Restore registers
BR R14 Return

The return codes your program can set are shown in Figure 106 on page 286.
<table>
<thead>
<tr>
<th>Return Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>URCOK</td>
<td>No errors detected which would affect further Dictionary execution of user input.</td>
</tr>
<tr>
<td>4</td>
<td>URCWARN</td>
<td>The user program wants the Dictionary to issue a warning message.</td>
</tr>
<tr>
<td>8</td>
<td>URCErrOR</td>
<td>The user program has detected an error such that subsequent Dictionary commands should be flushed, if FLUSH=YES.</td>
</tr>
<tr>
<td>12</td>
<td>URCSVERE</td>
<td>The user program has detected a severe error situation. Subsequent Dictionary commands should be flushed, if FLUSH=YES.</td>
</tr>
<tr>
<td>16</td>
<td>URCTERM</td>
<td>The user program has detected an error situation that requires termination of the current execution of the Dictionary.</td>
</tr>
</tbody>
</table>

Figure 106. Return Codes Available for Assembler Program Access

Link-Editing the Assembler Language Program

Your Assembler language user-written program must be link-edited into a library to which the Dictionary has access. This may be the library in which the Dictionary itself resides or a separate library that is defined in a STEPLIB or JOBLIB DD statement and concatenated with the Dictionary execution JCL. It must be link-edited with the DBDWLCLKA module, to allow the Dictionary to communicate with your program. The primary entry point of the resulting load module is DBDWLCLKA.

The necessary link-edit control statements are given below. "userpgma" is the name of your program (object module). "userpgm" is the name of the resulting load module; it is the name specified in an EXECUTE command.

Link-edit control statements are:

```
//SYSLMOD DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=OLD
//LOADLIB DD DSN=DBD60.IMSVS.DDPGMLIB,DISP=SHR
//OBJLIB DD (user parameters)
//SYSIN   DD *
       INCLUDE LOADLIB(DBDWLCLKA)
       INCLUDE OBJLIB(userpgma)
       ENTRY DBDWLCLKA
       NAME userpgm(R)
```
The link-edit job step's JCL statements should include a DD statement that defines an object library (PDS) for your program (object module). The INCLUDE statement above uses a ddname of OBJLIB.

*Note:* The ENTRY statement forces the Dictionary linkage module to be the primary entry point of the resulting load module.

**An Example of an Assembler Language Program**

The Dictionary sample library includes a sample program, written in Assembler language, that uses the Program Access facility. The sample program illustrates coding for the linkage, communication areas, and CALL statement format.

The Assembler language sample program interacts with the Dictionary to produce a report of the amount of virtual storage required for the master PADAFT and PADA of a specified category or relationship-type.

The sample program's prolog describes its function, the EXECUTE command parameters, and the required Assembler options.

You can obtain a listing of the Assembler language sample program by executing the IEBPTPCH utility for the DBDWSUPA member of the sample library (a partitioned data set). Before you can invoke the Assembler-language sample program, however, you need to compile and link-edit it.

**Documenting Your Program Access Program**

This section does not contain any rules about the form you use to document your program, or about the type and amount of information you include, but it does give some guidelines about the type of program documentation that is valuable. Installations establish standards in this area; make sure you are aware of the standards at your installation.

When you document your program, you record information about the program for two reasons:

- **Documentation for Other Programmers**

  One type of information you record is information about structuring and coding the program. You record this information for future use by other programmers who will maintain the program you've written.

- **Documentation for Users**

  The other type of information you should record is information for the people who use your program.
Documentation for Other Programmers

Documenting a program is not something you should wait until the end of the project to do; your documentation will be much more complete, and more useful to others, if you record information about the program as you structure and code it. Include any information you think may be useful to someone else who has to work with your program.

The reason you record this information is so that people who maintain your program will know something about why you chose certain PACA and PADAFT parameters, retrieval sequences, and PADA structures. For example, if the data base administrator was considering developing a report of some set of Dictionary data, information about why your program accesses the Dictionary the way it does would be helpful.

The best place to record information about your program is in the Data Dictionary itself. It is then available to anyone who might need it, and it is easy to maintain.

Some of the information you should include for other programmers is:

- Flowcharts and pseudocode for the program
- Comments about the program from code inspections
- A written description of the program flow
- Information about why you chose the PACA and PADAFT parameters you did, such as:
  - Which positioning parameters does your program use?
  - Which return codes are checked for when a CALL completes?
  - If the PACA is reused by several CALLs, which fields are reset before each CALL?
- Any problems you encountered in structuring or coding the program
- Any problems you had when you tested the program
- Warnings about what should not be changed in the program and why

All this information relates to structuring and coding the program. In addition, you should include the information described in "Documentation for Users" on page 289 with the documentation for programmers.

Again, the amount of information you include and the form in which you document it depend on you and your installation. These are provided as suggestions.
Documentation for Users

All the information listed under "Documentation for Other Programmers" on page 288 relates to the design of the program. In addition to this, you should record information about how you use the program. The amount of information that users need, and how much of it you should supply, depend on who the users of the program are and what type of program it is.

At a minimum, you should include this information for users of your program:

- What the user needs to use the program
  - For online programs, the name of the program (PGM=pgmname)
  - For batch programs, the required JCL
- The input users need to supply for the program
  - If a parameter list is required, the format your program expects (PARM='parameter string')
- The content and form of the program's output
  - If it's a report, show the format or include a sample listing.
  - For an online program, show what the screen will look like.
  - For all programs, describe each return code and error message your program produces.

If users of your program are unfamiliar with terminals, they will also need some type of a terminal user's guide. For example, this guide should give explicit instructions on how to use the terminal and what users can expect from the program. Although you may not be responsible for providing this type of information, you should provide any terminal use information that is unique to your application to the person responsible for this information.
Chapter 13. Administering the Dictionary Language Preprocessor

The purpose of this chapter is to:

- Explain how the Dictionary Language Preprocessor is used and how it can extract data from Dictionary data bases
- Supply information that will help you when planning to make the Language Preprocessor available for groups of end users
- Provide reference information for Language Preprocessor initialization and for coding optional user exit routines.
- Explain how to prepare the Dictionary so that it can respond to Language Preprocessor requests for data extraction
- Give reference information that needs to be conveyed to end users

This chapter assumes that you understand how COBOL, PL/I, or Assembler data structures are described in the Dictionary and formatted by the STRUCTURES__OUT command. For this information, see Chapter 14 in OS/VS DB/DC Data Dictionary Applications Guide.

The Language Preprocessor modifies a source program by recognizing and expanding inclusion statements (COBOL and Assembler COPY and PL/I INCLUDE statements) within the source program. As each inclusion statement is encountered, language data structures are obtained directly from the Dictionary data bases and embedded within the source program. Source programs written in COBOL, PL/I, or Assembler are accepted as input.

The Language Preprocessor does require a set of input operands that accompany the source program input, but most, if not all, of these input operands can be defaulted. DBCS users must use a uniform string for the category name default option.

The output of the Language Preprocessor, the modified source program, is appropriate for input to the host language compiler (or Assembler). The initial source input is assumed to be a "single" source program—CBL statements (COBOL) and PROCESS statements (PL/I) are not recognized, so that no segmenting of the output occurs.
Advantages of Language Preprocessor Use

The Language Preprocessor offers several advantages over alternative methods. Without the use of the Language Preprocessor, source code for the data structure is obtained from the Dictionary through the STRUCTURES_OUT command. The data structure is then placed in the COPY/INCLUDE library. A source program would then reference that member of the COPY/INCLUDE library with an inclusion statement (COPY or %INCLUDE statement). This method requires several steps to be followed and requires that the COPY/INCLUDE libraries be updated whenever the Dictionary data structures are updated. This method also keeps redundant information since the information in the COPY/INCLUDE libraries is obtainable from the Dictionary.

With the use of the Language Preprocessor, inclusion statements in a source program are recognized and the source program is modified accordingly. The Language Preprocessor performs the following steps:

1. Recognizes the inclusion statement in the source program and interprets it to determine what source code should be retrieved from the Dictionary.

2. Retrieves the desired source code by invoking the Dictionary and submitting commands to it.

3. Inserts the retrieved source code into the source program at the place in which the inclusion statement was found.

4. Converts the inclusion statement into a comment.

This method eliminates the need for a COPY/INCLUDE library and keeps the data structure current by retrieving the latest data structures from the Dictionary.

The Dictionary Language Preprocessor is a program that is provided as part of the Dictionary system. Although its use is optional, it can provide a valuable service to your installation. It can be an integral part of the development process and it contributes to programmer productivity. Language data structures that have been stored in the Dictionary data bases can be made available to a variety of end users without the need of Dictionary education. With appropriate preparation, source language data structures stored within the Dictionary can be used by application developers when they are compiling and testing programs.

Obtaining a language data structure from the Dictionary helps ensure that the structure meets standards and is the approved version. In fact, with appropriate preparation of the Dictionary, you can ensure that the current version is automatically used without the need for any alteration of the source program.

Making use of the Language Preprocessor does not preclude use of any existing source libraries (such as COPY/INCLUDE libraries) and does not require specialized source coding. The Language Preprocessor accepts as input a COBOL, PL/I, or Assembler source program. Source statements, in the host language, are recognized by the Dictionary Language Preprocessor as it scans the source program. The inclusion statements are simple in nature and familiar in their syntax.
Defining Groups of Users

The first task in making the Language Preprocessor productive is to identify groups of users who can benefit from its use. Things to consider are:

- The source language the group will use
- The level of flexibility in specifying Language Preprocessor control members of the group
- Getting agreement that the Dictionary Language Preprocessor will be part of their usual development activities

Of special interest is how the Dictionary will work with existing or parallel COPY/INCLUDE libraries. It should be emphasized that these libraries are not precluded when the Language Preprocessor is used. The COPY/INCLUDE library can be used to gradually migrate users to the Dictionary as the only source for language data structures commonly used by programs in development.

How the Language Preprocessor Works with the Dictionary

This section will help you understand the various aspects of planning for Language Preprocessor use. It explains the mechanism by which source statements are interpreted, the operating environment, and the Language Preprocessor initialization process.

Dictionary support for the Language Preprocessor uses the command list facility. You can become familiar with this support by reviewing the command list section of Chapter 7 in OS/VS DB/DC Data Dictionary Applications Guide. Also, for information on the inclusion statements, see “How Inclusion Statements Invoke Dictionary Processing” on page 296.

The Language Preprocessor Processing Environment

Figure 107 on page 295 illustrates how the Language Preprocessor works with the host language translator and the Dictionary. Refer to the boxed numbers in the figure:

1. The Language Preprocessor accepts as input a COBOL, PL/I, or Assembler source program from the SYSIN data set, together with a set of input operands contained in a Default Options Table. The Default Options Table can be overridden by operands specified in the PARM keyword of the JCL EXEC statement.

2. The Language Preprocessor performs one pass over SYSIN. If a statement is not recognized as an inclusion statement, it is written to the SYSPUNCH data set. See “How Inclusion Statements Invoke Dictionary Processing” on page 296.
3. Each inclusion statement causes a request to the Dictionary, specified as a set of commands entered into the DDINPUT data set. One of these commands (SETCLIST) identifies a Dictionary subject containing a command list. The Language Preprocessor derives the name of this subject from a combination of information from the inclusion statement and input operands. The subject name of the command list used by the Language Preprocessor cannot contain DBCS characters.

4. The Dictionary executes the input commands. One of these commands (RUNCLIST) causes the execution of the commands in the command list identified by the SETCLIST.

5. If the commands produce any source output, it is written into the DDPUNCH data set. Dictionary messages are written to the DDLIST data set.

6. Each record in the DDPUNCH data set is read by the Language Preprocessor. Optionally, each record is passed to an installation-written exit routine and the record is passed back to the Language Preprocessor.

7. The DDPUNCH records, with possible changes provided by the user exit, are written to the SYSPUNCH data set by the Language Preprocessor—replacing inclusion statements with source code from the Dictionary.

8. In addition, a listing is generated in the SYSPRINT data set. This contains Dictionary messages present in DDLIST, messages originating from the Language Preprocessor, and, optionally, contains a listing of the original input data in SYSIN.

9. The resulting program, contained in the SYSPUNCH data set, becomes input to the host language compiler or Assembler.
Figure 107. Components of Dictionary Preprocessing
How Inclusion Statements Invoke Dictionary Processing

Statements in the host language that are recognized by the Dictionary Language Preprocessor are forms of an inclusion statement (COBOL and Assembler COPY and PL/I INCLUDE statements). Such a statement contains a member name and, optionally, a library name. The following are examples of inclusion statements, where “PAYSEG” is a member name and “LIB1” is a library name.

**COBOL:**

COPY PAYSEG.
COPY PAYSEG OF LIB1.

**PL/I:**

`$INCLUDE PAYSEG;`
`$INCLUDE LIB1(PAYSEG);`

**Assembler:**

COPY PAYSEG

Because inclusion statements are valid host language inclusion statements, the host language compiler (or Assembler) translates them as such. That is, the library name is used to determine the name of the COPY/INCLUDE library (SYSLIB is used if the library name is not specified) and the member name is used to determine the member within the COPY/INCLUDE library. The source code stored in this member is retrieved and inserted into the source program at the location in which the inclusion statement was found.

The translation of an inclusion statement by the Language Preprocessor differs from this in that the member name and the library name have entirely different usages. They are used to locate a Dictionary subject. Within this subject’s definition are instructions that specify what source code is to be retrieved from the Dictionary (as opposed to identifying a single data set source for code to be retrieved from COPY/INCLUDE libraries).

Translating the inclusion statement into actual output is accomplished as follows:

1. The library name, if present, is concatenated in front of the member name, the two being separated by a special character (the special character is defined in the Default Options Table). This determines the *user name* of a Dictionary subject. Assuming that the special character is a period, from the inclusion statement,

```
COPY PAYSEG OF LIB1.
```

the username “LIB1.PAYSEG” is obtained. Note that, as the content of these inclusion statements varies—differing member names and, possibly, differing library names and different Dictionary user names are obtained.

*Note:* If the inclusion statement does not contain a library name, a default name is applied. This default can be null so that the resulting Dictionary name consists of the member name only.

2. From the input operands, the category, status, code, and occurrence of the Dictionary subject are obtained, which, together with the user name, uniquely identify the Dictionary subject. The input operands are obtained from the Default Options Table, which can be overridden by operands specified in the
PARM keyword of the EXEC statement. Suppose the category is SEGMENT and the other subject name qualifiers are defaulted. Then the Dictionary subject referenced by the above inclusion statement could be:

\[
\text{SEGMENT (T,C,LIB1,PAYSEG,0)}
\]

3. The Language Preprocessor determines from the input operands:

- A set of User Data (belonging to the subject identified through items 1 and 2 above), which contains a Dictionary command list

- The line number at which the Dictionary command list begins

For example, the Dictionary command list could begin in line 50 of the first set of User Data. It is the responsibility of the installation to establish the command list.

4. The Language Preprocessor generates two commands (a \text{SETCLIST} command and a \text{RUNCLLIST} command), invokes the Dictionary, and submits these two commands. The \text{SETCLIST} command identifies the location of a Dictionary command list contained in the set of User Data identified through item 3 above. Although, in general, Dictionary command lists can contain any Dictionary commands, when executed by the Language Preprocessor, only the \text{STRUCTURES\_OUT} and PUNCH commands are allowed. The \text{RUNCLLIST} command causes the Dictionary to execute those commands. It is the output from the execution of these \text{STRUCTURES\_OUT} and/or PUNCH commands that makes up the source code to be inserted by the Language Preprocessor into the source program at the location in which the inclusion statement was found.

In the example, the source code that the Language Preprocessor retrieves from the Dictionary is obtained as command output from executing a set of Dictionary commands that start in line 50 of the first set of User Data for the Dictionary subject, SEG (T,C,LIB1,PAYSEG,0). The \text{SETCLIST} command that specifies this would be:

\[
\text{SETCLIST SEG (T,C,LIB1,PAYSEG,0) UDNO=1 LINENO=50 ;}
\]

In addition to inserting the retrieved source code into the source program, the Language Preprocessor transforms the inclusion statement into a comment so that it will not be recognized and translated by a subsequent compiler or assembler. There is no further scan of the retrieved source code. The retrieved source code comes from the output of commands contained in the CLIST.

If the subject named in the \text{SETCLIST} command submitted by the Language Preprocessor does not exist in the Dictionary, two options are available to the user:

- The Language Preprocessor transforms the inclusion statement into a comment. With this option, the statement will not be recognized and translated by a subsequent compiler or assembler.

- The Language Preprocessor leaves the inclusion statement unchanged. With this option, the statement is recognized and processed by a subsequent compiler or assembler. Consequently, data not contained in the Dictionary can still be obtained from COPY/INCLUDE libraries.

Chapter 13. Administering the Dictionary Language Preprocessor 297
How Data Is Selected from the Dictionary

This section explains the processing invoked when an inclusion statement is recognized during the scan of the source program input by the Language Preprocessor. You need to know about the various parts of this process to enable you to:

- Understand the Language Preprocessor support and what kind of preparations have to be made
- Decide what controls you will put in effect for the Language Preprocessor use
- Plan your ongoing support for the end users

The Role of Command Lists in Language Preprocessor Use

A set of Dictionary commands stored in a set of User Data belonging to a Dictionary subject is called a command list. The Dictionary command, RUNCLIST, causes the commands in the identified command list to be executed. The location of a command list is identified by specifying its category, subject name, User Data number and starting line number, using the Dictionary command, SETCLIST. This command must be executed before execution of the RUNCLIST command.

The Language Preprocessor uses the SETCLIST and RUNCLIST commands to obtain the data from the Dictionary for insertion into the source program. For example, the Language Preprocessor submits the following commands to the Dictionary to cause the execution of the commands in the command list stored in User Data 1 for Dictionary subject, SEG (T,C,LIB1.PAYSEG,0), starting with line number 50.

```
SETCLIST SEG (T,C,LIB1.PAYSEG,0) UDNO=1 LINENO=50;
RUNCLIST STOPON=E;
```

There are several differences in the way the Language Preprocessor uses command lists compared to the general capabilities described in Chapter 7 of *OS/VS DB/DC Data Dictionary Applications Guide*. The principal difference is that, when invoked by the Language Preprocessor, only the STRUCTURES__OUT and PUNCH commands are permitted to be present in the command list. Typically, a command list would contain a single STRUCTURES__OUT command, although there is no restriction on the number of commands or their sequence within the command list—given that they conform to command list conventions.

The other major difference concerns the use of status codes. The Language Preprocessor uses a technique in which the status code of a subject whose User Data contains a command list, or of a subject in the command list, is not explicitly declared. Instead, a list of status codes is specified on the SETCLIST command and each status code in the list is substituted one by one as the subject name qualifier until the Dictionary finds a matching existing subject. A status code not explicitly specified is represented by an ampersand symbol (&) and is termed an indefinite status code. The use of this technique is explained in “Using Indefinite Status Codes” on page 323.
How the Language Preprocessor Requests Output from the Dictionary

Upon recognition of an inclusion statement, the Language Preprocessor places several commands in DDINPUT. These commands, in the order generated, are:

**SIGN_ON**
This command is entered if the Language Preprocessor defaults required the use of access control in the Dictionary servicing the preprocessor requests.

**SETLOGI=Y**
This command is entered so that each command that is executed within the Dictionary (in response to a Language Preprocessor request) is recorded in the DDLIST data set.

**SETCLIST**
This command identifies the location of the command list.

**RUNCLIST**
This command invokes the execution of each STRUCTURES_OUT and PUNCH command stored in the designated set of User Data.

The Dictionary is invoked and executes the four commands, producing output in the DDPUNCH data set, and a listing in the DDLIST data set. All four commands are automatically generated by the Language Preprocessor. The Language Preprocessor user does not specify them. The commands in the command list must, however, be supplied by the installation before invoking the Language Preprocessor. When the Language Preprocessor executes, the commands in the command list must be already entered and stored in appropriate User Data lines.

If this series of commands produces no output in the DDPUNCH data set, this is interpreted by the Language Preprocessor as a "not found" condition and the inclusion statement is passed unaltered to the SYSPUNCH data set. This condition can occur in several ways:

- The subject specified on the SETCLIST command is not found, or,
- If found, its User Data does not exist, or
- An error condition is detected by RUNCLIST processing, or
- The result of the RUNCLIST command produces no DDPUNCH output.

How the Language Preprocessor Interprets Dictionary Processing

The Language Preprocessor invokes the Dictionary for every inclusion statement. If the Dictionary subject name is found, the command list will be executed.

The return code issued by the Dictionary after executing the commands in the command list determines what action is to be taken by the Language Preprocessor. If the return code is 4 or less, the execution is considered successful by the Language Preprocessor. The source code retrieved from the Dictionary is inserted into the source program at the location in which the inclusion statement was found. In addition, the Language Preprocessor statement is transformed into a comment to prevent its translation by a subsequent processor. Note that if the inserted source code contains any Language Preprocessor statements, the Language Preprocessor...
will not translate these. "Nesting" of inclusion statements can be handled only by successive executions of the Language Preprocessor.

If the return code exceeds 4, then something went wrong—either the command list was not found, or some other error occurred. In this situation, the SEARCHRULE input operand determines what action is to be taken by the Language Preprocessor. This operand can be fixed for general use of the Language Preprocessor or can be specified as a JCL input operand, on the Language Preprocessor EXEC statement. The purpose of this operand is to declare how "not found" conditions are to be interpreted.

The SEARCHRULE operand is used to tell the Language Preprocessor whether the data structures are expected to be contained only in the Dictionary or a mixture of some in the Dictionary and some in COPY/INCLUDE libraries is expected. If all data structures are expected to be in the Dictionary only, then you should specify SEARCHRULE=DICONLY. If you expect the data structures to be in mixed mode, then you should specify SEARCHRULE=BOTH.

If the return code exceeds 4 and SEARCHRULE is DICONLY, the Language Preprocessor transforms the inclusion statement into a comment. In this way, it will not be recognized and translated by the host language compiler or Assembler in the subsequent step. With DICONLY, it is assumed that all data structures are in the Dictionary and no data structures should be retrieved from a COPY/INCLUDE library.

If the return code exceeds 4 and SEARCHRULE is BOTH, the inclusion statement Language Preprocessor statement is left unchanged by the Language Preprocessor and no source code from the Dictionary gets inserted. Because the inclusion statement is a valid host language inclusion statement, it will be translated by the host language compiler (or Assembler). In this way, a method of concatenating a COPY/INCLUDE library after the Dictionary is achieved and this allows a single source program to obtain source code not only from the Dictionary but also from COPY/INCLUDE libraries.

The Role of User Exits

You can specify a user exit routine, which is called after each call to the Dictionary and results in "good" output (indicated by a Dictionary return code of 4 or less). Thus, no call will be made if the command list was not found or if some other error occurred. The exit will be loaded only once, so it must be reusable. The call will be made with standard OS/VS register usage. The Language Preprocessor will retrieve records from DDPUNCH and pass them to the user exit routine one at a time. The user exit routine may modify these lines in any manner, but may not delete or add lines.
Your installation can assess the value of using such an exit routine. Some of the ways they can be used are:

- To ensure appropriate indentation of levels within language data structures
- To change field names; insert a prefix or substitute for a given character string
- To scan the data structure content; detect the use of a particular field in a data structure or the absence of comments

Planning information and programming considerations for user exits are discussed in “Incorporating a User Exit into the Language Preprocessor Execution” on page 317.

What Kind of Output Is Produced for the End User

There are several types of end user output produced as a result of Language Preprocessor processing:

- In the SYSPUNCH data set, the modified source program
- In the SYSPRINT data set, Language Preprocessor diagnostics and an optional listing of the SYSIN input
- In the DDLIST data set, Dictionary diagnostics—a user option is to include some of these within the SYSPRINT output

Source Program Output

The output file, SYSPUNCH, contains the modified version of source program input, which results from the translation of the inclusion statements within SYSIN. If Dictionary command output is inserted into the source program, the inclusion statement is converted to a comment. Source code obtained from the Dictionary and inserted into the source program is not modified by the Language Preprocessor in any way. This implies that the sequence number field, identification field, continuation column, and starting column for a source language statement on each line are copied unchanged to SYSPUNCH by the Language Preprocessor.

Figure 108 shows how input to the Language Preprocessor is expanded with a data structure extracted from the Dictionary:

- Input: a portion of a COBOL source program is shown. This is part of the SYSIN data set; the arrow points to an inclusion statement.
- Output: the same portion of the COBOL source is shown ready for compiler processing; the inclusion statement has been expanded.
SOURCE PROGRAM INPUT IN SYSIN:

```
DATA DIVISION.
FILE SECTION.
FD INDEXED FILE LABEL RECORDS STANDARD.
--->
COPY DISKREC.
FD IN FILE LABEL RECORDS STANDARD.
  01 IN-RECORD.
    02 IN-KEY   PICTURE X(10).
    02 IN-NAME  PICTURE X(10).
    02 IN-BAL   PICTURE S99999V99.
WORKING-STORAGE SECTION.
```

SOURCE PROGRAM PASSED TO COMPILER IN SYSPUNCH

```
DATA DIVISION.
FILE SECTION.
FD INDEXED FILE LABEL RECORDS STANDARD.
--->
COPY DISKREC.
FD IN FILE LABEL RECORDS STANDARD.
  01 IN-RECORD.
    02 IN-KEY   PICTURE X(10).
    02 IN-NAME  PICTURE X(10).
    02 IN-BAL   PICTURE S99999V99.
WORKING-STORAGE SECTION.
```

Figure 108. Portion of a COBOL Source Program with an Inclusion Statement Expanded

Diagnostic Output

The primary purpose of the SYSPRINT output is to inform the user of the conditions that have been detected during the physical scan of the SYSIN input data set. The Language Preprocessor reports problems encountered during its processing. Details of specific inclusion statement syntax recognized by the Language Preprocessor are given for each source language in the section “Giving Information about Code Requirements” on page 331.

Another source of messages is the Dictionary itself. By using the DDLIST input operand, these messages can be included in the SYSPRINT data set. There is a
series of messages for each inclusion statement and they are inserted immediately after each statement.

If there are problems regarding access to the Dictionary program, data set error conditions, or a failure during Language Preprocessor initialization, these errors are also detected and processing is terminated.

Both Language Preprocessor messages and messages issued by the Dictionary are documented in OS/VS DB/DC Data Dictionary Messages and Codes. You need to make this publication available to Language Preprocessor users.

**SYSPRINT Output**

The content of the SYSPRINT data set contains output appropriate for printing. The first column of each line in SYSPRINT contains a carriage control character. The listing file, in the SYSPRINT data set, contains the following items in the order in which they are listed:

1. The options specified in the input operand string via the PARM keyword on the JCL EXEC statement

2. Messages flagging errors in the input operand string, if any

3. The options used (a listing of each option, specified or defaulted, and its setting for the run)—if OPTIONS has been specified in the input operand string

4. A listing of the contents of SYSIN together with line numbers, which are assigned sequentially by the Language Preprocessor to the lines of the file—if SOURCE has been specified in the input operand string

5. Messages generated by the Language Preprocessor when translating SYSIN, together with:
   a. Dictionary listing output, if the messages refer to the execution of a command list and DDLIST is specified in the input operands
   b. An image of the line in SYSIN, if any, which is referred to by the message

6. Summary statistics:
   a. Number of lines scanned
   b. Number of inclusion statements recognized within SYSIN
   c. Number of inclusion statements replaced by source code from the Dictionary
   d. Number of messages generated at each severity level

7. The return code from the Language Preprocessor
Figure 109 illustrates the type of output sections that are produced within SYSPRINT:

- The options specified for the Language Preprocessor run; a suboperand of the HOST option has been misspelled but the execution continued with the value defaulting to COBOL.

- The input source is listed with sequence numbers assigned by the Language Preprocessor; these numbers continue throughout the listing and are referred to by Language Preprocessor diagnostics.

- The Dictionary messages are optionally included following an expanded data structure. In the example, these pertain to the command list processing caused by the inclusion statement COPY DISKREC. They can be suppressed by using the Language Preprocessor option, NODDLIST.

- Summary statistics for the Language Preprocessor run; these occur at the end of the SYSPRINT listing.

---

OPTIONS SPECIFIED:

HOST(COBOL), DDLIST, CLISTSTATUS(T)), LIBRARY

DBLP27 E INVALID OR MISSING OPERAND(S) FOR OPTION 'HOST

---

OPTIONS USED:

HOST(COBOL)
QUOTE
CLISTCATEGORY(SEGMENT), STATUS(T), CODE(C), OCCURRENCE(0), UDNO(1), LINENO(1))
LIBRARY
STATUSLIST(TP)
EXIT
DLI
SEARCHRULE(BOTH)
DDLIST
FLAG(I)
LINECOUNT(55)
OPTIONS
SOURCE
MODE(BATCH)

---

Figure 109 (Part 1 of 3). Example of SYSPRINT Output Produced by the Language Preprocessor
IDENTIFICATION DIVISION.
PROGRAM-ID. HCLPE001.
REMARKS.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT DISKREC ASSIGN TO UT-2400-S-DREC01.
   SELECT DISCRC-RPT ASSIGN TO UT-2400-S-DREC02.
DATA DIVISION.
FILE SECTION.
FD INDEXED FILE LABEL RECORDS STANDARD.
******************************************************************************
* * RETRIEVE STRUCTURE FROM THE DICTIONARY AS CREATED *
* * IN STEP 1 OF TESTCASE HCLPE001. *
* *
* COPY DISKREC.
******************************************************************************
WORKING-STORAGE SECTION.
77 PAY-Scale PIC X(10).
PROCEDURE DIVISION.
GOBACK.

Figure 109 (Part 2 of 3). Example of SYSPRINT Output Produced by the Language Preprocessor
Diagnostics:

COPY DISKREC.

DBDL01 I LINE 17 COLUMN 12 RETURN CODE IS 0 FOR COMMAND LIST SEGMENT (T,C,DISKREC,0) UDNO=1 LINENO=1
09:54:47 11/30/82 INPUT RECORD /SETLOGI=Y; /

DBD0026 I OK**
09:54:47 11/30/82 INPUT RECORD /SETCLIST SEGMENT (T,C,DISKREC,0) UDNO=1 LINENO=1 + /
09:54:47 11/30/82 INPUT RECORD /STATUSLIST=(T,P); /

DBD6423 I SETCLIST PROCESSING SUCCESSFULLY COMPLETED
DBD6413 I CLIST CONTAINED IN SEGMENT TC DISKREC 0
DBD6413 I UDNO = 1
DBD6413 I LINENO = 1

DBD6413 I STATUSLIST = (T,P)
09:54:50 11/30/82 INPUT RECORD /RUNCLIST LANG=C; /

DBD0728 I MEMBER NAME DEFAULTED TO 'NONAME'

DBD0735 I DVALUES OPTION DEFAULTED TO 'NO'

DBD0734 I COMPRESS OPTION DEFAULTED TO 'NO'

DBD0737 I LEVEL OPTION DEFAULTED TO '1'

DBD0736 I NARRATIVE OPTION DEFAULTED TO 'YES'

DBD0726 I STRUCTURES OUT COMPLETE

STRUCTURES_OUT SEG (P,C,DISKREC,0) LANG=C ; DBDLPROC RETURN CODE = 00

DBD6401 I CLIST PROCESSING COMPLETE

DBDL12 I NO SOURCE-COMPUTER PARAGRAPH - DEBUGGING MODE OFF

DB/DC DATA DICTIONARY RELEASE 6 - LANGUAGE PREPROCESSOR 11/15/85 09:5 4:15 PAGE 5

Summary:

------

NUMBER OF LINES SCANNED: 22

NUMBER OF PREPROCESSOR STATEMENTS RECOGNIZED: 1
NUMBER OF PREPROCESSOR STATEMENTS REPLACED: 1

NUMBER OF LANGUAGE PREPROCESSOR MESSAGES:

INFORMATION 2
WARNING 0
ERROR 1
SEVERE 0
UNRECOVERABLE 0

RETURN CODE: 8

Figure 109 (Part 3 of 3). Example of SYSPRINT Output Produced by the Language Preprocessor
Establishing Preprocessor Operations

This section gives you guidance in setting up the Language Preprocessor operating environment. The topics are aimed toward tailoring your installation to groups of users you have identified that can benefit from preprocessor use. For details of the installation requirements, modifying JCL procedures, and performing any required library initialization, refer to Part 1 of *OS/VS DB/DC Data Dictionary Installation Guide*.

Tailoring JCL Procedures

The Language Preprocessor usually operates as a job step within a compilation procedure. By adjusting the JCL control for the compiler or assembly step, so that it tests for the condition code set by the Language Preprocessor, you can avoid unnecessary compiler processing. Another approach might be to have users decide whether to automatically proceed to the compilation step; they could choose to run the Language Preprocessor only as a check, examining the expanded source program before compiling the program.

Tailoring Preprocessor Options

One of the features of the Language Preprocessor is the ability to tailor the operation to the needs of its group of end users. This is partly achieved by tailoring the Language Preprocessor itself and partly by preparation of the Dictionary content. You can tailor the Language Preprocessor in the following ways:

- Operate stand-alone as a batch IMS/VS, or with IMS/VS online (as a BMP), or with CICS/OS/VS Shared Data Base support.
- Process with COBOL, PL/I, or Assembler source.
- Allow the use of COPY/INCLUDE libraries with the Dictionary as sources of extracted language data structures.
- Specify diagnostic and output listing options.
- Specify the use of an installation-written user exit.
- Specify different ways for requests for data structure output to be resolved by the Dictionary.

With the exception of the last item, which has to be coordinated with preparations made within the Dictionary itself, the above tailoring actions are achieved with a set of *Language Preprocessor defaults*. The defaults are initialized with the *Default Options Processor*. The values for the defaults are built into a *Default Options Table* used by the Language Preprocessor during its execution. In addition, you can override many of the values for the defaults at Language Preprocessor execution time. This is done with a PARM string on the Language Preprocessor EXEC statement.

Although several of the defaults established for the Language Preprocessor relate to the ways that Dictionary processing responds to the discovery of inclusion
statements in the source program, these tailoring actions are best understood from
the viewpoint of preparing the Dictionary. Those items are described as
"Dictionary Control" defaults in the sections that follow.

Choosing Language Preprocessor Default Options

This section describes the Language Preprocessor defaults and gives guidance on
how to specify them to fit your installation's requirements.

The Default Options Table

The Language Preprocessor obtains the default values for all of its options from the
Default Options Table. (This table should not be confused with the Dictionary
Defaults module—DBDGFALT.) The table contains the Dictionary-supplied
default value for each option, but any of these values may be overridden by using
the Default Options Processor to create an installation-defined Default Options
Table.

Figure 110 summarizes all the options stored in the Default Options Table, gives a
brief explanation of each option's purpose, and indicates whether an
execution-time option override is allowed. The first nine options cannot be
overridden and are initialized only with the Default Options Processor. The other
options can be overridden by the user at execution time.

<table>
<thead>
<tr>
<th>Option</th>
<th>Execution-Time Override Allowed</th>
<th>Explanation and Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>BATCHPARMS</td>
<td>No</td>
<td>Specify a default PARM string for IMS/VS. The IBM-supplied default is: DBB,DBD5PTRN,DBDIMSO,,00</td>
</tr>
<tr>
<td>BMPPARMS</td>
<td>No</td>
<td>Specify a default PARM string for online IMS/VS. The IBM-supplied default is: BMP,DBD5PTRN,DBDIMSO,,C00100</td>
</tr>
<tr>
<td>CICSPARMS</td>
<td>No</td>
<td>Specify a default PARM string for CICS/OS/VS execution with Shared Data Base. The IBM-supplied default is: SSA=200,PGM=DBD5PTRN,PSB=DBDIMSO, LANG=A,CWTO=N,CMPAT=Y</td>
</tr>
<tr>
<td>GRAFG</td>
<td>No</td>
<td>Specify a default value for the graphic G character. The IBM-supplied default is 42C7.</td>
</tr>
<tr>
<td>GRAFQUOTE</td>
<td>No</td>
<td>Specify a default value for the graphic quotation mark. The IBM-supplied default is 427D.</td>
</tr>
</tbody>
</table>

Figure 110 (Part 1 of 3). Options Stored in the Default Options Table
<table>
<thead>
<tr>
<th>Option</th>
<th>Execution-Time Override Allowed</th>
<th>Explanation and Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRAFLDEL</td>
<td>No</td>
<td>Specify a default value for the graphic left delimiter. The IBM-supplied default is 0E.</td>
</tr>
<tr>
<td>GRAFRDEL</td>
<td>No</td>
<td>Specify a default value for the graphic right delimiter. The IBM-supplied default is 0F.</td>
</tr>
<tr>
<td>SEPCHAR</td>
<td>No</td>
<td>Specify, as part of Dictionary control, a separator character for user names built from inclusion statements that fits your installation's naming convention. The IBM-supplied default is '.'.</td>
</tr>
<tr>
<td>SIGNON</td>
<td>No</td>
<td>If user access control is active, specify a user ID and current password for each table. The IBM-supplied default is a null sign-on.</td>
</tr>
<tr>
<td>CLIST</td>
<td>Yes</td>
<td>Specify category and subject name qualifiers for command lists. The IBM-supplied defaults are: CATEGORY - SEG, CODE - C, LINENO - 1, OCCURRENCE - 0, STATUS - T, UDNO - 1.</td>
</tr>
<tr>
<td>DDLIST</td>
<td>Yes</td>
<td>Request Dictionary messages to be listed. The IBM-supplied default is NODDLIST.</td>
</tr>
<tr>
<td>DLI</td>
<td>Yes</td>
<td>Specify subject codes for language. The IBM-supplied default is DLI.</td>
</tr>
<tr>
<td>EXIT</td>
<td>Yes</td>
<td>Specify name of user exit, if any. The IBM-supplied default is a null user exit routine name.</td>
</tr>
<tr>
<td>FLAG</td>
<td>Yes</td>
<td>Control severity of messages listed. The IBM-supplied default is I.</td>
</tr>
<tr>
<td>HOST</td>
<td>Yes</td>
<td>Specify COBOL, PL/I, or Assembler. The IBM-supplied default is COBOL.</td>
</tr>
<tr>
<td>LIBRARY</td>
<td>Yes</td>
<td>Specify library name in user name. The IBM-supplied default is SYSLIB.</td>
</tr>
<tr>
<td>LINECOUNT</td>
<td>Yes</td>
<td>Set line count for listing page. The IBM-supplied default is 55.</td>
</tr>
<tr>
<td>MARGINS</td>
<td>Yes</td>
<td>Specify margins for source scan. The IBM-supplied default is 2,72.</td>
</tr>
</tbody>
</table>

Figure 110 (Part 2 of 3). Options Stored in the Default Options Table
<table>
<thead>
<tr>
<th>Option</th>
<th>Execution-Time Override Allowed</th>
<th>Explanation and Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td>Yes</td>
<td>Direct operation to be IMS/VS (batch or BMP), or CICS/OS/VS with Shared Data Base support. The IBM-supplied default is BATCH.</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>Yes</td>
<td>Request list of all default options. The IBM-supplied default is OPTIONS.</td>
</tr>
<tr>
<td>QUOTE</td>
<td>Yes</td>
<td>Specify ' or &quot; character for COBOL. The IBM-supplied default is QUOTE.</td>
</tr>
<tr>
<td>SEARCHRULE</td>
<td>Yes</td>
<td>Specify use of COPY/INCLUDE libraries or not. The IBM-supplied default is BOTH.</td>
</tr>
<tr>
<td>SOURCE</td>
<td>Yes</td>
<td>Request listing of source input. The IBM-supplied default is SOURCE.</td>
</tr>
<tr>
<td>STATUSLIST</td>
<td>Yes</td>
<td>Specify status codes for command lists. The IBM-supplied default is TP.</td>
</tr>
</tbody>
</table>

Figure 110 (Part 3 of 3). Options Stored in the Default Options Table

Options Initialized Only with the Defaults Options Processor

Figure 110 on page 308 summarizes those Language Preprocessor defaults that must be initialized using the Default Options Processor.

You can see that the defaults in the table are those of a more privileged type or those not subject to change. The meanings of these defaults and the Dictionary-supplied values follows.

**BATCHPARMS (ims-operand-string)**
This operand specifies the operand string that must be specified when the IMS/VS batch region (DFSRCC00), is executed and the Language Preprocessor is running as a batch job. For an explanation of this character string, refer to *IMS/VS System Programming Reference Manual*. ims-operand-string must be at least 3 bytes in length and at most 100 bytes. The Dictionary-supplied default is **DBB, DBD5PTRN, DBDMSO**, 00.

**BMPPARMS (bmp-operand-string)**
This operand specifies the operand string that must be specified when the IMS/VS batch region controller, DFSRCC00, is executed and the Language Preprocessor is running as a Batch Message Processing program. For an explanation of this character string, refer to *IMS/VS System Programming Reference Manual*. bmp-operand-string must be at least 3 bytes in length and at most 100 bytes. The Dictionary-supplied default is **BMP, DBD5PTRN, DBDMSO**, C00100.

**CICSPARMS (CICS-operand string)**
This operand specifies the operand string that must be specified when the CICS/OS/VS-IMS/VS batch region (DFHDRP) is executed and the
Language Preprocessor is running as a batch application with shared data support. For an explanation of this character string, refer to the CICS/OS/VS Installation and Operations Guide. CICS-operand-string must be at least 3 bytes in length and at most 100 bytes. The Dictionary-supplied default is SSA=200,PGM=DBD5PTRN,PSB=DBDMSO,LANG=A,CWTO=N, CMPAT=Y.

GRAFG (graphic-G)
This operand specifies the hexadecimal value for the graphic G. graphic-G must be four hexadecimal digits. The Dictionary-supplied default is 42C7.

GRAFLDEL (graphic-left-delimiter)
This operand specifies the hexadecimal value for the graphic left delimiter. graphic-left-delimiter must be two hexadecimal digits. The Dictionary-supplied default is 0E.

GRAFQUOTE (graphic-quotation-mark)
This operand specifies the hexadecimal value for the graphic quotation mark. graphic-quotation-mark must be four hexadecimal digits. The Dictionary-supplied default is 427D.

GRAFRDEL (graphic-right-delimiter)
This operand specifies the hexadecimal value for the graphic right delimiter. graphic-right-delimiter must be two hexadecimal digits. The Dictionary-supplied default is 0F.

SEPCHAR (separator-character)
This operand specifies the separator character for Dictionary concatenated user names. To obtain the user name of the Dictionary subject, the Language Preprocessor concatenates the library name in front of the member name (from the inclusion statement), the two being separated by the separator character. It is chosen as part of Dictionary control so that it fits both the source language requirements and your installation's naming conventions. The separator-character must be a single character from the following set: '@', '#', '$', ',', ';', '/', '.', '___', and '-'... The Dictionary-supplied default is '.

SIGNON (Dictionary-id, Dictionary-password)
This operand specifies the operands needed to sign on to the Dictionary. Dictionary-id must be from 1 to 31 bytes in length and Dictionary-password must be from 1 to 8 bytes in length. Each must be composed using the EBCDIC characters 'A' through 'Z', '0' through '9', '@', '#', '$', ',', ';', '/', '.', '___', and '-'. The first character must not be '___' or '-'. A null sign-on is specified by omitting Dictionary-id, Dictionary-password, and indicates that no SIGNON ON commands should be issued. The Dictionary-supplied default for this operand is a null sign-on.
Options That Users May Override

This section describes the input operands which may be specified at execution time with the PARM keyword of the JCL EXEC statement which invokes the Language Preprocessor (see Figure 110 on page 308). The Dictionary-supplied default value for each operand is given. These defaults can be overridden by modifying the Default Options Table, using the Default Options Processor.

Syntax Rules for Language Preprocessor PARM Keyword on EXEC Statement

- Operands can be specified in any order.
- Operands must be separated by blanks, or a comma if desired.
- Blanks are allowed within an operand.
- Opening and closing parentheses are required when using suboperands.

Note that, because the maximum length of an operand string is 100 bytes, not all of the input operands can be specified in a operand string. Therefore, defaults must be used.

The following is an example of an operand string showing many of the variations allowed in the syntax:

'HOST(COBOL),CLIST(OCR(1),STATUS(IP),CODE(L))
LIBR(LIB1),APOST,DDLST READ(SHORT)' 

Available Input Operands

Following is a list of the input operands.

CLIST (suboperands)
This operand specifies the category, status, code, occurrence number, User Data number, and starting line number of command lists. The suboperands are:

CATEGORY | CAT (category-name)
This suboperand specifies the category of command list subjects.
category-name must be from 1 to 8 bytes in length and be composed using the characters, 'A' through 'Z', '0' through '9', '@', '#', '$', '!', '!', ',', '-', '. '. The first character must be 'A'-'Z', '@', '#', or '$'. DBCS users must enter a uniform string. The Dictionary-supplied default is SEG.

Note: The restriction on the length of category-name means that the standard category name, TRANSACTION, will not be allowed. To specify the TRANSACTION standard category as the category of command lists, one of its synonyms, T, TRN, or TRANSACT, must be used.

CODE (code-letter)
This suboperand specifies the subject code of command lists.
code-letter must be a single character, which is one of the valid values for Dictionary subject codes. A null code is specified by omitting code-letter. A null code is required for extensibility subjects. The Dictionary-supplied default is C.
LINENO | LN (starting-line-number)
This suboperand specifies the User Data line number of the first
command in command lists. starting-line-number must be an integer in
the range of 1 to 999 inclusive. The Dictionary-supplied default is 1.

OCCURRENCE | OCC (occurrence-number)
This suboperand specifies the occurrence number of command lists.
ocurrence-number must be an integer in the range of 0 to 255,
inclusive. The Dictionary-supplied default is 0.

STATUS (status-character)
This suboperand specifies the status of command lists.
status-character must be a single character from 'A' through 'T', '0'
through '9' or '&'. If status-character is '&', then the status list
determined by the STATUSLIST input operand will be used in
determining the status of each command list. The Dictionary-supplied
default is 'T'.

Note: To specify an ampersand in an input operand string in JCL, it
is necessary to code two successive ampersands. Therefore, to specify
a status of '&', STATUS(&&) must be coded.

UDNO ( 1 | 2 | 3 | 4 | 5 )
This suboperand specifies the User Data number of command lists.
The Dictionary-supplied default is '1'.

The values of the CATEGORY, CODE, and OCCURRENCE suboperands
must be consistent with Dictionary rules. For a listing of these rules, see
Chapter 1 of OS/VS DB/DC Data Dictionary Terminal User's Guide and
Reference.

DDLIST | NODDLIST
This operand specifies whether Dictionary listing output should be included
in the preprocessor listing. The Dictionary-supplied default is NODDLIST.

DLI | NODLI
This operand specifies the type of language data structure definition that will
be obtained from the Dictionary. DLI indicates DL/I data base structures
(language codes A, B, and C). NODLI indicates "flat" files (language codes
J, K, and L). The Dictionary-supplied default is DLI.

EXIT (user-exit-routine-name)
This operand specifies the name of a user exit routine to be called after a call
to the Dictionary has resulted in a return code of '4' or less.
user-exit-routine-name must be from 1 to 8 bytes in length and composed
using the EBCDIC characters 'A' through 'Z', '0' through '9', '@', '#',
and '$'. The first character must be 'A'-'Z', '@', '#', or '$'. No DBCS
characters are allowed. A null user exit routine name is specified by omitting
user-exit-routine-name, and indicates that no user exit routine should be
called. The Dictionary-supplied default is a null user exit routine name.

FLAG | F ( I | W | E | S | U )
This operand specifies that Language Preprocessor messages below the
specified severity level are to be suppressed. I specifies information-level
(severity 0), W specifies warning-level (severity 4), E specifies error-level (severity 8), S specifies severe-level (severity 12), and U specifies unrecoverable-level (severity 16). The Dictionary-supplied default is I, which causes no messages to be suppressed.

HOST (COBOL | PLI | ASM)
This operand specifies the host language. The Dictionary-supplied default is COBOL.

LIBRARY (library-name)
This operand specifies the default library name for preprocessor statements. library-name must be from 1 to 8 EBCDIC characters in length and be composed using 'A' through 'Z', '0' through '9', '@', '#', and '$'. The first character must be 'A'-'Z', '@', '#' or '$'. No DBCS characters are allowed. A null library name is specified by specifying the keyword, LIBRARY, but omitting library-name. If a null library name is specified, the Dictionary user name created by the Language Preprocessor will consist of only the member name from the inclusion statement (unless the inclusion statement itself contains a library name). The Dictionary-supplied default is SYSLIB.

LINECOUNT (line-count)
This operand specifies the number of lines per page of the preprocessor listing. line-count must be an integer greater than 9 and less than 100. The Dictionary-supplied default is 55.

MARGINS (m,n)
This operand specifies the left margin (m) and the right margin (n) of the input source program. This operand has meaning only if HOST(PLI) is specified. m and n are each integers. m must be greater than 0. n must be less than 81 and greater than m + 4. The Dictionary-supplied default is 2,72.

Note: Variable margins are not supported in COBOL. Margins are always 8,72. In Assembler, variable margins are supported by the ICTL instruction.

MODE (BATCH | BMP | CICS)
This operand specifies whether the Language Preprocessor is to execute as an IMS/VS Batch job (BATCH), a Batch Message Processing program (BMP), or as a batch application with CICS/OS/VS Shared Data Base support (CICS). The Dictionary-supplied default is BATCH.

OPTIONS (OPTN | NOOPTIONS | NOOPTN)
This operand specifies whether the options in effect should be included in the preprocessor listing. The Dictionary-supplied default is OPTIONS.

QUOTE (APOST | Q)
This operand specifies whether a double quote or an apostrophe (single quote) should serve as the delimiter of literals. This operand has meaning only if HOST(COBOL) is specified. The Dictionary-supplied default is QUOTE.
SEARCHRULE | SR (BOTH | DICTONLY)
This operand specifies that if the execution of commands in a command list results in a message of severity level E or above from the Dictionary, then the inclusion statement should be transformed into a comment to prevent its translation by a subsequent processor (DICTONLY), or the inclusion statement should be copied unchanged to allow translation by the host language compiler or Assembler (BOTH). The Dictionary-supplied default is BOTH.

SOURCE | S | NOSOURCE | NOS
This operand specifies whether SYSIN, the input source program to the Language Preprocessor, should be included in the preprocessor listing. The Dictionary-supplied default is SOURCE.

STATUSLIST | SL (status-list)
This operand specifies Dictionary status codes and the order in which they are to be used when testing for the existence of subjects containing command lists or used in commands within a command list. The status-list must be from 1 to 10 EBCDIC characters in length (within the parentheses) and be composed using the characters 'A' through 'T' and '0' through '9'. The Dictionary-supplied default is TP.

Using the Default Options Processor

The Default Options Processor is used to modify IBM-supplied defaults to create an installation-defined Default Options Table. As discussed in Chapter 3, "Executing the Language Preprocessor Programs" in OS/VS DB/DC Data Dictionary Installation Guide, the Dictionary provides a job to execute the Default Options Processor.

The Default Options Processor accepts as input a SYSIN data set, which contains overriding default values for one or more options. The output of the Default Options Processor is a set of Assembler statements in the SYSPUNCH data set. When assembled, a new Default Options Table is generated (module DBDLPOPT). Multiple Default Options Tables may be generated and a different Table can be used for different groups of Language Preprocessor users.

In addition, the Default Options Processor produces a listing in the SYSPRINT data set, which lists the set of default values stored in the table, as well as messages indicating errors in the input.

As input to the Default Options Processor, defaults are coded in SYSIN in columns 1 to 72, inclusive. Defaults are coded in the form of operand (value). Defaults may "wrap around" from one line to the next, because SYSIN is treated as a continuous string of characters—column 72 of one line is immediately followed by column 1 of the next line. The format for specifying the defaults values is completely free form. The values may be placed anywhere in columns 1 through 72, and may be separated by any number of blanks. No continuation indicator is required if the input length exceeds one line, and continuations may start in any column.

The same syntax is used in SYSIN for specifying the overriding default values for options, as is used in specifying options at execution time with the PARM keyword. However, the limitation upon the length of a string of options when using the PARM string (100-byte maximum) does not apply. There is no limitation on the
length of SYSIN. Also, it is not necessary to code two successive ampersands to obtain a single ampersand when specifying default values in SYSIN, as is required when coding options using the PARM keyword.

Each default value for the options listed in “Options That Users May Override” on page 312 can be given a new default value at execution time. However, the options described in “Options Initialized Only with the Defaults Options Processor” on page 310 may not be specified at execution time with the PARM keyword, but only by executing the Default Options Processor.

Allowing End Users to Specify Language Preprocessor Options

When you are explaining to the end user how to run the Language Preprocessor, your instructions will depend on what degree of flexibility you have decided to allow. You do not have to give information about all the input operands supported by the Language Preprocessor. If you develop a JCL procedure with symbolic operands that override values in the EXEC statement, you will, of course, have to supply appropriate values that can be used—matching the values required for the Language Preprocessor operands. See “Options That Users May Override” on page 312 for a complete description of the operands.

Considerations for the Use of Other Preprocessors

The Language Preprocessor performs an independent scan of an input source program and modifies the source content with language data structures by either the action of the STRUCTURES=OUT command or data produced by the action of the PUNCH command. The checking that it performs on the source program assumes source language syntax rules. If another preprocessor is to be used with the Language Preprocessor, it may be important to decide in which order the programs should be executed.

With the PL/I Preprocessor, the Language Preprocessor must be the first preprocessor to examine the source program. The Language Preprocessor recognizes %INCLUDE inclusion statements and uses them.

There are cases in which the order does not matter. For example, the CICS Command Language Translator, supplied with the CICS/OS/VS licensed program, recognizes and processes only statements within EXEC CICS statements. The Language Preprocessor does not scan within EXEC statements so that there is no constraint that keeps the Language Preprocessor from executing first.

If your installation uses a preprocessor-like program and you have doubts about its compatibility with the Language Preprocessor, refer to the syntax rules in “Giving Information about Code Requirements” on page 331 for the particular source language. You should be able to determine from the rules given there whether or not there is a conflict.
Incorporating a User Exit into the Language Preprocessor Execution

If a user exit name is specified in the EXIT input operand, that user exit will be loaded during Language Preprocessor initialization and then called after each call to the Dictionary if the Dictionary return code is 4 or less. The user exit will not be entered if the return code is greater than 4, because this indicates that a serious problem has probably prevented the generation of valid source code by the Dictionary. The user exit will be called once for each record that is to be copied to SYSPUNCH.

A pointer to the parameter area, a data area containing the Dictionary Language Preprocessor input operands, a communication flag, and the input record are passed to the routine. This data may be used to make decisions on the user exit's execution. The exit may be written in Assembler, COBOL, or PL/I and must be reusable. The exit is allowed to alter the records in any manner, but may not delete or add records. It should set a return code before returning each record to the Language Preprocessor. A return code of 16 from the user exit will cause the Language Preprocessor to terminate processing. All other return codes are ignored.

Coding Requirements for the Language Preprocessor User Exit

The parameter areas for each language are shown in Figure 111, Figure 112, and Figure 113:

- Figure 111 on page 318 relates to COBOL routines.
- Figure 112 on page 319 relates to PL/I routines.
- Figure 113 on page 320 relates to Assembler routines.

The parameter areas are provided with the Dictionary licensed program.
* LANGUAGE PREPROCESSOR PARAMETER AREA FOR A USER EXIT ROUTINE
* WRITTEN IN COBOL
  01 LPPARMS.
  * HOST LANGUAGE INDICATOR
    03 HOST-LANGUAGE
       USAGE COMP PICTURE 9(5).
       88 HOST-ASSEMBLER VALUE ZERO.
       88 HOST-PLI VALUE 1.
       88 HOST-COBOL VALUE 2.
  * LANGUAGE CODE INDICATOR
    03 DLI-FLAG
       USAGE COMP PICTURE 9(5).
       88 DLI-NO VALUE ZERO.
       88 DLI-YES VALUE 1.
  * SEARCH RULE
    03 SEARCH-RULE
       USAGE COMP PICTURE 9(5).
       88 SEARCH-DICTIONARY VALUE ZERO.
       88 SEARCH-BOTH VALUE 1.
  * LEFT MARGIN - WILL ALWAYS CONTAIN THE VALUE 8
    03 LEFT-MARGIN
       USAGE COMP PICTURE 9(5).
  * RIGHT MARGIN - WILL ALWAYS CONTAIN THE VALUE 72
    03 RIGHT-MARGIN
       USAGE COMP PICTURE 9(5).
  * CONTINUE MARGIN - WILL ALWAYS CONTAIN THE VALUE 12
    03 CONTINUE-MARGIN
       USAGE COMP PICTURE 9(5).
  * COMMUNICATION FLAG
    03 RECORD-FLAG
       USAGE COMP PICTURE 9(5).
       88 NOT-FIRST-OR-LAST VALUE 0.
       88 FIRST-RECORD VALUE 1.
       88 LAST-RECORD VALUE 2.
       88 FIRST-AND-LAST VALUE 3.
  * DICTIONARY STATUS LIST
    03 STATUS-LIST
       USAGE DISPLAY PICTURE X(10).
  * RESERVED
    03 LP-RESERVED-1
       USAGE DISPLAY PICTURE X(2).
  * COMMAND LIST NAME
    03 COMMAND-LIST.
  * CATEGORY OF COMMAND LIST
    05 CLIST-CATEGORY
       USAGE DISPLAY PICTURE X(8).
  * STATUS OF COMMAND LIST
    05 CLIST-STATUS
       USAGE DISPLAY PICTURE X.
  * CODE OF COMMAND LIST (BLANK = NULL CODE)
    05 CLIST-CODE
       USAGE DISPLAY PICTURE X.
  * USERNAME OF COMMAND LIST
    05 CLIST-USER-NAME
       USAGE DISPLAY PICTURE X(17).
  * RESERVED
    05 LP-RESERVED-2
       USAGE DISPLAY PICTURE X(1).
  * OCCURRENCE NUMBER OF COMMAND LIST
    05 CLIST-OCCUR-NO
       USAGE COMP PICTURE 9(5).
  * USERDATA NUMBER OF COMMAND LIST
    05 CLIST-USERDATA-NO
       USAGE COMP PICTURE 9(5).
  * STARTING LINE NUMBER OF COMMAND LIST
    05 CLIST-START-LINE-NO
       USAGE COMP PICTURE 9(5).
  * CURRENT RECORD
    03 DD-RECORD
       USAGE DISPLAY PICTURE X(80).
    03 DD-RECORD-CHAR
       REDEFINES DD-RECORD PICTURE X
       OCCURS 80 TIMES.
  * USER EXIT RETURN CODE
    03 USER-RET-CODE
       USAGE COMP PICTURE 9(5).
       88 LP-TERMINATE VALUE 16.

Figure 111. Language Preprocessor User Exit: COBOL Parameter Area
/* LANGUAGE PREPROCESSOR PARAMETER AREA FOR A USER EXIT ROUTINE */
DCL LPPTR POINTER;
DCL
  1 LPPARMS BASED(LPPTR),
     3 HOSTLANG BIN FIXED(31), /* HOST LANGUAGE INDICATOR */
                0 = ASSEMBLER
                1 = PLI
                2 = COBOL */
  3 DLIFLAG BIN FIXED(31), /* LANGUAGE CODE INDICATOR */
             0 = FLAT FILE CODES
             1 = DL/I CODES */
  3 SRCHRULE BIN FIXED(31), /* SEARCH RULE */
             0 = DICTIONARY ONLY
             1 = DICTIONARY AND COPYLIBS */
  3 LMARGIN BIN FIXED(31), /* LEFT MARGIN */
  3 RMARGIN BIN FIXED(31), /* RIGHT MARGIN */
  3 CMARGIN BIN FIXED(31), /* NOT APPLICABLE */
  3 LPFLAG BIN FIXED(31), /* COMMUNICATION FLAG */
             0 = NOT FIRST OR LAST
             1 = FIRST RECORD
             2 = LAST RECORD
             3 = BOTH FIRST AND LAST */
  3 STATLIST CHAR(10), /* DICTIONARY STATUS LIST */
  3 LPRSRVD1 CHAR(2), /* RESERVED */
  3 CLIST, /* COMMAND LIST NAME */
  5 CLISTCAT CHAR(8), /* COMMAND LIST CATEGORY */
  5 CLISTSTA CHAR(1), /* COMMAND LIST STATUS */
  5 CLISTCOD CHAR(1), /* COMMAND LIST CODE (BLANK=NULL) */
  5 CLISTUSR CHAR(17), /* COMMAND LIST USERNAME */
  5 LPRSRVD2 CHAR(1), /* RESERVED */
  5 CLISTOCCR BIN FIXED(31), /* COMMAND LIST OCCURRENCE NO. */
  5 CLISTUDN BIN FIXED(31), /* COMMAND LIST USERDATA NO. */
  5 CLISTLNO BIN FIXED(31), /* COMMAND LIST STARTING LINE NO. */
  3 DDRECORD CHAR(80), /* CURRENT RECORD */
  3 USRRCODE BIN FIXED(31); /* USER EXIT RETURN CODE */
       16 = REQUEST PREPROCESSOR TERMINATION */

Figure 112. Language Preprocessor User Exit: PL/I Parameter Area

Entry Points and Returns for User Exit Routines

You use your own entry points for the Language Preprocessor exit routines and make use of the parameter area passed by the Language Preprocessor. To return to the Language Preprocessor, you use standard returns. The return code set by an exit routine is part of the parameter area passed to the routine. Examples of these statements are:

For COBOL:
Entry point:  ENTRY 'user-entry' USING LPPARMS.
To return:  GOBACK.

For PL/I:
Entry point:  user_entry:  PROC(LPPTR) OPTIONS(MAIN);
To return:  RETURN;
            END user-entry;

For Assembler:
Entry point:  ENTRY user-entry
To return:  BR R14
* LANGUAGE PREPROCESSOR PARAMETER AREA FOR A USER EXIT ROUTINE
LPPARMS DSECT
HOSTLANG DS F HOST LANGUAGE INDICATOR
HOSTASM EQU 0 LANGUAGE = ASSEMBLER
HOSTPLI EQU 1 LANGUAGE = PL/I
HOSTCBL EQU 2 LANGUAGE = COBOL
DLIFLAG DS F LANGUAGE CODE INDICATOR
DLINO EQU 0 USING FLAT FILE CODES (J,K,L)
DLIYES EQU 1 USING DL/I CODES (A,B,C)
SRCHRULE DS F SEARCH RULE
SRCHDICT EQU 0 GET ALL STRUCTURES FROM DICTIONARY
SRCHBOTH EQU 1 GET FROM DICTIONARY AND COPY LIBS
LMARGIN DS F LEFT MARGIN (BEGIN COLUMN)
RMARGIN DS F RIGHT MARGIN (END COLUMN)
CMARGIN DS F CONTINUE MARGIN (CONTINUE COLUMN)
NOCONT EQU 0 CONTINUATIONS NOT ALLOWED
LPFLAG DS F COMMUNICATION FLAG
LPNONE EQU 0 NOT FIRST OR LAST RECORD
LPFIRST EQU 1 FIRST RECORD FROM DDPUCH
LPLAST EQU 2 LAST RECORD FROM DDPUCH
LPBOTH EQU 3 RECORD IS BOTH FIRST AND LAST RECORD
STATLIST DS CL10 DICTIONARY STATUS LIST
LPRSRVD1 DS CL2 RESERVED
CLIST DS OCL40 COMMAND LIST DEFINITION
CLISTCAT DS CL8 COMMAND LIST CATEGORY
CLISTSTA DS CL1 COMMAND LIST STATUS
CLISTCOD DS CL1 COMMAND LIST CODE (BLANK = NULL CODE)
CLISTUSR DS CL17 COMMAND LIST USERNAME
LPRSRVD2 DS CL1 RESERVED
CLISTOCR DS F COMMAND LIST OCCURRENCE NO.
CLISTUDN DS F COMMAND LIST USERDATA NO.
CLISTLNO DS F COMMAND LIST STARTING LINE NO.
DDRECORD DS CL80 CURRENT RECORD
USRRCODE DS F USER EXIT RETURN CODE
LPTERM EQU 16 REQUEST PREPROCESSOR TERMINATION

00010000
00020000
00030000
00040000
00050000
00060000
00070000
00080000
00090000
00100000
00110000
00120000
00130000
00140000
00150000
00160000
00170000
00180000
00190000
00200000
00210000
00220000
00230000
00240000
00250000
00260000
00270000
00280000
00290000
0030000
00310000
00320000
00330000
00340000
00350000
00360000

Figure 113. Language Preprocessor User Exit: Assembler Parameter Area

Link-Editing Language Preprocessor Exit Routines

If you use an INCLUDE statement with your link-editing control statements, your
link-editing step’s JCL has a DD statement that defines an object library (PDS)
that contains your exit routine. Otherwise, you can include the object with the
link-editing control statements:

```java
//SYSIN DD *
  object deck
  ENTRY user-entry
  NAME userpgm(R)
```

The entry point for exit routines is assumed by the Dictionary Language
Preprocessor to be at the beginning of the module, the primary entry point.
"userpgm" is the name of the resulting load module. This name is specified with
the EXIT operand in the Language Preprocessor Default Options Table or in the
PARM keyword of the EXEC statement.
Preparing the Dictionary for Language Preprocessor Access

This section explains the preparation you need to make within the Dictionary so that a request for output (caused by recognition of an inclusion statement by the Language Preprocessor) can be resolved. You need to know about the various ways that this can be achieved to enable you to:

- Understand the Dictionary support and decide which ones to use
- Decide what controls you will put in place for supporting Language Preprocessor use
- Plan your ongoing support for the end users

Authorizing the Use of the Language Preprocessor

If your installation has access control enabled, you must have a sign-on and password for each Language Preprocessor user. To avoid having each user specify a unique sign-on and password, you can first set up a special sign-on and password in the Default Options Table through the SIGNON option. Then you can set up an access profile for a group of users with that special sign-on and password and provide the appropriate authorization. You can set up a default module and access profile for any number of different user groups. The following summarizes the actions required:

- Establish a unique user ID and password for each operating version of the Language Preprocessor, assuming that a group of users will use it.

- Each group becomes a Dictionary user; you add each user ID with the password as a subject in the DDUSER category.

- Add access profiles to the above DDUSER subjects; you need:
  
  - Authorization to the SETLOGI, SETCLIST, RUNCLIST, STRUCTURES__OUT, and PUNCH commands

  - Use authorization to the categories used for storing command lists and any from which language data structure data is extracted

  - View authorization for all status codes used in the SETCLIST and implied by the RUNCLIST execution, including those to be used by the STATUSLIST operand

For further details of how to create these user access profiles, refer to Chapter 2, "Controlling Access to Dictionary Definitions" on page 13.
Coordinating Language Preprocessor Use with Dictionary Data

The key to successful use of the Language Preprocessor is the preparation made in the Dictionary. With careful tracking of those language data structures that are potentially available to Language Preprocessor users, you can provide a service that has the advantage of being mostly transparent to those users yet offering a means of controlling both currency and installation standards regarding language data structures. The sections that follow explain the mechanisms available to you when coordinating Language Preprocessor use and Dictionary content.

Understanding How the Language Preprocessor Uses Command Lists

The Language Preprocessor uses the Dictionary's command list facility. A complete description of this facility is in OS/VS DB/DC Data Dictionary Applications Guide. A description of how the Language Preprocessor uses the command list facility is given in “How Inclusion Statements Involve Dictionary Processing” on page 296 and “How Data Is Selected from the Dictionary” on page 298.

Before the Language Preprocessor can extract any data from the Dictionary, you need to establish the necessary command lists. Command lists used by the Language Preprocessor are limited to the STRUCTURES__OUT and PUNCH commands. You may establish your command lists in a set of User Data for a subject in any Dictionary category—standard or extensibility. Your choice of category may depend on the application development and maintenance procedures used in your installation. See “Using Indefinite Status Codes” on page 323 for more information. You can establish your command lists, using any of the Dictionary's facilities that support the creation and maintenance of User Data text—display forms, update commands, or batch forms input. For DBCS users, the user name portion of the subject name in which the command list is stored cannot contain DBCS characters.

How the Language Preprocessor Controls the SETCLIST Command

When the Language Preprocessor encounters an inclusion statement in the source program, it automatically generates a SETCLIST command. It determines the values specified for the operands from its input operands, either from the override operands entered on the PARM keyword of the EXEC statement or defaulted from the Default Options Table.

The values in Figure 114 on page 323 are obtained from the Default Options Table (except for user name) unless overridden by the PARM keyword of the EXEC statement. The user name is obtained from the inclusion statement.
<table>
<thead>
<tr>
<th>SETCLIST Operand</th>
<th>Obtained from</th>
<th>May Be Overridden</th>
</tr>
</thead>
<tbody>
<tr>
<td>category</td>
<td>CLIST CATEGORY</td>
<td>yes</td>
</tr>
<tr>
<td>subjectname status</td>
<td>CLIST STATUS</td>
<td>yes</td>
</tr>
<tr>
<td>code</td>
<td>CODE</td>
<td>yes</td>
</tr>
<tr>
<td>occurrence</td>
<td>OCCURRENCE</td>
<td>yes</td>
</tr>
<tr>
<td>user name library name</td>
<td>Source program or LIBRARY</td>
<td>yes*</td>
</tr>
<tr>
<td>separator character</td>
<td>SEPCHAR</td>
<td>no</td>
</tr>
<tr>
<td>user name</td>
<td>Source program</td>
<td>no</td>
</tr>
<tr>
<td>UDNO</td>
<td>CLIST UDNO</td>
<td>yes</td>
</tr>
<tr>
<td>LINENO</td>
<td>CLIST LINENO</td>
<td>yes</td>
</tr>
<tr>
<td>STATUSLIST</td>
<td>STATUSLIST</td>
<td>yes</td>
</tr>
</tbody>
</table>

*If the library name is contained in the inclusion statement, it cannot be overridden. If the inclusion statement contains no library name, the LIBRARY input operand is used.

**Figure 114. Where SETCLIST Command Obtains Operand Information**

When a SETCLIST command is generated, the category and subject name of the location of the command list must be included. The action of the Language Preprocessor in recognizing an inclusion statement determines the user name portion; the subject name qualifiers, and the category name come from the input operands. With the exception of the user name portion of the subject name, all other operands (including subject name qualifiers) are fixed for all inclusion statements encountered in the source program.

If an indefinite status code is coded as the value for the CLIST suboption in the PARM keyword of the EXEC statement, a double ampersand (&&) must be used. This appears as a single & in the generated SETCLIST command. Otherwise, the CLIST status code can be any valid code.

**Using Indefinite Status Codes**

During development of an application program, a language data structure will be created and will go through several iterations. For example, a data structure may be created in test status, promoted to production status, demoted to test status, modified, and finally promoted to production status. Your command lists can be built in such a way that no changes need to be made to command lists when a language data structure moves from one development phase to another.

There are two ways to manage the variations that occur in the status codes for test and production level subjects:
• Specify an **indefinite status code** for the subject names referenced in the commands in the **command list** by use of an ampersand symbol (&) instead of a particular status code.

Subjects referenced within command lists can use an indefinite status code. For all STRUCTURES__OUT and PUNCH commands in the command list, those referencing a subject name having an & status code do **not have their status code fixed**. The status code used in the command is obtained from the list specified in the STATUSLIST operand of the SETCLLIST command.

For example, if you have a STRUCTURES__OUT command of:

```plaintext
STRUCTURES_OUT SEG (&,A,PAYSEG,0) ;
```

and the STATUSLIST operand of the SETCLLIST command has a status of TP, the Dictionary will first check for a subject with a status of T. If that can't be found, a subject with a status of P will be checked for next.

• Specify an **indefinite status code** for the name of the subject containing the command list on a SETCLLIST command. The Dictionary will use the STATUSLIST operand on the SETCLLIST command to get the name of the subject containing the command list. If an indefinite status code is also used within the command list, the Dictionary will use the status of the subject containing the command list for the subjects referenced within the command list.

For example, if you have the following SETCLLIST command:

```plaintext
SETCLLIST SEG (&,A,PAYSEG,0) STATUSLIST=(T,P) ;
```

the generated name for the subject containing the command list would be (T,A,PAYSEG,0). If that subject could not be found, the Dictionary would then generate a name of (P,A,PAYSEG,0). If subject (P,A,PAYSEG,0) is found and its command list is, for example:

```plaintext
STRUCTURES_OUT SEG (&,A,PAYSEG,0) ;
```

the Dictionary will generate a name of (P,A,PAYSEG,0) for the STRUCTURES__OUT command.

The indefinite status code feature of command list processing is used directly by the Dictionary Language Preprocessor. “Using an Indefinite Status Code in the Command List” on page 325 and “Using an Indefinite Status Code in the Command List and SETCLLIST Command” on page 326 explain in greater detail the mechanism used to select a status code to replace an indefinite status code for each of the above cases.

The use of indefinite status codes is permitted only on the SETCLLIST, STRUCTURES__OUT, and PUNCH commands.

In the case of a STRUCTURES__OUT command that specifies more than one subject having an indefinite status, the status of each subject will be determined independently. If no status in the status list generates the subject name of an existing subject, the command will not be submitted, an error message will be issued, and subsequent processing of the command list will continue as if a
"subject-not-found" condition code had been returned from execution of the command and the command will be subject to the value of the STOPON operand.

Using an Indefinite Status Code in the Command List

Suppose you have created the language data structures, (T,A,PAYSEG1,0), (T,A,PAYSEG2,0), and (T,A,PAYSEG3,0). The following Dictionary commands will build a command list in the first set of User Data for the subject (P,,LIST,0) in the CLIST Extensibility category:

ADD CLIST (P,,LIST,0) +
  EXTUSER1=(1,'STRUCTURES_OUT SEG +
    ((&A,PAYSEG1,0),(&A,PAYSEG2,0))')
  ;
ADD CLIST (P,,LIST,0) +
  EXTUSER1=(2,'STRUCTURES_OUT SEG (&A,PAYSEG3,0)')
  ;
ADD CLIST (P,,LIST,0) EXTUSER1=('3','ENDDATA')
  ;

You can modify any of the three data structures without having to change the command list. After testing is completed, you can use the Dictionary COPY command to obtain a production level definition, for example, (P,A,PAYSEG1,0). Delete the test level structure. The other two data structures, PAYSEG2 and PAYSEG3, remain in test level status.

Let's assume that the PARM keyword of the EXEC statement, which executes the Language Preprocessor, is specified as follows:

CLIST(STATUS(P),CODE,OCC(0),CAT(CLIST)),LIBRARY

Let's also assume that the following operands are taken from the Default Options Table:

- UDNO=1
- LINENO=1
- STATUSLIST=TP

The user name, LIST, is obtained from the COPY or INCLUDE statement.

Then the Language Preprocessor will generate the following commands when it encounters a COPY or INCLUDE statement in the source program:

SETCLIST CLIST (P,,LIST,0) UDNO=1 LINENO=1 STATUSLIST=(T,P) ;
RUNCLIST

The SETCLIST command uses a STATUSLIST operand to specify the list of status codes. The order in which the status codes are coded is the order in which they are used. You can see that coding more than one status is a way of concatenating status codes much like concatenating COPY or INCLUDE libraries.

The RUNCLIST command will cause the Dictionary to test first for status T. Because PAYSEG1 no longer exists in status T, the Dictionary will search for a PAYSEG1 with a status of P. The Dictionary will find the production definition of PAYSEG1, and the test levels for PAYSEG2 and PAYSEG3, and execute:

STRUCTURES_OUT SEG ((P,A,PAYSEG1,0),(T,A,PAYSEG2,0))
STRUCTURES_OUT SEG (T,A,PAYSEG3,0)
If you do not want to delete the test level version, you can cause the STATUSLIST specification on the SETCLIST command shown above to be changed to STATUSLIST=(P,T) by specifying STATUSLIST(PT) in the PARM keyword of the EXEC statement. The Dictionary would test for and find the production level PAYSE1 structure and will not be aware of the test level version. If all of the data structures in the command list have been promoted to production status, you could simply specify STATUSLIST=P.

Using an Indefinite Status Code in the Command List and SETCLIST Command

Although, in the previous section, a fixed status code was used in the SETCLIST command, you can combine the above technique, indefinite status codes in the command lists, with an indefinite status code in the SETCLIST command. This gives you full flexibility in coordinating the version of a language data structure that is obtained by the Language Preprocessor when expanding inclusion statements.

Let's assume that the command list to be used by the Language Preprocessor is contained in User Data 1 of subjects in the SEGMENT category. You create a subject named (T,A,PAYSEG,0) in the SEGMENT category and subsequently, complete its definition, adding attributes and relating fields to it. In addition, you create a command list in User Data 1 belonging to that subject. The following Dictionary commands will build a command list in the first set of User Data of the PAYSEG segment subject.

ADD SEG (T,A,PAYSEG,0) +
    SEGUSER1=(1, 'STRUCTURES_OUT SEG (A,PAYSEG,0)') ;
ADD SEG (T,A,PAYSEG,0) +
    SEGUSER1=(2, 'ENDDATA') ;

You now execute the Language Preprocessor specifying in the PARM keyword of the EXEC statement:

CLIST (STATUS (&&), CODE (A), OCC (0), CAT (SEGMENT'')), LIBRARY ;

You will notice that the major difference here from the example in the preceding section is in the use of an indefinite status code in the STATUS suboperand of the CLIST operand. In addition, the category of subjects containing command lists is identified as the SEGMENT category.

Again, let's assume that the following operands are taken from the Default Options Table:

- UDNO=1
- LINENO=1
- STATUSLIST=TP

The Language Preprocessor will then generate the following commands when it encounters a COPY PAYSEG or %INCLUDE PAYSEG statement in the source program:

SETCLIST SEG (&&, A, PAYSEG, 0) UDNO=1 LINENO=1 STATUSLIST=(T,P) ;
RUNCLIST ;

When the SETCLIST command is executed, the Dictionary will test for and find the test level subject and the associated command list. When the RUNCLIST
command is executed, the Dictionary will substitute 'T' for the ampersand in all subjects in the command list that use an indefinite status code.

In this example, the command list contains one such command, and RUNCLIST will execute:

```
STRUCTURES_OUT SEG (T,A,PAYSEG,0)  
```

After you have modified the data structure represented by (T,A,PAYSEG,0), and testing is completed, you are ready to promote this structure to production status. You can use the Dictionary COPY command to obtain (P,A,PAYSEG,0), and then you can delete the test level version. No changes need to be made to the command list. Using the same STATUSLIST input operand as above, the RUNCLIST command causes the Dictionary to search for a subject with T status code but not find it, then search for the production level subject. If the test level structure had been deleted subsequent to the promotion of the production structure, the Dictionary will find only the production level subject. The Dictionary will then execute:

```
STRUCTURES_OUT SEGMENT (P,A,PAYSEG,0)  
```

You may decide not to delete the test level structure definition. Instead, the STATUSLIST specification in the Default Options Table can be overridden by specifying STATUSLIST(P) or STATUSLIST(PT) in the PARM keyword of the EXEC statement. The Dictionary would test for and find the production level structure, and will not be aware of the existence of the test level version.

You can continue to modify and promote versions of the language data structure. By altering the STATUSLIST or by appropriate deletions, the Dictionary will execute a STRUCTURES_OUT command on the appropriate test or production level subject.

Establishing Other Command List Parameters

The combination of host language and data structure program usage (IMS/VS has name length restrictions), specified with Language Preprocessor input operands HOST and DLI, allows you to specify the appropriate LANG keyword for any STRUCTURES_OUT commands in the command list. In this way, a stored language data structure can be extracted from the Dictionary with appropriate source language syntax, although the command list does not contain STRUCTURES_OUT commands with a precoded LANGUAGE keyword.

Remember, only the STRUCTURES_OUT and PUNCH commands are permitted in command lists that are executed as the result of a Language Preprocessor request. For your test activities, you may use output routing to direct the command output to a device other than DDPUNCH and, in doing so, specify a DEST operand within the command list. However, for Language Preprocessor execution, the commands in the command list must use the default destination to the punch, or be explicitly coded DEST=P.
Establishing an Extensibility Category for Language Preprocessor Use

In a previous section you noticed the category "CLIST" was used—not a standard Dictionary category. You can use such an installation-defined category to control the use of command lists for Language Preprocessor use. The category can be any name you choose; not just "CLIST". When the Language Preprocessor is executed, the end user would specify, as part of the CLIST option for the EXEC statement, the suboperand CAT (CLIST). Alternatively, this category name could be in the Default Options Table, so that the end user does not have to specify the operand.

Because the only data a subject needs to contain command lists is User Data, the definition of a category like CLIST is simple. In fact, all you have to do is establish a category name and number pair in production status and then issue the INSTALL command. For example:

ADD CATEGORY (P,,CLIST,0) SBJCODE=101 DATE=111182 +
   DESC=('SUBJECTS IN THIS CATEGORY +
        ARE USED TO STORE COMMAND LISTS') ;
INSTALL CATEGORY (P,,CLIST,0) UPDATE=YES RP"=GUIDE DEST=L ;

You might add some description text and EXPLAIN text before installing the category, although this text data can be added afterward if needed. If you decide you need an alias name for the category, you can establish one with the ADD__RELATIONSHIP command before the INSTALL command is used.

----

* * * * * * *

DB/DC DATA DICTIONARY REPORT 01/04/34 09:23:16
CATEGORY GUIDE PAGE: 01

CATEGORY: CLIST

DESCRIPTION:

SUBJECTS IN THIS CATEGORY ARE USED TO STORE COMMAND LISTS

SUBJECT NAME RULES: DATE: 111182
NAME TYPE: N
MIN LENGTH: 01
MAX LENGTH: 31
VAL ROUTINE:

ATTRIBUTES:

REPEAT LENGTH DATA VALIDATION
--------- ------- ------- ---- ------- ------- ------- -------

* * * * * * *

DB/DC DATA DICTIONARY REPORT 01/04/83 09:23:16
CATEGORY GUIDE PAGE: 02

RELATIONSHIP TYPES:

FORWARD NAME TO CATEGORY INVERSE NAME SEQOPT SEQATTR DIRECTED
--------- --------- ------- ------- ------- ------- -------

END-OF-REPORT

* * *

Figure 115. GUIDE Report for Category Used for Command Lists
Testing Command List Readiness

An important step in preparing the Dictionary for use with the Language Preprocessor is to ensure that the correct language data structures, at the appropriate level, are extracted from the Dictionary. You need to develop testing procedures to do this.

You do not, however, have to execute the Language Preprocessor itself to check whether a language data structure is extracted correctly. If you can see the result of a RUNCLIST command executing a command list, you can determine whether the command output usually returned to the Language Preprocessor is correct.

The Dictionary allows you to use the same command features, the use of indefinite status codes in both the SETCLIST and in entries in command lists, so that status concatenation can be tested.

The actions shown in Figure 116 give equivalent results.

<table>
<thead>
<tr>
<th>Command and Operands</th>
<th>Processor Default Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETCLIST</td>
<td></td>
</tr>
<tr>
<td>category name</td>
<td>CLIST CAT</td>
</tr>
<tr>
<td>subject status</td>
<td>STATUS CODE</td>
</tr>
<tr>
<td>code</td>
<td>OCCURRENCE</td>
</tr>
<tr>
<td>occurrence</td>
<td>LIBR-SEPCHAR-member*</td>
</tr>
<tr>
<td>user name</td>
<td>CLIST UDNO</td>
</tr>
<tr>
<td>UDNO</td>
<td>LINENO</td>
</tr>
<tr>
<td>LINENO</td>
<td>STATUSLIST</td>
</tr>
<tr>
<td>STATUSLIST</td>
<td></td>
</tr>
<tr>
<td>RUNCLIST</td>
<td></td>
</tr>
<tr>
<td>STOPON</td>
<td>FLAG</td>
</tr>
<tr>
<td>LANG</td>
<td>HOST COBOL, PLI, ASM DLI (or NODLI)</td>
</tr>
<tr>
<td>STRUCTURES_OUT</td>
<td></td>
</tr>
<tr>
<td>subject status</td>
<td>STATUSLIST</td>
</tr>
<tr>
<td>LANG</td>
<td>HOST COBOL, PLI, ASM DLI (or NODLI)</td>
</tr>
<tr>
<td>PUNCH</td>
<td></td>
</tr>
<tr>
<td>subject status</td>
<td>STATUSLIST</td>
</tr>
</tbody>
</table>

* Member cannot be specified as a default option. It comes from the member name specified in the inclusion statement.

Figure 116. Comparison of Equivalent Command and Processor Default Options

From the above, you can see that by specifying SETCLIST commands, using the & for status code if required, you can simulate the Language Preprocessor automatic generation. The RUNCLIST command with its two keywords can also be simulated. Entries in the command list will correspond to those required to produce the data implied by the content of the inclusion statement.
Preparing Language Preprocessor Information for End Users

When you are preparing information for end users about the use of the Language Preprocessor, you need to decide what level of detail is desirable. One approach would be to install the Language Preprocessor with appropriate defaults within the compiler or assembler procedures so that no input operands or Dictionary information is necessary. All Language Preprocessor users would need access to *OS/VS DB/DC Data Dictionary Messages and Codes*, although it might be possible to limit this access to only the chapter containing Language Preprocessor messages.

Depending on the end users, you could add to their flexibility of use by giving information about the input operands and perhaps a brief summary of how to interpret the content of SYSPRINT. In this case, you would not have to describe all possible input operands, only those you assess as being relevant. When you are subsetting information, select only information applicable to the host source language used by that group. If the users will use more than one source language, it might be better not to combine the information. For detailed information about code requirements, see “Giving Information about Code Requirements” on page 331.

Explaining How to Interpret Language Preprocessor Output

It is important that the Language Preprocessor user realizes the value of the preprocessor pass through the source code. The following suggestions outline the type of information that would encourage the productive use of the Language Preprocessor.

- Develop a small program in the host source language.
  
  You can include inclusion statements that either do or do not get resolved by extracting data from the Dictionary.
  
  You could include comments in the program that highlight the Language Preprocessor actions.

- Distribute a sample SYSPRINT listing.
  
  You could annotate the listing to emphasize such things as inclusion statements, recognition exceptions, diagnostics that show record numbers and starting column numbers, and the summary statistics.

- Develop a list of the type of syntax errors detected.
  
  Point out the various conditions that could exist, such as debugging mode for COBOL.

After the Language Preprocessor user gets used to the compiler-like messages and has some experience in looking through the SYSPRINT output, your original instructions won’t be used as much. You can answer any further questions on an informal basis, using the information given earlier in the coding requirements sections.
If you decide not to include Dictionary messages in the SYSPRINT listing, you might need to have the DDLIST data set contents available to help interpret the Language Preprocessor actions. You might specify the DDLIST data sets as generation data sets, or have the operations staff route the output to Dictionary Administration. However, it should not be necessary to provide any explanation of the Dictionary messages regarding command list processing to the general user.

Reference Material for the Language Preprocessor

Giving Information about Code Requirements

This section gives the following reference information for each of the source languages, COBOL, PL/I, and Assembler:

- Syntax of inclusion statements recognized by the Language Preprocessor
- Restrictions about the detection capabilities
- Any special handling performed for inclusion statements
- Any special source language considerations

Inclusion Statements in COBOL Programs

The following syntax rules and conditions apply:

Inclusion Statement Syntax

The syntax of inclusion statements in COBOL programs is:

\[
\text{COPY member-name [ \{ IN | OF \} library-name ]}.
\]

- There must be a blank preceding COPY and a blank following the period. There must be one or more blanks between COPY and member-name, member-name and IN (or OF), and IN (or OF) and library-name. There may be one or more blanks between member-name and the period, and library-name and the period.

- Each of the two names (member-name and library-name) must be from 1 to 30 EBCDIC characters in length and be composed using the characters 'A' through 'Z', '0' through '9', and the hyphen. A hyphen must not be the first or last character.

- The statement must not have more than one continuation line.

- The statement must not have embedded comment lines.
Recognition Exceptions

The Language Preprocessor will recognize and process an inclusion statement in any context except within:

- An EXECUTE statement (see "Other COBOL Language Considerations")
- A nonnumeric literal (delimited by the quotation mark character, which is determined by the QUOTE/APOST input operand)
- An EGCS literal (delimited by "G<quotation mark><graphic left delimiter>" and "<graphic right delimiter><quotation mark>," where <graphic left delimiter> and <graphic right delimiter> are obtained from the Default Options Table and <quotation mark> is determined by the QUOTE/APOST input operand)
- Pseudotext (delimited by '==')
- A comment line ('**' or '/-' in column 7)

Note that the above implies that the Language Preprocessor will recognize and process an inclusion statement within a comment entry in the Identification Division or a NOTE (which is an obsolete form of comment).

Anything in a COBOL program that is not a valid preprocessor statement will be copied unchanged to SYSPUNCH by the Language Preprocessor.

Special Handling for Inclusion Statements

To be consistent with the treatment of the member name and library name by the COBOL compiler, the following transformations will be performed upon the names before translation of the inclusion statement:

- Each name is truncated to 8 bytes.
- All hyphens are replaced by '0'.
- If the first character of a name is '0', it is replaced by 'J'.
- If the first character of a name is '1' through '9', it is replaced by 'A' through 'I', respectively.

Inclusion statements will be translated using the 1974 ANS interpretation of the COBOL COPY statement and not the 1968 ANS interpretation. That is, the first data name in the retrieved source code will not be replaced by the member name.

Other COBOL Language Considerations

- When the WITH DEBUGGING MODE clause is specified in the SOURCE-COMPUTER paragraph of the Configuration Section, debugging lines ('D' in column 7) are treated as standard lines (blank in column 7); otherwise, they are treated as comment lines (asterisk in column 7).
• An inclusion statement within an EXECUTE statement will not be recognized and processed by the Language Preprocessor. The opening delimiter of an EXECUTE statement is a blank followed by EXECUTE (or EXEC), followed by a blank. The closing delimiter of an EXECUTE statement is a blank followed by END-EXEC, which is followed by either a blank, a comma and a blank, a semicolon and a blank, or a period and a blank. EXECUTE (or EXEC) must be entirely within Area B of a single line in the Data Division or Procedure Division, and likewise for END-EXEC. The closing delimiter will be recognized in any context within an EXECUTE statement except within:

- A literal delimited by quotation marks
- A literal delimited by apostrophes
- An EGCS literal
- A comment line

InclusionStatements in PL/I Programs

The following syntax rules and conditions apply:

Inclusion Statement Syntax

The syntax of inclusion statements in PL/I programs is:

```%
%INCLUDE { member-name | library-name(member-name) } ;
```

• There may be one or more blanks between the percent sign and INCLUDE, member-name and the semicolon, library-name and the left parenthesis, the left parenthesis and member-name, member-name and the right parenthesis, and the right parenthesis and the semicolon. There must be one or more blanks between INCLUDE and member-name, and INCLUDE and library-name. INCLUDE must be entered in uppercase.

• Each of the two names (member-name and library-name) must be from 1 to 8 EBCDIC characters in length and be composed using the characters 'A' through 'Z', '0' through '9', '@', '#', and '$'. The first character must be 'A' through 'Z', '@', '#', or '$'.

• The statement must not span more than two lines.

• The statement must not have embedded comments.
Recognition Exceptions

The Language Preprocessor will recognize and process an inclusion statement in any context except within:

- An EXECUTE statement (see “Other PL/I Language Considerations”).

- A character string (delimited by apostrophes).

- A graphic string (delimited by “<graphic left delimiter><graphic quotation mark>” and “<graphic quotation mark><graphic G><graphic right delimiter>,” where <graphic left delimiter>, <graphic right delimiter>, <graphic quotation mark>, and <graphic G> are obtained from the Default Options Table). DBCS users should not confuse the "<" and ">" symbols used here for shift-out and shift-in symbols.

- A comment (delimited by '/*' and '*/')

Anything in a PL/I program that is not a valid preprocessor statement will be copied unchanged to SYSPUNCH by the Language Preprocessor.

Other PL/I Language Considerations

- The Language Preprocessor will recognize graphic strings within comments (delimited by “<graphic left delimiter>” and “<graphic right delimiter>”). A graphic character must not be split between two lines.

- A Language Preprocessor statement within an EXECUTE statement will not be recognized and processed by the Language Preprocessor. The opening delimiter of an EXECUTE statement is EXECUTE (or EXEC) followed by a blank. EXECUTE (or EXEC) must not be split between two lines and must not be immediately preceded by a character that is 'A' through 'Z', '0' through '9', '@', '#', '$', or '_'.

The closing delimiter of an EXECUTE statement is a semicolon. The closing delimiter will be recognized in any context within an EXECUTE statement except within:

- A character string delimited by quotation marks
- A character string delimited by apostrophes
- A graphic string
- A comment

Graphic strings within comments within EXECUTE statements will be recognized.
Inclusion Statements in Assembler Programs

The following syntactical rules and conditions apply:

Inclusion Statement Syntax

The syntax of Language Preprocessor statements in Assembler programs is:

COPY member-name [ remarks ]

- **COPY** is the operation entry and **member-name** is the operand entry. A remarks entry is optional. A name entry, that is, a label, is not permitted.

- **member-name** must be from 1 to 8 EBCDIC characters in length and be composed using the characters, 'A' through 'Z', '0' through '9', '@', '#', and '$'. The first character must be 'A' through 'Z', '@', '#', or '$'.

- The statement must not span more than two lines.

The Language Preprocessor will recognize and process an inclusion statement within any context. This implies that conditional assembly instructions will have no effect on the Language Preprocessor. (OPSYN instructions will not be recognized.)

Anything in an Assembler program that is not a valid preprocessor statement will be copied unchanged to SYSPUNCH by the Language Preprocessor.

Other Assembler Considerations

The Language Preprocessor will support variable margins by recognizing the following form of an Assembler ICTL statement:

ICTL b[e,c] [ remarks ]

- **ICTL** is the operation entry. The operand entry consists of one to three decimal self-defining terms, where **b** specifies the begin column (the left margin), **e** specifies the end column (the right margin), and **c** specifies the continue column. A remarks entry is optional. A name entry is not permitted.

- **b** must be greater than 0 and less than 41. If **e** is specified, it must be greater than 40, less than 81, and greater than **b** + 4. If **e** is specified, it must be less than 41 and greater than **b**.

- The **ICTL** statement must not span more than two lines.

- The **ICTL** statement must be the first statement in the source program.
• If e is not specified or e is 80, then continuations are not permitted; otherwise, continuations are allowed and the continuation of a line is indicated by a nonblank character in column e + 1.

• If e is not specified, it defaults to 71.

If the Language Preprocessor does not find a valid ICTL statement, 1,71,16 will be used for begin, end, and continue columns, respectively.

Examples of Language Preprocessor Use

The following examples show a segment of a COBOL, PL/I, and an Assembler program as used with the Language Preprocessor.

These examples assume that the following defaults are contained in the Default Options Table:

• SEPCHAR - '!'  
• SIGNON - null sign-on  
• MODE - BATCH  
• CLIST
  - CATEGORY - SEG  
  - CODE - C  
  - LINENO - 1  
  - OCCURRENCE - 0  
  - STATUS - T  
  - UDNO - 1  
• DDLIST - NODDLIST  
• DLI - DLI  
• EXIT - null user exit routine name  
• FLAG - I  
• HOST - COBOL  
• LIBRARY - SYSLIB  
• OPTIONS - OPTIONS  
• QUOTE - QUOTE
- SEARCHRULE - BOTH
- SOURCE - SOURCE
- STATUSLIST - TP

Example of Language Preprocessor Use with COBOL

In this example, the Language Preprocessor is going to be used to provide programmers with a data description about a bank account (name, id, balance, and so forth). The following identifies the data items by their COBOL language names and by their corresponding subject names in the Dictionary.

<table>
<thead>
<tr>
<th>COBOL Names</th>
<th>Dictionary Subject Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISKREC</td>
<td>SEGMENT (P,C,DISKREC,0)</td>
</tr>
<tr>
<td>REC-ID</td>
<td>ELEMENT (P,C,REC-ID,0)</td>
</tr>
<tr>
<td>DISK-FLD</td>
<td>ELEMENT (P,C,DISK-FLD,0)</td>
</tr>
<tr>
<td>DISK-NAME</td>
<td>ELEMENT (P,C,DISK-NAME,0)</td>
</tr>
<tr>
<td>DISK-BAL</td>
<td>ELEMENT (P,C,DISK-BAL,0)</td>
</tr>
</tbody>
</table>

It is assumed that the data base administrator has already created a structure for the record DISKREC and that the Dictionary does not contain an entry for SEG (T,C,DISKREC,0).

The subject SEG (P,C,DISKREC,0) contains the following in User Data 1:

1  STRUCTURES_OUT SEG (&,C,DISKREC,0)
2  ENDDATA
   .
   .

Because the subject SEG (T,C,DISKREC,0) does not exist, the following options are appropriate for the PARM keyword of the EXEC statement:

HOST(COBOLO), DDLIST, CLIST (STATUS(&)), LIBRARY
Shown below is a segment of a COBOL program from SYSIN that uses the DISKREC structure:

```
DATA DIVISION.
FILE SECTION.
FD INDEXED-FILE LABEL RECORDS STANDARD.
COPY DISKREC.
FD IN-FILE LABEL RECORDS STANDARD.
  01 IN-RECORD.
    02 IN-KEY PICTURE X(10).
    02 IN-NAME PICTURE X(10).
    02 IN-BAL PICTURE S99999V99.
WORKING-STORAGE SECTION.
```

Upon receiving the above program from SYSIN, the Language Preprocessor places the following commands into DDINPUT and invokes the Dictionary:

```
SETLOGI=Y;
SETCLIST SEG (&,C,DISKREC,0) +
UDNO=1 LINENO=1 STATUSLIST=IP;
RUNCLIST STOPON=E LANG=C;
```

The Dictionary uses the status list (obtained from the Defaults Option Table) to determine the status of the command list subject. Because SEG (T,C,DISKREC,0) does not exist, SEG (P,C,DISKREC,0) is the subject of the command list. The Dictionary then processes the command list in User Data 1 of SEG (P,C,DISKREC,0) starting at line number 1. Because the STRUCTURES_OUT command in the command list contains a subject with an indefinite status, the Dictionary replaces this status by the replacement status determined for the subject containing the command list, which was 'P'. In addition, the Dictionary appends "LANG=C" to the command. The following command is then executed:

```
STRUCTURES_OUT SEG (P,C,DISKREC,0) LANG=C;
```

Output to DDPUNCH is:

```
./ ADD NAME=NONE
./ NUMBER NEW1=100, INCR=100
  01 DISKREC.
    03 REC-ID PIC X(10).
    03 DISK-FLD1 PIC X(10).
    03 DISK-NAME PIC X(20).
    03 DISK-BAL PIC S99999V99.
```

338 Administration and Customization Guide
Output to DDLIST is:

10:20 03/28/83 INPUT RECORD /SETLOGI=Y; /  
DBD0026 I OK**  
10:20 03/28/83 INPUT RECORD /SETCLIST SEG ($,C,DISKREC,0) + /  
10:20 03/28/83 INPUT RECORD /UDNO=1 LINENO=1 STATUSLIST=TP; /  
10:20 03/28/83 INPUT RECORD /RUNCLIST STOPON=E LANG=C; /  
10:20 03/28/83 INPUT RECORD /STRUCTURES_OUT SEG (P,C,DISKREC,0) LANG=C; /  
DBD0728 I MEMBER NAME DEFAULTED TO 'NONAME'  
DBD0735 I DVALUES OPTON DEFAULTED TO 'NO'  
DBD0734 I COMpress OPTION DEFAULTED TO 'NO'  
DBD0737 I LEVEL OPTON DEFAULTED TO '1'  
DBD0736 I NARRATIVE OPTON DEFAULTED TO 'YES'  
DBD0726 I STRUCTURES OUT COMPLETE  
DBD6401 I CLIST PROCESSING COMPLETE  

Dictionary return code: 0

Because no user exit routine name has been specified in the input operands, the Language Preprocessor will not call a user exit routine. The Language Preprocessor inserts DDPUNCH into the source program as follows:

```

DATA DIVISION.
FILE SECTION.
FD INDEXED-FILE LABEL RECORDS STANDARD.
•
*COPY DISKREC.
*
01 DISKREC.
  03 REC-ID PIC X(10).
  03 DISK-FLD1 PIC X(10).
  03 DISK-NAME PIC X(20).
  03 DISK-BAL PIC $99999999.
FD IN-FILE LABEL RECORDS STANDARD.
01 IN-RECORD.
  02 IN-KEY PICTURE X(10).
  02 IN-NAME PICTURE X(10).
  02 IN-BAL PICTURE $99999999.
WORKING-STORAGE SECTION.
```
The output to SYSPRINT IS:

```
COPY DISKREC.
```

Because DDLIST was specified as an input operand, the Dictionary listing output gets copied to the Language Preprocessor listing.
Example of Language Preprocessor Use with PL/I

The Language Preprocessor is used in this example to supply PL/I programmers with data declarations for some meteorological data. The primary difference between this example and the last is that the STRUCTURES__OUT is stored in a CLIST subject rather than in a SEGMENT subject. The following table shows the names of the data items as they would appear in a program and as they are known to the Dictionary.

<table>
<thead>
<tr>
<th>PL/I Names</th>
<th>Dictionary Subject Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEATHER</td>
<td>SEGMENT (T,B,WEATHER,0)</td>
</tr>
<tr>
<td>TEMPERATURE</td>
<td>ELEMENT (T,B,TEMPERATURE,0)</td>
</tr>
<tr>
<td>HIGH</td>
<td>ELEMENT (T,B,HIGH,0)</td>
</tr>
<tr>
<td>LOW</td>
<td>ELEMENT (T,B,LOW,0)</td>
</tr>
<tr>
<td>WIND__VELOCITY</td>
<td>ELEMENT (T,B,WIND__VELOCITY,0)</td>
</tr>
<tr>
<td>MAX</td>
<td>ELEMENT (T,B,MAX,0)</td>
</tr>
<tr>
<td>MIN</td>
<td>ELEMENT (T,B,MIN,0)</td>
</tr>
<tr>
<td>PRECIPITATION</td>
<td>ELEMENT (T,B,PRECIPITATION,0)</td>
</tr>
<tr>
<td>TOTAL</td>
<td>ELEMENT (T,B,TOTAL,0)</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>ELEMENT (T,B,AVERAGE,0)</td>
</tr>
</tbody>
</table>

This example assumes that a structure has already been created for SEG (T,B,WEATHER,0) and that the Dictionary does not contain entries for SEG (5,B,WEATHER,0) or SEG (7,B,WEATHER,0).

The subject CLIST (P,,LIB1.WEATHER,0) contains the following in User Data 3:

```
100 STRUCTURES.OUT SEG (&,B,WEATHER,0)
101 ENDDATA
```

The options that should be specified for the PARM keyword of the EXEC statement are:

```
HOST(PLI),CLIST(CAT(CLIST),CODE,STATUS(P),LINENO(100),UDNO(3)),
(EXIT(EXITRT1),STATUSLIST(57T))
```

Note that the "P" value for STATUS refers to the status code of the subject CLIST (P,,LIB1.WEATHER,0) and that the STATUSLIST values refer to the status code of SEG (&,B,WEATHER,0).
Shown below is a part of a PL/I program from SYSIN that processes the WEATHER structure:

```
.
.
.
DECLARE  I TVARS,
     2 A          BINARY       FIXED(31),
     2 B          BINARY       FIXED(31),
     2 C          BINARY       FIXED(31);
%INCLUDE LIB1(WEATHER);
DECLARE M(10,10) BINARY FIXED(31);
.
.
```

After accepting the above program, the Language Preprocessor places the following commands into DDINPUT and invokes the Dictionary:

```
SETCLIST CLIST (P,,LIB1,WEATHER,0) +
UDNO=3 LINENO=100 STATUSLIST=57T;
RUNCLIST STOPON=E LANG=B;
```

The Dictionary, upon encountering the indefinite status in the subject of the STRUCTURES _OUT command, uses the STATUSLIST values to determine the status. The Dictionary first attempts to locate SEG (5,B,WEATHER,0) and then SEG (7,B,WEATHER,0). Because neither subject exists, and because SEG (T,B,WEATHER,0) does exist, the Dictionary executes this command:

```
STRUCTURES_OUT SEG (T,B,WEATHER,0) LANG=B;
```

Because a nonnull user exit routine name has been specified in the input operands, the Language Preprocessor calls the routine, EXTRT1, for each line in DDPUNCH before inserting it into SYSPUNCH. The following is the result:

```
DCL
1 WEATHER, 
  3 TEMPERATURE,  
  5 HIGH DEC FIXED(4,1),
  5 LOW DEC FIXED(3,1),
  3 WIND VELOCITY, 
  5 MAX DEC FIXED(3),
  5 MIN DEC FIXED(2),
  3 PRECIPITATION,
  5 TOTAL DEC FIXED(3,1),
  5 AVERAGE DEC FIXED(3,1)
/* END OF STRUCTURE */;
```
The Language Preprocessor inserts DDPUNCH into the source program as follows:

DECLARE 1 TVARS,
  2 A  BINARY  FIXED(31),
  2 B  BINARY  FIXED(31),
  2 C  BINARY  FIXED(31);

/*
%INCLUDE LIB1(WEATHER);
*/

DCL
  1 WEATHER,
  3 TEMPERATURE,
  5 HIGH DEC FIXED(4,1),
  5 LOW DEC FIXED(3,1),
  3 WIND_VELOCITY,
  5 MAX DEC FIXED(3),
  5 MIN DEC FIXED(2),
  3 PRECIPITATION,
  5 TOTAL DEC FIXED(3,1),
  5 AVERAGE DEC FIXED(3,1)
/* END OF STRUCTURE */;
DECLARE M(10,10)  BINARY  FIXED(31);

The output on SYSPRINT is:

63  %INCLUDE LIB1(WEATHER);
DBDLP01 W LINE  63 COLUMN  2 RETURN CODE IS  0 FOR COMMAND LIST
CLIST (P,,LIB1.WEATHER,0) UDNO=3 LINENO=100
  .
  .
Chapter 13. Administering the Dictionary Language Preprocessor  343
Example of Language Preprocessor Use with Assembler

In this example, the Language Preprocessor uses both the STRUCTURES__OUT and the PUNCH commands to generate source code for a program. The following table shows the data items that the STRUCTURES__OUT command will produce.

<table>
<thead>
<tr>
<th>Assembler Names</th>
<th>Dictionary Subject Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMVAR</td>
<td>SEGMENT (P,A,COMVAR,0)</td>
</tr>
<tr>
<td>VAR1</td>
<td>ELEMENT (P,A,VAR1,0)</td>
</tr>
<tr>
<td>VAR2</td>
<td>ELEMENT (P,A,VAR2,0)</td>
</tr>
</tbody>
</table>

It is assumed that a structure exists for SEG (P,A,COMVAR,0). The PUNCH command will be used to supply the program with register equates. The equates are contained in subject (P,REGEQUI,0) of the ASMCCDE category.

User Data 1 of subject CLIST (P,SYSLIB.COMVAR,0) contains the two commands:

```
1  PUNCH ASMCCDE (P,REGEQUI,0) UDNO=1
2  STRUCTURES_OUT SEG (P,A,COMVAR,0)
3  ENDDATA
    .
    .
```

These are the options that are specified on the PARM keyword of the EXEC statement:

```
HOST(ASM),DIDLIST,NODLI,SEARCHRULE( BOTH),
CLIST(CAT(CLIST),STATUS(P),CODE)
```

Because a status code is explicitly coded in the STRUCTURES__OUT and PUNCH commands, the Dictionary will not make use of the TP default for STATUSLIST. That is, even if a SEGMENT subject (T,A,COMVAR,0) existed, it would not be used.
Shown below is a segment of an Assembler program from SYSIN:

```
.
.
ZERO EQU 0
ONE EQU 1
    COPY COMVAR
A1 DS F
A2 DS F
    COPY PAYSEG
COST DS F
.
.
```

The subject CLIST (P,,SYSLIB.PAYSEG,0) does not exist.

When processing the first inclusion statement, the Language Preprocessor places the following commands in DDINPUT and invokes the Dictionary:

```
SETLOGI=Y;
SETCLIST CLIST (P,,SYSLIB.COMVAR,0) +
UDNO=1 LINENO=1 STATUSLIST=TP;
RUNCLIST STOFON=E LANG=J;
```

The Dictionary executes the PUNCH command and then appends "LANG=J" to the STRUCTURES_OUT command before executing it. The following is the result:

```
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
./ ADD NAME=NONAME
./ NUMBER NEW=100,INCR=100
COMVAR DS CL8
    ORG COMVAR
VAR1 DS CL4
VAR2 DS CL4
```
Output to DDLIST is:

10:23 03/28/83 INPUT RECORD /SETLOC.=Y; /DBD0026 I OK**
10:23 03/28/83 INPUT RECORD /SETCLIST CLIST (P,,SYSLIB.COMVAR,0) + /DBD0735 I DVALUES OPTION DEFAULTED TO 'NO'
10:23 03/28/83 INPUT RECORD /UDNO=1 LINENO=1 STATUSLIST=TP; /DBD0734 I COMPRESS OPTION DEFAULTED TO 'NO'
10:23 03/28/83 INPUT RECORD /RUNCLIST STOPON=E LANG=J; /DBD0737 I LEVEL OPTION DEFAULTED TO '1'
10:23 03/28/83 INPUT RECORD /PUNCH ASM CODE (P,,REGEQU,0) UDNO=1 /DBD0736 I NARRATIVE OPTION DEFAULTED TO 'YES'
10:23 03/28/83 INPUT RECORD /STRUCTURES OUT SEG (P,A,COMVAR,0) LANG=J; /DBD0726 I STRUCTURES OUT COMPLETE
DBD6401 I CLIST PROCESSING COMPLETE

Dictionary return code: 0

When processing the second preprocessor statement, the Language Preprocessor places the following commands in DDINPUT and invokes the Dictionary:

SETLOGI=Y;
SETCLIST CLIST (P,,SYSLIB.PAYSEG,0) +
UDNO=1 LINENO=1 STATUSLIST=TP;
RUNCLIST STOPON=E LANG=J;

Because the subject containing the command list does not exist, the Dictionary returns a code of 8.

DDLIST will contain the following:

10:23 03/28/83 INPUT RECORD /SETLOGI=Y; /DBD0026 I OK**
10:23 03/28/83 INPUT RECORD /SETCLIST CLIST (P,,SYSLIB.PAYSEG,0) + /DBD0735 I DVALUES OPTION DEFAULTED TO 'NO'
10:23 03/28/83 INPUT RECORD /UDNO=1 LINENO=1 STATUSLIST=TP; /DBD0734 I COMPRESS OPTION DEFAULTED TO 'NO'
10:23 03/28/83 INPUT RECORD /RUNCLIST STOPON=E LANG=J; /DBD0737 I LEVEL OPTION DEFAULTED TO '1'
DBD6405 E COMMAND LIST CLIST (P,,SYSLIB.PAYSEG,0) NOT FOUND
10:23 03/28/83 INPUT RECORD /RUNCLIST STOPON=E LANG=J; /DBD6403 E SETCLIST COMMAND HAS NOT BEEN ISSUED

Dictionary return code: 8

Because SEARCHRULE(both) was specified in the input operands, the Language Preprocessor will copy the inclusion statement unchanged to SYSPUNCH.
The following is a result of the translation of both inclusion statements by the Language Preprocessor as output to SYSPUNCH:

```
.
.
ZERO EQU 0
ONE EQU 1
*
* COPY COMVAR
*
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
COMVAR DS CL8
    ORG COMVAR
VAR1 DS CL4
VAR2 DS CL4
A1 DS F
A2 DS F
    COPY PAYSEG
COST DS F
.
.
```
The output to SYSPRINT is:

. .

115 COPY COMVAR
DBDLPO1 I LINE 115 COLUMN 10 RETURN CODE IS 0 FOR COMMAND LIST
CLIST (P,,SYSLIB.COMVAR,0) UDNO=1 LINENO=1
10:23 03/28/83 INPUT RECORD /SETCLIST CLIST (P,,SYSLIB.COMVAR,0) + /  
10:23 03/28/83 INPUT RECORD /UDNO=1 LINENO=1 STATUSLIST=TP;  
10:23 03/28/83 INPUT RECORD /RUNCLIST STOPON=E LANG=J;  
10:23 03/28/83 INPUT RECORD /PUNCH ASMCODE (P,,REGEQU,0) UDNO=1  
10:23 03/28/83 INPUT RECORD /STRUCTURES_OUT SEG (P,A,COMVAR,0) LANG=J;  
DBD0728 I MEMBER NAME DEFAULTED TO 'NONAME'
DBD0735 I DVALUES OPTION DEFAULTED TO 'NO'
DBD0734 I COMPRESS OPTION DEFAULTED TO 'NO'
DBD0737 I LEVEL OPTION DEFAULTED TO '1'
DBD0736 I NARRATIVE OPTION DEFAULTED TO 'YES'
DBD0726 I STRUCTURES OUT COMPLETE
DBD6401 I CLIST PROCESSING COMPLETE

118 COPY PAYSEG
DBDLPO1 W LINE 118 COLUMN 10 RETURN CODE IS 8 FOR COMMAND LIST
CLIST (P,,SYSLIB.PAYSEG,0) UDNO=1 LINENO=1
10:23 03/28/83 INPUT RECORD /SETCLIST CLIST (P,,SYSLIB.PAYSEG,0) + /  
10:23 03/28/83 INPUT RECORD /UDNO=1 LINENO=1 STATUSLIST=TP;  
DBD6405 E COMMAND LIST CLIST (P,,SYSLIB.PAYSEG,0) NOT FOUND
10:23 03/28/83 INPUT RECORD /RUNCLIST STOPON=E LANG=J;  
DBD6403 E SETCLIST COMMAND HAS NOT BEEN ISSUED
Chapter 14. Transferring Dictionary Data

This chapter describes the facilities available for extracting data from a central or “source” Dictionary and for transferring that data to a “receiving” Dictionary. The extracted data is in a form that can be easily understood, edited, and previewed for its effect on the receiving Dictionary.

The task of transferring data allows you to extend the availability of Dictionary definitions beyond a single data base:

- A receiving Dictionary can be initialized with data extracted from a centrally maintained Dictionary.

- Definitions that are independently developed within a Dictionary at another location can be incorporated into a central receiving Dictionary.

To specify what data is to be extracted you need only use one general purpose command. This command, EXPORT, generates a stream of Dictionary update commands that can be used as a batch input stream to the receiving Dictionary. The EXPORT command also provides a processing report so that you can trace how subjects in a Dictionary structure are being selected and determine when relationships are not followed.

You can export standard Dictionary categories, installation-defined categories, and extensibility control information categories. The EXPORT command generates ADD and ADD__RELATIONSHIP commands and batch forms images for the data to be unloaded. The commands and batch forms images will re-create the selected data. The batch forms images are used to unload text data for individual subjects. If extensibility control information categories are being unloaded, you can request that INSTALL commands be generated to install the CATEGORY, RELTYPE, and STRTYPE categories. The ATTRTYPE category may be installed only as part of a CATEGORY or RELTYPE category.

You can, optionally, cause the EXPORT command to generate CHANGE__IN, CHANGE__RELATIONSHIP__DATA, and DELETE__DATA commands to allow you to change existing data.

The EXPORT command output stream can be submitted to the receiving Dictionary to examine the impact without updating its contents. You can also make changes to the command stream, or data base contents before executing the command stream to receive the data. This is done by using the PREVIEW command in the receiving Dictionary to process a data set containing the generated command stream. The output stream generated by the EXPORT command may be directed to a punch, terminal, or line printer.
This chapter gives you:

- Planning guidance for data transfer activities
- Information about using the EXPORT command
- Information about using the PREVIEW command and validating the data before updating the receiving Dictionary
- General guidance about completing the data transfer

**Planning for Dictionary Data Transfer**

When you need to extract data for use in another Dictionary, your first task is the identification of that data:

- Which subjects and their categories?
- What subject data is to be included?
- What relationships existing in the source Dictionary have to be reestablished?
- How much data is involved?
- Will the data be new to the receiving Dictionary or will it update already existing subjects?

The extracted data can be for subjects in standard categories or in installation-defined categories. Similarly, instances of relationships between those subjects selected for data transfer can also be included. Also, definitions of installation-defined categories, relationship-types, attribute-types, and structure-types can be transferred.

**Initializing a Receiving Dictionary**

You can select from a central repository of Dictionary data an appropriate working set to be made available to users of an independently operated Dictionary.

You do not have to use a complete data base reload strategy and can:

- Select only subjects and relationships required in the working set.
- Select only subjects with appropriate subject name qualifiers, such as a designated status code.
- Select only those installation-defined entities that are required in the receiving Dictionary.

You will need to make installation arrangements for any installation-developed programs or validation routines that are required for use with the receiving Dictionary.
Maintaining Data in a Receiving Dictionary

There are two general types of maintenance activities for a receiving Dictionary:

- Refreshing reference information and definitions that are not directly maintained by the receiving Dictionary users

  For example:
  - A modified production-level data base definition replaces the existing one
  - Additional data that has been added to subject definitions, including subject data for installation-defined categories, is made available to the receiving Dictionary users

- Providing current information from the source Dictionary for use in ongoing development in the receiving Dictionary

  For example:
  - Current specifications for job execution JCL are migrated
  - Language data structures are updated so that they can be accessed by the Dictionary Language Preprocessor

The EXPORT command support can be a valuable tool in performing these maintenance and migration tasks. Integrity of the data is preserved by extracting from the source Dictionary and having this data replace the existing data in the receiving Dictionary. Data entry resources for maintaining the receiving Dictionary are significantly reduced.

Any installation-developed programs or validation routines that have been modified need to be separately reinstalled.

Subject Selection Criteria

Using the EXPORT command, you have the capability of:

- Selecting a single subject
- Selecting a set of subjects defined in a stack
- Selecting all subjects in a category
- Using any of the above under the control of a structure-type

The last of the preceding selection criteria provides a powerful method of extracting many subjects in one export action in addition to the relationships existing between those selected subjects.

You can combine the use of a stack with a specified structure-type. Starting with a given subject, the EXPORT processing follows relationships under the direction of
the given structure-type and selects subordinate subjects. The relationship instances are also generated as ADD_RELATIONSHIP commands.

Similarly, you can combine the specification of all subjects in a category and a specified structure-type. In this case, each subject in the category is used as a starting or "topmost" subject in a Dictionary structure.

**Controlling the Data Extraction Specification**

The summary in Figure 117 shows how the use of Dictionary stacks or "all subjects in a category" can be combined with an AND DEPENDENTS option when you are specifying the action of an EXPORT command. The effect of the AND DEPENDENTS option is similar to that available with the COPY command. When you specify this option, you can use the default structure-type or one established by your installation (specified with the STRUCTURE option).

<table>
<thead>
<tr>
<th>Specification Used</th>
<th>Data Extracted</th>
</tr>
</thead>
<tbody>
<tr>
<td>category + subject name</td>
<td>A single subject.</td>
</tr>
<tr>
<td>category + subject name + AND DEPENDENTS + structure-type</td>
<td>Multiple subjects and relationships. The specified subject and all subjects selected under structure-type control are extracted. The specified subject is used as the &quot;topmost&quot; subject in a Dictionary structure.</td>
</tr>
<tr>
<td>*STACK</td>
<td>Multiple subjects. All subjects in the Dictionary stack are extracted; categories can vary.</td>
</tr>
<tr>
<td>*STACK + AND DEPENDENTS + structure-type</td>
<td>Multiple subjects and relationships. The specified subjects in the Dictionary stack are used as &quot;topmost&quot; subjects and all subjects associated with each one are selected under control of a single structure-type.</td>
</tr>
<tr>
<td>*CAT category</td>
<td>Multiple subjects. All subjects in the category are extracted.</td>
</tr>
<tr>
<td>*CAT category + AND DEPENDENTS + structure-type</td>
<td>Multiple subjects and relationships. All subjects associated with each subject in the category used as the &quot;topmost&quot; subject in a Dictionary structure. Data extraction is under control of a single structure-type.</td>
</tr>
</tbody>
</table>

**Figure 117. Data Extraction Choices Used with the EXPORT Command**
Status Code as a Selection Criteria

Independently of criteria for selecting subjects to be extracted from the source Dictionary, you can specify selection by and assignment of status code values. This capability is to help you control collections of data that are to be incorporated into the receiving Dictionary. The Dictionary status code is used to "partition" data across one or more categories for use by a set of end users.

You can control status codes in two ways:

- Your selection criteria can be adjusted to export only subjects with status codes named in a list.

- Independently of the selection by status code, you can specify a specific status code to be assigned to all the exported subjects.

For example, you could select subjects in status code 'T' and assign status code 'D'.

If you don't assign a status code to be used globally in the output stream, each selected subject's status code remains unaltered (with the exception of subjects in installed status, which will be changed to production status).

Selecting Subject Data

With each Dictionary subject that is extracted, you can include all data associated with that subject. This is the default. However, you can suppress the export of some parts of the data and select one or more of the following choices:

- All or no attribute data
- All or no alias names
- All, none, or part of the Description data
- All, none, or part of the contents of a set of User Data
- All, none, or part of any PL/I text data

Transferring Data and Definitions

For standard categories, you transfer subjects and subject data but not the definitions. The category definitions for standard categories are fixed in the receiving Dictionary.

For installation-defined entities, you can also transfer the definitions in addition to installation-defined subjects, the subject data, and instances of relationships. Refer to "Exporting Extensibility Definitions" on page 369 for more information.
Using EXPORT Processing Reports

One of the features of the EXPORT command is the processing report. This can be used as a form of documentation. The details included in the report are especially useful when exporting a large number of subjects and their relationships. When data extraction is under the control of a structure-type, you can trace how subjects in a Dictionary structure are being selected and determine when relationships are not followed. You can request that a printed version of the structure-type definition be included in the processing report. If you are exporting a number of subjects that were specified in a Dictionary stack, or all subjects in a category, all actions are recorded in a single EXPORT processing report.

The EXPORT command has a DEST operand so that the command stream output can be directed to a printer or terminal destination for initial checking. The DEST operand does not affect the processing report. Output routing is required to alter the processing report destination. Although exporting can be done in the online mode, you may choose to perform the data extraction during a batch run to avoid impacting any online users. This would depend on the amount of data involved.

Editing Output from the EXPORT Command

The EXPORT command generates a stream of standard Dictionary commands and uses batch forms for text data. Comment statements are included in the output before each subject being exported, or each INSTALL command. This output is usual Dictionary punched output and is controlled by the DEST operand.

Using output routing, you can direct the punch output, the extracted data in a command stream, to an OS/VS data set. The generated command stream can then be inspected offline. The editing capabilities available to your installation may alter the data set—within the constraints of the Dictionary command syntax.

Although the placement of command operands in the output stream follows a uniform format, there is no requirement to maintain this format when you perform any editing actions. The format specifications are explained in Appendix C, "Format Specifications for EXPORT Command Output Streams" on page 467. You can delete records or alter subject attribute values as desired. If you make any changes, your only requirement is to follow the Dictionary batch input syntax rules. If changes are made to batch form input, the required header and field placements in individual records must conform to batch form format. Instead of adding subjects that you have determined as missing from the output stream, you may prefer to adjust the EXPORT command specifications and rerun the command so that data conformity with the source Dictionary is preserved.

To assist you in identifying an individual EXPORT command's output, a date and time-stamp record is generated as a comment in the first record in the output stream. Succeeding comment records are also generated. These contain the EXPORT command with defaults inserted to assist in identifying the source of the output. A comment record is also generated for each exported subject.

When exporting a subject that has associated user data, a DBCS user may encounter a special situation. When output in batch form format is generated by the EXPORT command, the User Data text may be split onto two lines (when it is too long to fit on one line). If the data is DBCS data and the split occurs with a
blank inserted before the required continuation character, an unintended space may occur between the two DBCS strings. DBCS users should carefully inspect the output in batch form format before it is executed and correct it, if necessary. A possible solution is to insert a space at the beginning of the line to eliminate the space at the end of the line.

Output Considerations

EXPORT produces a command stream and an EXPORT processing report plus messages. The DEST operand controls only the command stream output.

With Output Routing

With output routing in batch mode, the processing report is directed by an active SETPRINT command. The command stream can be directed by an active SETPUNCH command. Messages are directed to DDLIST and by an active SETPRINT command. See Figure 118 for the EXPORT command destination in batch mode.

<table>
<thead>
<tr>
<th>Types of Output</th>
<th>Output Controlled by DEST Operand</th>
<th>DEST=T</th>
<th>DEST=L</th>
<th>DEST=P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Report</td>
<td></td>
<td>SETPRINT</td>
<td>SETPRINT</td>
<td>SETPRINT</td>
</tr>
<tr>
<td>Command Stream</td>
<td>Command Stream</td>
<td>SETPUNCH</td>
<td>SETPUNCH</td>
<td>SETPUNCH</td>
</tr>
<tr>
<td>Messages</td>
<td></td>
<td>DDLIST/SETPRINT</td>
<td>DDLIST/SETPRINT</td>
<td>DDLIST/SETPRINT</td>
</tr>
</tbody>
</table>

Figure 118. EXPORT Command Destination in Batch Mode with Output Routing

In online mode, the processing report is directed to the terminal and by an active SETPRINT command. The command stream is directed to the terminal or by an active SETPUNCH command. The messages are directed to the terminal and to SPOOL1. In addition, messages may be directed to the SETPRINT destination. See Figure 119 for the EXPORT command destination in online mode.

<table>
<thead>
<tr>
<th>Types of Output</th>
<th>Output Controlled by DEST Operand</th>
<th>DEST=T</th>
<th>DEST=L</th>
<th>DEST=P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Report</td>
<td></td>
<td>SETPRINT &amp; TERMINAL</td>
<td>SETPRINT &amp; TERMINAL</td>
<td>SETPRINT &amp; TERMINAL</td>
</tr>
<tr>
<td>Command Stream</td>
<td>Command Stream</td>
<td>TERMINAL</td>
<td>SETPRINT</td>
<td>SETPUNCH</td>
</tr>
<tr>
<td>Messages</td>
<td></td>
<td>SPOOL1/TERMINAL</td>
<td>SPOOL1/TERMINAL/SETPRINT</td>
<td>SPOOL1/TERMINAL</td>
</tr>
</tbody>
</table>

Figure 119. EXPORT Command Destination in Online Mode with Output Routing
Without Output Routing

Without output routing in batch mode, the processing report, command stream, and messages are sent to DDLIST with the exception of the command stream, which can be directed to DDPUNCH. See Figure 120 for the EXPORT command destination in batch mode.

<table>
<thead>
<tr>
<th>Types of Output</th>
<th>Output Controlled by DEST Operand</th>
<th>DEST=T</th>
<th>DEST=L</th>
<th>DEST=P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Report</td>
<td>Command Stream</td>
<td>DDLIST</td>
<td>DDLIST</td>
<td>DDLIST</td>
</tr>
<tr>
<td>Command Stream Messages</td>
<td>Command Stream</td>
<td>DDLIST</td>
<td>DDLIST</td>
<td>DDPUNCH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DDLIST</td>
<td>DDLIST</td>
<td>DDLIST</td>
</tr>
</tbody>
</table>

Figure 120. EXPORT Command Destination in Batch Mode without Output Routing

In online mode, the processing report and messages are sent to the terminal and SPOOL1. The command stream is sent to the terminal if DEST=T is specified, to SPOOL1 if DEST=L is specified, and to SPOOL2 if DEST=P is specified. See Figure 121 for the EXPORT command destination in online mode.

<table>
<thead>
<tr>
<th>Types of Output</th>
<th>Output Controlled by DEST Operand</th>
<th>DEST=T</th>
<th>DEST=L</th>
<th>DEST=P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Report</td>
<td>Command Stream</td>
<td>SPOOL1 &amp; TERMINAL</td>
<td>SPOOL1 &amp; TERMINAL</td>
<td>SPOOL1 &amp; TERMINAL</td>
</tr>
<tr>
<td>Command Stream Messages</td>
<td>Command Stream</td>
<td>TERMINAL</td>
<td>SPOOL1</td>
<td>SPOOL2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SPOOL1/T</td>
<td>SPOOL1/T</td>
<td>SPOOL1/T</td>
</tr>
</tbody>
</table>

Figure 121. EXPORT Command Destination in Online Mode without Output Routing
Using the EXPORT Command

This section describes the options available with the EXPORT command. Following a summary of the command format, the method by which you combine the various options is discussed.

Format and Options for the EXPORT Command

The general format of this command is:

```
EXPORT
  category subjectname
    *STACK
    *CAT category
  +

Required

Optional
```

```
  AND
  DEPENDENTS
  STRUCTURE=structuretype
  MODE=
    NEW
    UPDATE
  +

  INSTALL=
    NO
    YES
  SELSTAT=
    (status1,...,status10)
  NEWSTAT=status
  +

  DETAIL=
    ALL
    (detail, id1)
    (detail, id1, id2)
    (detail1, id1, ..., detail9)
  PROCRPT=
    NAMES
    detail
    (detail1, ..., detail3)
  +

  STRDEF=
    NO
    YES
  ISR=
    NO
    YES
  DEST=
    P
    L
    T
```

The options for the command are explained below.

**EXPORT**

The command verb. There is no abbreviated form.

category subjectname | *STACK | *CAT category

A required operand that specifies the type of data to be exported.

category subjectname An operand that specifies any valid standard or installation-defined category, and the name of the subject to be exported. For DBCS users, the category name and the user name portion of the subject name must be a uniform string.
*STACK
An operand that specifies that the subjects to be exported are contained in a stack. The location of the stack (either a set of User Data or an OS/VS data set) must be set before this command is executed, via the SETSTACK command.

*CAT category
An operand that specifies that all subjects within a standard or installation-defined category are to be exported. For DBCS users, the category operand must be a uniform string.

AND
An optional operand.

DEPENDENTS
An optional operand that specifies that the structure, beginning with the specified subject, be exported. Multiple structures may be exported with one command when used in conjunction with the *STACK or *CAT options.

STRUCTURE | STRUC=structuretype
An optional operand, used with the AND DEPENDENTS option, that specifies the name of the structure-type to be referenced in determining the "dependents" of the subject to be exported. If omitted, the default structure-type will be used to perform the "dependents" export. DBCS users must enter a uniform string for structure type if this operand is not omitted.

MODE=NEW | UPDATE
An optional operand that specifies the type of commands to be generated in the receiving Dictionary for the named set of subjects. Valid values are:

NEW
Commands will be generated to add subjects and relationships in the receiving Dictionary. This option assumes exporting data to an "empty" Dictionary that does not contain the exported data. NEW is the default.

UPDATE
Commands will be generated to update existing subjects and relationships in the receiving Dictionary.

INSTALL=NO | N | YES | Y
An optional operand that specifies that the command necessary to install the definitions to be exported from the CATEGORY, RELTYPE, or STRTYPE categories, be generated. This keyword is applicable only when the category of the subject being exported is CATEGORY, RELTYPE, or STRTYPE and will be otherwise ignored. NO is the default value if this operand is omitted.

SELSTAT=status | (status1,...,status10)
An optional operand, which specifies that only subjects (and any dependents) with the listed status code(s) are to be exported. If the status code of a subject in a structure does not match one of the codes in the list, neither the subject nor its dependents will be exported. View authorization for the status codes in the list is required if access control is enabled. Up to 10 valid status codes may be specified. If this keyword is omitted, no status code checking will occur.
NEWSTAT=status
An optional operand, which specifies that the status code of all subjects to be
exported be changed to the specified status code. Any valid status code may
be specified, except '*' (installed status). If omitted, no changes to the
status code of exported subjects will be made, except for installed subjects.
In this case, the status code will be changed from '*' to 'P'.

DETAIL=ALL | detail | (detail,id1) | (detail,id1,id2) | (detail,...,detail9)
An optional operand that specifies the amount of data to be exported with
each exported subject. Up to nine detail entries may be specified. Valid
values are:

  ALL Requests that all subject and text data associated with each
         subject be exported.

  ATTR Requests that all subject attributes associated with each subject
         be exported.

  ALIAS Requests that all alias names associated with each subject be
         exported.

  DESC Requests that Description text associated with each subject be
         exported. The values id1,id2 specify the range of lines of text to
         be exported. If both values are specified, id1 must be less than
         or equal to id2. id1 and id2 may be from 1 to 999. All lines of
         text within the specified range will be exported. If only one
         value is specified, only that line of text will be exported. If both
         values are omitted, the entire set of Description text will be
         exported.

  UDn Requests that the set of User Data text (specified as n in UDn),
         be exported. The values id1,id2 specify the range of lines of
         text to be exported. If both values are specified, id1 must be
         less than or equal to id2. id1 and id2 may be from 1 to 999. All
         lines of text within the specified range will be exported. If only
         one value is specified, only that line of text will be exported. If
         both values are omitted, the entire set of User Data text will be
         exported.

  PLI Requests that the PLIDATA text associated with SEGMENT
         and ELEMENT subjects, be exported. The values id1,id2
         specify the range of lines of text to be exported. If both values
         are specified, id1 must be less than or equal to id2. id1 and id2
         may be from 1 to 255. All lines of text within the specified
         range will be exported. If only one value is specified, only that
         line of text will be exported. If both values are omitted, the
         entire set of PLIDATA text will be exported.

If this operand is omitted, the default is ALL.
PROCRPT=NAMES | detail I (detail1,...,detail3)
An optional operand that specifies the amount of information desired in the EXPORT processing report. Up to three detail entries may be specified. Valid values are:

NAMES Requests that only exported subject and alias names be reported.

RELS Requests that only exported relationships be reported.

INST Requests that each exported INSTALL command be reported, if INSTALL=YES has been specified on the EXPORT command.

ALL Requests that all information be reported. This is the same as specifying NAMES, RELS, and INST.

NONE Requests that no EXPORT processing report be generated.

If this operand is omitted, the default is NAMES. If NONE is specified, no other value may be specified.

STRDEF=NO | N | YES | Y
An optional operand that specifies whether or not the structure-type definition specified on the STRUCTURE operand is to be included in the EXPORT processing report. Valid values are:

NO Specifies that the structure-type definition will not be included in the report.

YES Specifies that the structure-type definition is to be included in the processing report, and will precede the list of subjects that were exported.

If this operand is omitted, the default is NO.

ISR=NO | N | YES | Y
An optional operand used with NEWSTAT to change the status of an Indirect Subject Reference. ISR=NO will not change the status of an ISR. ISR=YES will change the status of an ISR.

If omitted, the default is ISR=NO.

DEST=P | L | T
This optional operand is used to specify the destination of the command stream generated by the EXPORT command. Valid values are:

P Punch

L Line Printer

T Terminal

If this operand is omitted, the default is P.
Single subjects may be exported by specifying **category subjectname**. Multiple subjects may be exported by specifying **STACK**. All subjects in a category may be exported by specifying **CAT** or **STACK**.

Structures may be exported by specifying the **AND DEPENDENTS** option of the command. Specify **AND DEPENDENTS** with **category subjectname** to export single structures, and **STACK** and **CAT** to export multiple structures.

Output from execution of the command is a file containing update commands, batch forms images, and **INSTALL** commands. You may assess the impact that execution of this command stream may have on the receiving Dictionary by using the **PREVIEW** command, discussed later in this chapter.

**Exporting Single Subjects**

The simplest form of the **EXPORT** command is that used to extract a single subject from a source Dictionary. As a default, the **EXPORT** command assumes that the subject will be new to the receiving Dictionary. For example:

```
EXPORT DATABASE (P,P,PAYDB,0) ;
```

The above command would cause all of the characteristics (the subject’s attribute data) that exist in the source Dictionary for the production level DATABASE subject with user-name PAYDB to be extracted, and all Description and User Data text lines. Text line numbers are unchanged and match those in the source Dictionary. The subject name qualifiers can be defaulted in the usual way.

The default is to export all existing data for a selected subject. The selection of subject data included is controlled by the **DETAIL** option. The above command assumed a default of **DETAIL=ALL**.

A command stream containing an **ADD** command with various keyword specifications and batch form input would be generated as punch output and a processing report as print output.

**Exporting to an Existing Subject in a Receiving Dictionary**

Suppose that the above DATABASE subject already exists in the receiving Dictionary. The generated **ADD** command would be rejected—the subject would be found when the command was executed. You solve this by using the **MODE** option on the **EXPORT** command. Your command would be coded:

```
EXPORT DATABASE (P,P,PAYDB,0) MODE=UPDATE ;
```

The **MODE** operand would cause an **ADD** command with no subject attributes to be generated, followed by a **CHANGE__IN** command with all the requested attribute data. Required keywords would also be generated. If text data existed for the DATABASE subject, it would first be deleted by a generated **DELETE__DATA** command, then added using batch forms.

In this way, the values in the exported command stream would take effect in the receiving Dictionary whether or not the subjects or relationships already exist.
Selecting a Subset of a Subject's Existing Data

When exporting one or more subjects, you will probably want to include all of the available data associated with that subject. There may be situations where not all of the text data is required to be exported—your installation could be using a set of User Data for administrative purposes. Another situation might be that your installation was making use of a particular range of Description lines for documentation. Other Description lines, beyond the chosen range, would not need to be exported.

The EXPORT command allows you some control over the data exported with each subject. The DETAIL option allows up to nine entries in combination. Using the DETAIL operand, you can:

- Include or suppress all attribute data for exported subjects.
- Include or suppress all aliases for exported subjects.
- Specify that only Description data, PL/I text data, or a specified set of User Data be exported.
- Specify only text lines that occur in a range be exported (or a single line).
- Specify combinations of the above.

When specifying combinations, there are some limitations. Usually, you would require that the attribute data and any aliases to be exported, specified with the ATTR and ALIAS operand values, so that you have seven other text data choices.

You cannot specify a subset of the attribute data. Required keyword ids will be generated regardless of the detail operand.

Selecting Text Data for Export

The DETAIL option for the EXPORT command allows you to specify which type of text data to select. Also, you can specify a range of line numbers or suppress exporting of any text data belonging to Description data, PL/I data, or a set of User Data.

The DETAIL options used for Description data and User Data are DESC and UDn (n=1 to 5), respectively. These can be specified for any subject selected by the EXPORT command.

Your installation may be making use of PL/I data segments to support language data structures. Line selection in the range from 1 to 255, or the complete set of PL/I data, can be specified for SEGMENT and ELEMENT subjects. The DETAIL option for this type of data is PLI.

To select a set of text data, you name the set as a value for the DETAIL option. For example, to select all the Description data and the first set of User Data, you could code:

```
EXPORT DATABASE (P,P,PAYDB,0) MODE=UPDATE +
DETAIL=((ATTR),(ALIAS),(DESC),(UD1)) ;
```
If you do code any of the values for text data for the DETAIL operand, only that
text data is exported. In addition, you have to explicitly request attribute data and
any aliases with the values, ATTR and ALIAS, respectively.

Selecting Ranges of Text Lines: You do not have to transfer the whole set of text
data. A subset of the text data lines can be specified. For example, to export only
the first ten description lines together with the attribute data and any aliases, you
could code:

```
EXPORT DATABASE (P,P,PAYDB,0) MODE=UPDATE +
DETAIL=((ATTR),(ALIAS),(DESC,1,10)) ;
```

The pair of line numbers specifies a range, the first value is the starting line and the
second value the ending line. (The ending line number must be greater than or
equal to the starting line number.) It is not an error if text lines that match the
starting and ending line numbers do not exist. If you specify just one line number
value (or a pair of equal values), a single line is extracted.

You can specify more than one range selection, as necessary. For example,

```
((DESC,1,10),(UD3,100,120),(UD5,10,50)).
```

Note that, if you specify two DESC ranges, the second DESC operand will override
the first DESC operand specified. The same is true if two or more ranges are
specified for the same set of User Data.

The way that a range of text lines appears in the output command stream is in a
batch form for that particular subject and type of text data. The batch form input
contains only those lines that were retrieved from the source Dictionary, and line
numbers are not altered.

If the export mode is update, a DELETE__DATA command is generated to delete
any existing text lines prior to the batch form, which will add the text lines.

If existing text lines were present in the receiving Dictionary, they would remain in
place if they were not in the range being exported. Only the line numbers in the
batch form input would be added.

Exporting Subjects in a Structure

A structure of Dictionary subjects can be exported through the use of the AND
DEPENDENTS operand of the EXPORT command. Beginning with a specified
Dictionary subject (the subject name specified in the command, the first/next
subject specified in a stack, or the first/next subject in the specified category), a
subject's structure consisting of the specified subject and its dependent is exported.
The structure to be used for the EXPORT is described by the structure-type
definition specified by the STRUCTURE option. Instances of relationship types
that are not part of the structure-type definition will be ignored. The
structure-type definition will also be used to determine the direction of
relationships encountered during EXPORT processing.

The different types of relationships that EXPORT may encounter during AND
DEPENDENTS processing are described below. EXPORT will first generate an
ADD command for the subject specified on the command (or for the first/next
subject in the stack or specified category), followed by additional commands and
batch forms images for subject and text data, respectively, as requested by the user. Then the relationships emanating from this subject will be handled as follows:

- **Downward Relationship with FOLLOWDN=YES**

  An ADD command will be generated for the target subject (the dependent). Subject and text data for this subject will be exported according to the DETAIL option. This data will be followed by an ADD__RELATIONSHIP command relating the “from” subject to the “target.” Relationship data, if any, will be automatically exported.

  If the SELSTAT option has been specified on the EXPORT command and the target subject status is not contained in the SELSTAT list, only an ADD__RELATIONSHIP command relating the “from” subject to the “target” subject will be generated. In this case, any relationship data will also be automatically exported.

- **Downward Relationship with FOLLOWDN=NO**

  This will be treated as if it were a relationship with no operational direction (see below).

- **Upward Relationship with FOLLOWUP=YES (or NO)**

  No ADD or ADD__RELATIONSHIP commands will be generated. The relationship will not be followed.

- **No Operational Direction**

  An ADD command will not be generated for the subject in the target category. However, an ADD__RELATIONSHIP command relating the “from” subject to the “target” subject will be generated. Relationship data, if any, will be automatically exported. Note that the execution of this command will result in the creation of a dummy “target” subject if it does not already exist in the receiving Dictionary.

- **Extrastructural Relationship.**

  These relationships will be ignored.

- **Whenever an ADD command is produced for a target subject, the relationships emanating from that subject will also be processed, as described above.**

An example of the structure type of EXPORT is:

```
EXPORT DBS (P,P,DS1,0) AND DEPENDENTS STRUC=STRUC100 + DETAIL=(ATTR,ALIAS,DES,UD1,UD5) DEST=T;
```

This export will cause ADD and ADD__RELATIONSHIP commands and batch forms images to be generated for the structure beginning with DBS (P,P,DS1,0) according to structure-type STRUC100. Assuming that structure-type STRUC100 describes the current Release 4 hierarchy, an ADD command will be generated for the data base subjects and its attributes. ADD__RELATIONSHIP commands will
be generated for its aliases. Batch forms images will be generated for the Description, User Data 1, and User Data 5 associated with the data base subject.

Then an ADD command will be generated for the first segment and its attributes related to the data base subject. ADD__RELATIONSHIP commands will be generated for the aliases of the segment and batch forms images for the Description, User Data 1, and User Data 5 associated with the segment. Then an ADD__RELATIONSHIP command will be generated for the relationship and relationship data between the data base and the segment.

Next, an ADD command will be generated for the first element and its attributes related to the first segment subject. Again, ADD__RELATIONSHIP commands will be generated for the aliases of the element and batch forms images for the Description, User Data 1, and User Data 5 associated with the element. Then, an ADD__RELATIONSHIP command will be generated for the relationship and relationship data between the segment and the element.

Assuming that there are no ELEMENT/CONTAINS/ELEMENT relationships, the next element related to the segment will be processed in the same manner as the first element. After all elements related to the first segment have been exhausted, the next segment related to the data base will be processed in the same manner as the first segment. When all segments related to the data base have been exhausted, the EXPORT is complete. The EXPORT command stream will be directed to the terminal.

Exporting Collections of Subjects Using Stacks

A convenient way of using the EXPORT command when you have several single subjects to export is to use a Dictionary stack. You build a stack containing entries for all the subjects you want to export as one collection. The stack could contain subjects in several categories and the subject names could have different subject name qualifiers.

On completion of your stack preparation you would use the following form of the command:

```
EXPORT *STACK ;
```

The above command would require the prior use of the SETSTACK command to declare the location of the stack. You might need appropriate keyword values for the MODE and DETAIL options. The command execution would result in the generation of a command form input stream for all members of the stack. A combined EXPORT processing report would be produced. The report combines the actions taken for all the subjects in the stack; separate reports are not issued for the individual subjects.

Any options that you specify on the EXPORT command will apply to all subjects in the stack. If you require, for instance, different text data to be selected for members of the collection of subjects, you would have to build separate stacks and use several EXPORT commands, each with an appropriate DETAIL option.

In the same way that you can use a single subject as a starting point for extracting many subjects in a structure, you can use each entry in a Dictionary stack. By specifying an appropriate structure-type that enables you to start with any of the
subjects in the stack as the topmost subject in a structure and follow relationships, whole collections of data can be extracted for each member of the stack. You are limited to a single structure-type specified with the EXPORT command operand, STRUCTURE.

The above uses of a Dictionary stack require flexibility in building the list of subjects. You could generate the entries using the stack-building options provided by the SCAN and STRUCTURE__REPORT commands.

For further information on the use of Dictionary stacks, refer to Chapter 7 of OS/VS DB/DC Data Dictionary Applications Guide.

Exporting Multiple Subjects

You can use the RUNSTACK command to export multiple subjects. However, this method may not be the most efficient command to use (for large amounts of data) in terms of storage space and performance. Let's take, for example, a structure with two segments related to three data elements, as shown below. Suppose the two subjects, SEGMENTA and SEGMENTB, are included in the stack and are the only two subjects in the SEGMENT category.

```
  SEGMENTA
      
    DTE1
    
  SEGMENTB
      
    DTE2
    
    DTE3
```

You could export these subjects in two different ways. If you issue the following command:

```
RUNSTACK EXPORT $CAT $SUBJ AND DEPENDENTS ;
```

it will be the same as issuing a separate EXPORT command for each segment and its dependents. Consequently, the data element, DTE2, will be exported twice; once for SEGMENTA and once for SEGMENTB.

If you issue the EXPORT command, using *CAT SEGMENT AND DEPENDENTS, all of the segments and their dependents will be exported. However, the data element, DTE2, will be exported only once. When exporting SEGMENTB, the EXPORT command will be able to detect that the data element, DTE2, has already been exported with SEGMENTA.

Using Command Lists with the EXPORT Command

You can use the EXPORT command within a command list. You could build up a set of EXPORT commands that together form the desired collection of data to be exported as a single command stream, with each command selecting the appropriate subjects and specifying appropriate command options. To execute the set of EXPORT commands, you use the RUNCLIST command.
An advantage to this technique is that the RUNCLIST command allows an option for controlling whether processing continues if an error occurs while processing any of the commands in the command list. The STOPON option can be specified so that any message of severity 'E' or 'S', resulting from an individual command execution, terminates further processing of the list. (The severity 'W' for warning messages is also available.) By using STOPON=E you can ensure that further processing of EXPORT commands following the error situation is suppressed. The absence of a diagnostic would help you evaluate whether the complete set of EXPORT commands performed the desired extraction.

The disadvantage of this technique is the possibility of generating duplicate ADD, ADD_RELATIONSHIP, and INSTALL commands for the same subject.

For further information on the use of command lists, refer to Chapter 7 of OS/VS DB/DC Data Dictionary Applications Guide.

Exporting to an Empty Receiving Dictionary

When you are exporting data to an empty receiving Dictionary, you use MODE=NEW. For example, to export all of the production subjects in the installation-defined category REPORTS, you could specify:

```
EXPORT *CAT REPORTS +
MODE=NEW SELSTAT=P PROCRPT=NAMES ;
```

The above command assumes that the installation-defined category, REPORTS, is already installed in the receiving Dictionary. Only subjects in the REPORTS category in status “P” are exported.

It is more likely that you will need to transfer not just the single subjects but all of their relationships and dependents too. You do this by specifying a structure-type with the STRUCTURE keyword and the AND DEPENDENTS option.

For example, to make a set of segments and their related fields available, you could use the default structure-type for the command:

```
EXPORT *CAT SEGMENT AND DEPENDENTS +
MODE=NEW SELSTAT=P NEWSTAT=T PROCRPT=NAMES ;
```

All segments in status “P” are exported along with their dependent elements and the relationships between them. All exported subjects are assigned a status of “T” in the EXPORT command stream.

Exporting Data to Maintain a Receiving Dictionary

When you export data to maintain a receiving Dictionary, you use MODE=UPDATE.
When you are maintaining subject definitions and relationships between those subjects in a receiving Dictionary, there are several considerations:

- Do the subjects already exist and can all associated data be replaced?
- What relationship requirements have to be met?
- What is the impact on other existing data when a set of exported data is incorporated into the receiving Dictionary?

You can use a combination of several options provided by the EXPORT command to assist you in controlling what effect the generated command stream will have on the receiving Dictionary.

The MODE option determines the type of commands generated for each exported subject. If the subject does not already exist, you use the default, MODE=NEW. An ADD command with keywords is generated so that the subject and all of its attribute data are added to the receiving Dictionary. Batch forms are used for text data transfer; these use a mode of O (Old) because they always follow the ADD command. However, if the subject exists or may exist, specify MODE=UPDATE. An ADD command is generated with no attribute data to ensure that the subject will exist for the CHANGE__IN commands that will follow. The generated CHANGE__IN commands will contain the subject's attributes. The keyword TO is used so that the keyword data specified on the command replaces the existing data, whatever the current value. The CHANGE__IN command will also add nonexistent attribute data to the receiving dictionary, if the attribute does not require an ID keyword or line number. New data for attributes that require IDs or line numbers will be added only with MODE=NEW.

The batch form input is preceded by a DELETE__DATA command to remove existing text.

The MODE=UPDATE operand causes an ADD__RELATIONSHIP command to be generated with no relationship data for each relationship. The ADD__RELATIONSHIP command would be followed by a CHANGE__RELATIONSHIP__DATA command that would change all relationship attributes in the receiving Dictionary to the new values.

In this way, the values in the exported command stream would take effect in the receiving Dictionary whether or not the subjects or relationships already exist.

The NEWSTAT option is useful when you want to control the status code of all subjects in the generated output stream. The single value specified is used as a status qualifier in all subjects extracted by the EXPORT command containing this operand. Relationship commands are generated with both subjects in the command having this same status code.

The SELSTAT option enables you to select only those subjects that have a status code that matches one in the list specified for SELSTAT. For example, SELSTAT=P might be used to extract only those subjects in production status and ignore any in either a test or installed status.

As with exporting to an empty Dictionary, you can use the *STACK and *CAT options to specify many subjects as candidates for exportation. If any subject in a
stack or in a category does not meet the SELSTAT criteria, no further processing is performed. When processing with a structure-type, relationships are not followed from a subject that fails to satisfy the SELSTAT criteria.

As a check on the appropriateness of the commands that were generated by an EXPORT command, you can use the PREVIEW command in the receiving Dictionary. The processing report helps you find any potential conflicts: PREVIEW result of INDICT or INSTREAM with an ADD command, or NEW with a CHANGE__IN command.

As an example of an EXPORT to update existing information, examine the following command:

```
EXPORT *CAT REPORTS MODE=UPDATE +
       SELSTAT=P NEWSTAT=G +
       DETAIL=(ATTR,ALIAS,(DESC,1,10), (UD1,1,12)) ;
```

All subjects in category REPORTS with status code P are extracted. Any attribute data, any aliases, the first 10 Description lines, and the first 12 lines in the first set of User Data (used as EXPLAIN text) are to be extracted. In the receiving Dictionary, all these subjects should (but do not have to) exist in status G and be replaced. The definition for the REPORTS category had been previously installed. Note that, to ensure the existence of the subject, an ADD command will precede each CHANGE__IN command.

**Exporting Extensibility Definitions**

If your requirements are to transfer data belonging to an installation-defined category, you have an additional consideration. That data cannot be added to the receiving Dictionary unless the customized entity *definition* is in installed status.

With the EXPORT command, you can specify that the CATEGORY or RELTYPE definition itself is exported. All necessary operands are generated with appropriate command keywords. An installed definition is exported as a P status subject (in the absence of a NEWSTAT operand) with accompanying batch forms for Description or User Data text. Also, using the AND DEPENDENTS option, the relationships to ATTRTYPE subjects can be included. Further, the required INSTALL command can be automatically generated (using the INSTALL=YES option). For 'I' status subjects, the INSTALL command is generated with the UPDATE=YES operand. UPDATE is set to NO if the subject is exported in a test status.

If the DETAIL option is specified as ALL or defaulted to ALL for a CATEGORY or RELTYPE category export, an ADD command is generated for the specified category and its attributes. ADD__RELATIONSHIP commands are generated for each one of its aliases. Batch forms images are generated for all text data associated with the category. If the AND DEPENDENTS option is specified, the same actions are performed and, in addition, an ADD command is generated for each ATTRTYPE subject (and its attributes) related to the CATEGORY or RELTYPE subject. ADD__RELATIONSHIP commands are generated for its aliases, and batch forms images are generated for the text data associated with the ATTRTYPE. Then, an ADD__RELATIONSHIP command is generated to relate the CATEGORY or RELTYPE subject to the ATTRTYPE subject. Whenever a RELTYPE subject is exported, you should first export the referenced
CATEGORY subjects. The RELTYPE may not be installed on the receiving Dictionary unless the categories involved in the RELTYPE are already present and installed.

ATTRTYPE definitions may also be exported although they may only be installed as part of a CATEGORY or RELTYPE installation. For an ATTRTYPE category EXPORT, an ADD command is generated for the specified category and its attributes. ADD_RELATIONSHIP commands are generated for each of its aliases. Batch forms images are generated for all text data associated with the category.

Part of your checking before a transfer of a customized entity developed by your installation would be to see if the definition already existed in the receiving Dictionary. You could use a subject-specific report to discover if the CATEGORY or RELTYPE name was known to the receiving Dictionary. Your preliminary checking would also examine the status of attribute-types used by that definition.

Should the definition be already installed in the receiving Dictionary, the INSTALL command will fail, since it does not use the REPLACE option.

The following example shows what the EXPORT command generates when an installation-defined category is exported. Figure 115 on page 328 shows the existing definition's GUIDE report.

```
**** EXPORT GENERATED 12/02/82 AT 15:37:48
* EXPORT CATEGORY (P, ,CLIST,000) AND DEPENDENTS STRUCTURE=ECI+
* STRDEP=NO +
* MODE=NEW DETAIL=((ALL)) PROCRPT=(NAME) +
* DEST=P INSTALL=YES ;
*
* EXPORTING CATEGORY (P, ,CLIST,000)
A CATEGORY (P, ,CLIST,000) +
  SBJCODE=00101 NAMETYPE=N MINNAME=01 MAXNAME=31 ;
DBDFDSC OCATEGORYP CLIST 000
001 SUBJECTS IN THIS CATEGORY ARE USED TO STORE COMMAND LISTS
DBDFEND
INSTALL CATEGORY (P, ,CLIST,000) UPDATE=YES ;
```

If any attribute-types were used by the definition and the AND DEPENDENTS option is used, following the ADD command for the CLIST subject and its batch form input would be one or more ATTRTYPE subject exports. These would be followed by ADD_RELATIONSHIP commands. In this way, the INSTALL command would have all required subjects and relationships established before it is executed.

The same process might be used if you wanted to migrate a definition that had been modified to a receiving Dictionary. The EXPORT command would include MODE=UPDATE INSTALL=YES as operands. You could edit the output command stream to include the REPLACE=YES option but make UPDATE=NO so that you could use the trial INSTALL to detect any definition changes or data migration problems.

If you want to export an extensibility category and its instances, you should export the definition first and install it in the receiving Dictionary. Then the instances should be exported. If you specify INSTALL=YES on the EXPORT command, EXPORT generates the data stream in the proper sequence.
The steps involved for transferring an installation-defined category and its data are:

1. Export the installation-defined category specifying the INSTALL=YES operand.

2. Preview the generated command stream on the receiving Dictionary and resolve any conflicts.

3. Execute the command stream in batch to cause the definition to be installed.

4. Export the data in the installation-defined category using EXPORT *CAT category, for example.

5. Preview the command stream on the receiving Dictionary and resolve any conflicts.

6. Execute the command stream in batch to cause the data to be added to the newly installed category.

Exporting a Structure-Type Definition

A structure-type consists of a subject in the STRTYPE category, plus relationships to subjects in the RELTYPE category. That is, a structure-type definition is, in itself, a structure. Therefore, to export a structure-type definition you need to reference a structure-type to control the processing.

When you are exporting a structure-type definition, you can use the default structure-type or a modified structure-type. Note that when you use the default structure-type supplied by IBM, you will be exporting only the structure-type definition and its relationships (not the relationship-types). Therefore, in this case, the STRTYPE can be successfully installed on the receiving Dictionary, only if the RELTYPE subjects to which it is related are present and installed.

Using the Default Structure-Type

You can specify the name of a structure-type as a subject to be exported. It is possible that you would have the relationship-types used in the structure-type definition already in operation (that is, installed) in the receiving Dictionary. If so, you would not need to export these definitions.

For the STRTYPE subject export, an ADD command is generated for the specified STRTYPE subject. ADD_RELATIONSHIP commands are generated for each of its aliases. Batch forms images are generated for all text data associated with the subject. ADD_RELATIONSHIP commands are generated for the relationships to RELTYPE subjects.

The generated command stream contains ADD_RELATIONSHIP commands for relationship-types with a special difference: An installed relationship-type subject name uses status code of 'S' rather than 'P' as for other installed entities. You could code the EXPORT command:

```
EXPORT STRTYPE (*,ECI,0) AND DEPENDENTS +
     MODE=NEW INSTALL=YES ;
```
The above command uses the default structure-type. Under control of the DEFAULT structure-type supplied by IBM, relationships would not be followed to RELTYPE subjects but all the relationships between the STRTYPE subject at status 'P' and RELTYPE subjects at status '**' would be exported. The generated INSTALL command would be successful.

If the structure-type were not installed, you would have to perform some additional work in the receiving Dictionary to have the structure-type operational. The EXPORT command, in this case, does not use '**' in the status code of the related RELTYPE subjects (or in any ATTRTYPE subjects that are extracted by the given structure-type control). Don't use the INSTALL option. Export the structure-type and any relationship-type definitions that are not present in the receiving Dictionary. Then, install the RELTYPEs as necessary, followed by the installation of the STRTYPE subject. This is the same as exporting a STRTYPE to an empty Dictionary; the INSTALL commands will be generated in the proper sequence.

Again, if the STRTYPE subject already exists in the receiving Dictionary, you can omit the INSTALL option on the EXPORT command. This would cause the STRTYPE subject at status P to be exported, but an INSTALL command would not be generated. You could then make appropriate changes in the receiving Dictionary—you could delete the existing installed structure-type definition and use the exported definition with a subsequent INSTALL command. This procedure is necessary for structure-type definitions, because they cannot use the REPLACE=YES option and they have to be deleted and installed again.

Using a Modified Structure Type

If you need to export the structure-type and all of its related relationship-types, you must use a modified structure-type. You will need to modify the IBM-supplied default structure-type (or define and install a special one) to follow the relationships between the structure-type and its relationship-types.

For the STRTYPE subject export, an ADD command is generated for the specified STRTYPE subject. ADD_RELATIONSHIP commands are generated for each of its aliases. Batch forms images are generated for all text data associated with the subject. ADD_RELATIONSHIP commands are generated for the relationships to RELTYPE subjects. An ADD command is generated for each RELTYPE. ADD_RELATIONSHIP commands are generated for each of its aliases. Batch forms images are generated for all text data associated with the category. ADD_RELATIONSHIP commands are generated for relationships between the STRTYPE and its RELTYPE subjects.

You could code the EXPORT command as:

```
EXPORT STRTYPE (*,,ECI,0) AND DEPENDENTS +
   MODE=NEW STRUCTURE=USERSTR INSTALL=YES ;
```

The above command uses the modified structure-type previously described. Under control of the structure-type USERSTR, the structure-type (*,,ECI,0) would be exported along with its relationships to RELTYPE subjects. These relationships, in turn, would be followed and the RELTYPE subjects exported. INSTALL commands for RELTYPE and STRTYPE subjects would be generated in the correct order.
Command Stream Output Produced by the EXPORT Command

Only the following commands, batch form, and comment records are produced:

- A first comment record, containing the date and a time stamp corresponding to when the EXPORT command was executed, followed by comment records that show the EXPORT command with defaults and specified operands

- ADD commands and any accompanying batch forms images, preceded by comment records

- CHANGE__IN commands and any accompanying batch forms images, preceded by an ADD command with no attribute data

- DELETE__DATA commands to remove text data prior to batch forms images when MODE=UPDATE

- ADD__RELATIONSHIP commands for aliases or for establishing relationships

- CHANGE__RELATIONSHIP__DATA commands for updating relationship data preceded by ADD__RELATIONSHIP commands with no related data

- INSTALL commands for installation-defined entities, preceded by comment records

See Appendix C, “Format Specifications for EXPORT Command Output Streams” on page 467.

Using output routing, you can direct the output from one or more EXPORT commands to an OS/VS data set. This is a convenient method for later editing actions, for use with Dictionary PREVIEW support, and for eventual batch input to the receiving Dictionary. Refer to Chapter 6 in OS/VS DB/DC Data Dictionary Applications Guide for details of output routing support.

Figure 122 shows a generated command stream when a stack of subjects is exported and a customized structure-type was used to control the extent of the data extraction.

You can see the resulting processing report in Figure 123 on page 377.
**** EXPORT GENERATED 01/22/84 AT 15:49:50  
* EXPORT *STACK AND DEPENDENTS STRUCTURE=CUSTOM STRDEF=YES MODE=NEW +  
  SELSTAT=(P) DETAIL=((ALL)) PROCRPT=(ALL) DEST=P INSTALL=YES ;  
*  
* EXPORTING ELEMENT (P,A,EMPADDR,000)  
A ELEMENT (P,A,EMPADDR,000) +  
  BYTES=00080 ;  
DBDFDSC OELEMENT PAEMPADDR 000  
 001 FIELD - EMPADDR CONTAINS EMPLOYEE ADDRESS  
003 SUBFIELD - POBOX CONTAINS EMPLOYEE P.O. BOX NUMBER  
005 SUBFIELD - STREET CONTAINS EMPLOYEE STREET ADDRESS  
007 SUBFIELD - CITY CONTAINS EMPLOYEE CITY OR TOWN  
009 SUBFIELD - ZIP CONTAINS EMPLOYEE ZIP CODE  
DBDFEND  
*  
* EXPORTING ELEMENT (P,A,POBOX,000)  
A ELEMENT (P,A,POBOX,000) +  
  BYTES=00010 TYPE=C ;  
AR ELEMENT (P,A,POBOX,000) HAS_ALIAS +  
  ELEMENT (P,B,BOX_NUM,000) ;  
DBDFDSC OELEMENT PAPobox 000  
 001 FIELD CONTAINS P.O.BOX: UP TO 10 CHARACTERS, RIGHT ADJUST  
DBDFEND  
*  
AR ELEMENT (P,A,EMPADDR,000) CONTAINS +  
  ELEMENT (P,A,EMPADDR,000) +  
  START=00001 ;  
*  
* EXPORTING ELEMENT (P,A,STREET,000)  
A ELEMENT (P,A,STREET,000) +  
  BYTES=00030 TYPE=C ;  
AR ELEMENT (P,A,STREET,000) HAS_ALIAS +  
  ELEMENT (P,B,STREET_ADDR,000) ;  
DBDFDSC OELEMENT PASTstreet 000  
 001 FIELD CONTAINS STREET ADDRESS: UP TO 30 CHARACTERS, RIGHT ADJUST  
DBDFEND  
*  
AR ELEMENT (P,A,EMPADDR,000) CONTAINS +  
  ELEMENT (P,A,EMPADDR,000) +  
  START=00011 ;  
*  
* EXPORTING ELEMENT (P,A,CITY,000)  
A ELEMENT (P,A,CITY,000) +  
  BYTES=00030 TYPE=C ;  
AR ELEMENT (P,A,CITY,000) HAS_ALIAS +  
  ELEMENT (P,B,CITY_ADDR,000) ;  

Figure 122 (Part 1 of 2). Example of Generated Output Commands from an EXPORT Command
The EXPORT Processing Report

A report is produced for each individual execution of an EXPORT command. The detail contained in the report is controlled by values for the PROCRPT operand on the command. The size of a report will depend on the amount of extracted data, according to your selection criteria.

Figure 122 (Part 2 of 2).  Example of Generated Output Commands from an EXPORT Command
Your best choice for processing a report option is to use the PROCRPT=All operand, and obtain all the information provided. If you want to simplify the report somewhat, you can use:

**NAMES**  This option lists only the names and aliases of exported subjects. An entry in the report with "EXPORTED" in the first column has the subject name listed to the right, and the "ACTED UPON" column has an entry "PRIMARY" or "ALIAS," respectively.

**RELS**  This option lists all relationships exported. An entry in the report with "EXPORTED" in the first column and "REL'SHIP" in the "ACTED UPON" column indicates this action. The second subject name that will occur in the **ADD_RELATIONSHIP** command is listed to the right. The first subject name occurs earlier in the report in the entry identified by "RESUMED" in the "ACTION" column.

**INST**  This option is of interest only if your **EXPORT** command uses the **INSTALL** option and lists the generation of an **INSTALL** command as an entry "INSTALCMD" in the "ACTION" column.

If you want to suppress the body of the report, use option, PROCRPT=NONE. The command options and summary sections will always be printed.

Figure 123 on page 377 illustrates a processing report. You can see that the report is composed of the following sections:

- **Command options**
  
  This section confirms which operands were specified for the **EXPORT** command and which defaults are in effect.

- **Relationship-types**
  
  This section lists the structure-type content and appears if you specify STRDEF=YES for the command.

- **Exported subjects**
  
  This is the main body of the report. Each line records the action taken by **EXPORT** command processing in response to the selection criteria. The information contained in this section is controlled by the PROCRPT operand.

- **Reason code legend**
  
  This section explains the meaning of error conditions encountered during the command processing; these conditions are flagged with a single-letter code in the column headed "RC" in the main report section. Only codes that have occurred are listed.

- **Summary of processing**
  
  This final section gives statistics about the content of the generated command stream and any subjects that were not exported for lack of authorization.
Figure 123 (Part 1 of 2). Example of an EXPORT Processing Report
### EXPORT PROCESSING REPORT

11/15/85  09:30:55

<table>
<thead>
<tr>
<th>ACTION</th>
<th>RC</th>
<th>ACTED-UPON</th>
<th>CATEGORY</th>
<th>SUBJECT</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOLLOWING</td>
<td></td>
<td>REL'SHIP'TO</td>
<td>ELEMENT</td>
<td>PA</td>
<td>ZIP 0</td>
</tr>
<tr>
<td>EXPORTED</td>
<td></td>
<td>PRIMARY-NAME</td>
<td>ELEMENT</td>
<td>PA</td>
<td>ZIP 0</td>
</tr>
<tr>
<td>EXPORTED</td>
<td></td>
<td>ALIAS-NAME</td>
<td>ELEMENT</td>
<td>PB</td>
<td>ZIP_NUM 0</td>
</tr>
<tr>
<td>EXPORTED</td>
<td></td>
<td>REL'SHIP'TO</td>
<td>ELEMENT</td>
<td>PA</td>
<td>ZIP 0</td>
</tr>
<tr>
<td>FINISHED</td>
<td></td>
<td></td>
<td>ELEMENT</td>
<td>PA</td>
<td>ZIP 0</td>
</tr>
<tr>
<td>RESUMED</td>
<td></td>
<td></td>
<td>ELEMENT</td>
<td>PA</td>
<td>EMPADDR1 0</td>
</tr>
<tr>
<td>FINISHED</td>
<td></td>
<td></td>
<td>ELEMENT</td>
<td>PA</td>
<td>EMPADDR1 0</td>
</tr>
<tr>
<td>EXPORTED</td>
<td></td>
<td>PRIMARY-NAME</td>
<td>STRTYPE</td>
<td>P</td>
<td>STRTYPE1 0</td>
</tr>
<tr>
<td>FOLLOWING</td>
<td></td>
<td>REL'SHIP'TO</td>
<td>RELTYPE</td>
<td>P</td>
<td>RELTYPE1 0</td>
</tr>
<tr>
<td>EXPORTED</td>
<td></td>
<td>PRIMARY-NAME</td>
<td>RELTYPE</td>
<td>P</td>
<td>RELTYPE1 0</td>
</tr>
<tr>
<td>EXPORTED</td>
<td></td>
<td>ALIAS-NAME</td>
<td>RELTYPE</td>
<td>P</td>
<td>SOURCE/ISSUES/REPORTS 0</td>
</tr>
<tr>
<td>EXPORTED</td>
<td></td>
<td>REL'SHIP'TO</td>
<td>RELTYPE</td>
<td>P</td>
<td>RELTYPE1 0</td>
</tr>
<tr>
<td>SKIPPED</td>
<td>C</td>
<td>PRIMARY-NAME</td>
<td>ATTRTYPE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INSTALCMD</td>
<td></td>
<td>PRIMARY-NAME</td>
<td>RELTYPE</td>
<td>P</td>
<td>RELTYPE1 0</td>
</tr>
<tr>
<td>FINISHED</td>
<td></td>
<td>RELTYPE</td>
<td>P</td>
<td>RELTYPE1 0</td>
<td></td>
</tr>
<tr>
<td>RESUMED</td>
<td></td>
<td>STRTYPE</td>
<td>P</td>
<td>STRTYPE1 0</td>
<td></td>
</tr>
<tr>
<td>SKIPPED</td>
<td>S</td>
<td>PRIMARY-NAME</td>
<td>RELTYPE</td>
<td>*</td>
<td>RELTYPE2 0</td>
</tr>
<tr>
<td>FOLLOWING</td>
<td></td>
<td>REL'SHIP'TO</td>
<td>RELTYPE</td>
<td>P</td>
<td>RELTYPE2 0</td>
</tr>
<tr>
<td>EXPORTED</td>
<td></td>
<td>PRIMARY-NAME</td>
<td>RELTYPE</td>
<td>P</td>
<td>RELTYPE2 0</td>
</tr>
<tr>
<td>EXPORTED</td>
<td></td>
<td>REL'SHIP'TO</td>
<td>RELTYPE</td>
<td>P</td>
<td>RELTYPE2 0</td>
</tr>
<tr>
<td>SKIPPED</td>
<td>C</td>
<td>PRIMARY-NAME</td>
<td>ATTRTYPE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INSTALCMD</td>
<td></td>
<td>PRIMARY-NAME</td>
<td>RELTYPE</td>
<td>P</td>
<td>RELTYPE2 0</td>
</tr>
<tr>
<td>FINISHED</td>
<td></td>
<td>RELTYPE</td>
<td>P</td>
<td>RELTYPE2 0</td>
<td></td>
</tr>
<tr>
<td>RESUMED</td>
<td></td>
<td>STRTYPE</td>
<td>P</td>
<td>STRTYPE1 0</td>
<td></td>
</tr>
<tr>
<td>INSTALCMD</td>
<td></td>
<td>PRIMARY-NAME</td>
<td>STRTYPE</td>
<td>P</td>
<td>STRTYPE1 0</td>
</tr>
<tr>
<td>FINISHED</td>
<td></td>
<td>STRTYPE</td>
<td>P</td>
<td>STRTYPE1 0</td>
<td></td>
</tr>
<tr>
<td>SKIPPED</td>
<td>P</td>
<td>PRIMARY-NAME</td>
<td>RELTYPE</td>
<td>P</td>
<td>RELTYPE1 0</td>
</tr>
</tbody>
</table>

**REASON CODE LEGEND:**

- C - INSUFFICIENT CATEGORY AUTHORIZATION
- P - SUBJECT PREVIOUSLY EXPORTED
- S - INSUFFICIENT STATUS AUTHORIZATION

**SUMMARY:**

- NUMBER OF SUBJECTS EXPORTED: 8
- NUMBER OF ALIASES EXPORTED: 5
- NUMBER OF RELATIONSHIPS EXPORTED: 6
- NUMBER OF SUBJECTS TO BE INSTALLED: 3

---

**Figure 123 (Part 2 of 2). Example of an EXPORT Processing Report**
If you intend to work with the EXPORT command in the online environment, you should assess both the number of subjects that might be selected and the amount of associated text data.

Validating Data for a Receiving Dictionary

The output stream of commands and batch forms generated by EXPORT commands becomes an input stream to the receiving Dictionary. When you have verified that the contents are complete and as desired, you can proceed to incorporate this data into the receiving Dictionary’s data base.

Although you would routinely examine the outcome of processing the input stream to see if there was any error condition, it is recommended that you validate the transferred data in other ways prior to updating the receiving Dictionary.

The Dictionary offers two kinds of assistance:

- **EXPORT** command processing reports
  
  These reports confirm that data selection was successful and document the contents of the output command stream. They can help you verify that all required subjects and attribute data are included, in addition to any required relationships between those subjects.

- **PREVIEW** command processing reports
  
  This command supplies you with a report that helps you discover the impact of the incoming data without updating the receiving Dictionary’s content.

Remember that it is better to do a thorough check of the exported data before altering the receiving Dictionary. This will cause less confusion and inconvenience to the users of the receiving Dictionary. Your installation can use the reports obtained with the EXPORT and PREVIEW commands within the control procedures for data transfer.

In your examination of the generated stream, you may perform an offline edit to alter or delete some of the records. The editing process is another part of validation. This process may be repeated after you have looked at the results from PREVIEW processing.

The PREVIEW command is available only in a batch environment. It uses as input the command and batch form input stream generated as a sequential file or as a set of records by one or more EXPORT commands. It also works on user-generated command streams, which contain the set of Dictionary commands supported by EXPORT. This input stream is defined as an input data set, ddbname DDPREV, for the batch Dictionary.

You submit a PREVIEW command as normal command input (DDINPUT), to the receiving Dictionary. When the PREVIEW command is encountered, the Dictionary accesses the DDPREV data set and examines each command and batch form in turn. A data base comparison is made; added subjects or relationships should not exist in the receiving Dictionary, and changed data should have subjects
or relationships already existing in the receiving Dictionary. No data base updates are made. The commands are checked for syntactical errors and normal command diagnostics included within the PREVIEW processing report. But any other command processing is suppressed.

The PREVIEW processing report consolidates all the findings and records the results of the examination of the whole contents of the sequential input file, which may consist of the output from one or more EXPORT commands. You can use only a single PREVIEW command and a single sequential input data set for each run of the batch Dictionary.

The remainder of this section explains the options available with the PREVIEW command and the interpretation of its reports.
Format and Options for the PREVIEW Command

The general format of this command is:

<table>
<thead>
<tr>
<th>Required</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREVIEW</td>
<td></td>
</tr>
<tr>
<td>Required</td>
<td>Optional</td>
</tr>
<tr>
<td>STATCHK=status</td>
<td>PROCRPT=ALL</td>
</tr>
<tr>
<td>(status1,...,status10)</td>
<td>(detail1,...,detail6)</td>
</tr>
<tr>
<td>DEST=L,P</td>
<td></td>
</tr>
</tbody>
</table>

The options for the command are explained below.

**PREVIEW**

The command verb. There is no abbreviated form.

**STATCHK=status | (status1,...,status10)**

An optional operand that specifies that the listed status code(s) be substituted for the existing status code of each subject name in the input command stream. This check is made for all commands. The input command stream is not updated by this option. Up to 10 valid status codes may be specified. If this operand is omitted, the status codes specified in the command stream will be used to check for name duplication.

**PROCRPT=ALL | detail | (detail1,...,detail6)**

An optional operand that specifies the type of information to be listed in the PREVIEW processing report. Valid values are:

- **ALL** Requests that the information requested by all of the following operands be reported.
- **NEW** Requests that a list of subjects that will be created by execution of the input stream be reported. These subjects do not currently exist in the receiving Dictionary.
- **INDICT** Requests that a list be reported for those subjects in the input stream that match subjects that currently exist in the receiving Dictionary.
- **INSTRM** Requests that a list be reported for those subjects in the input stream that match subjects that were previously encountered in the command stream.
- **INVALID** Requests that a list be reported of all commands that will be invalid. This includes commands with syntax or authorization errors in addition to commands that reference subjects that do not exist in the Dictionary and will not be created by the
command stream, but must exist in order for the command to work.

**UNSUPP** Requests that unsupported commands encountered by the PREVIEW command be reported. PREVIEW supports only the subset of commands generated by EXPORT.

**INPUT** Requests that a copy of the input command stream be included in the report.

Up to 6 values may be specified. If omitted, ALL is the default value. If ALL appears anywhere in the list, the entire PREVIEW report will be printed.

*Note:* If a command is both invalid and unsupported, it will appear in the INVALID section of the processing report.

**DEST=** L | P
An optional operand that specifies the destination for the PREVIEW processing report. Valid values are:

- **L** Line Printer
- **P** Punch

If omitted, the default value is **L**.

**Specifying Input for the PREVIEW Command**

When you generate an output stream, using the EXPORT command, you are able to control what form the output data takes:

- An OS/VS sequential data set
- A member of a partitioned data set
- A set of output records, possibly in card form

Whatever form this output takes, this data becomes eligible for input to the receiving Dictionary. It must be specified with a JCL DD statement with ddbname DDPREV. Your installation needs to make provision for batch operation of the receiving Dictionary with this additional JCL. For example,

```
//DDPREV DD DSN=DBDCDICT.EXP0067,DISP=(OLD,KEEP)
```

Although the generated stream produced by the EXPORT command has a uniform format, there are no requirements beyond the usual command and batch form syntax for the PREVIEW input stream. If you did some offline editing of the exported stream, you do not have to conform to the EXPORT output conventions. These conventions are documented in Appendix C, "Format Specifications for EXPORT Command Output Streams" on page 467.

The EXPORT output stream contains comment records generated with the update commands. You can alter or delete these records, or add comments of your own.
The only syntactical requirement is an asterisk (*) in the first position of the record.

PREVIEW support is to help you in testing the effect of subject additions or modifications, without actually making those changes, so that other Dictionary processing cannot be accommodated in the DDPRIV input stream.

Submitting the PREVIEW Command

You use the usual DDINPUT data set to submit the PREVIEW command to the receiving Dictionary. Any problems occurring during operation of the PREVIEW command are reported in the usual way as diagnostics to DDPRINT. You can submit other Dictionary commands at the same time, such as reports that identify the current data content.

If user access control is active in the receiving Dictionary, you will need to precede the PREVIEW command in the DDINPUT data set with a SIGN__ON command. The access profile selected needs to include appropriate command, category, and status code authorization.

Although subjects generated in the EXPORT command output use fully qualified subject names, as a result of editing actions, some command entries might need to use subject name default qualifiers. The access profile can be used to specify these defaults, which may be different in the receiving Dictionary from those in the source Dictionary. Also, SET commands can be included before the PREVIEW command to make appropriate adjustments, for example, to logging and language code defaults.

Using Additional Status Code Checking

A primary concern when initializing or adding to a receiving Dictionary is the integrity of the data. The PREVIEW command will help you find out what the effect of each subject definition is when added to the receiving Dictionary, or when the definitions replace existing ones. But, you need to consider that other data already existing in the receiving Dictionary's data base may be impacted by the changes to the data base content.

For example, suppose you export a production segment and its related fields and want to make this a test structure in the receiving Dictionary. The EXPORT command allows you to set the status code as 'T' (test) in the generated output stream. What if some of these subjects had been created already, using status code 'D'? One way to discover this would be to try both status code 'T' and 'D' in the commands in the input stream that adds the subjects. If the subject name with status code 'D' was found to be a duplicate name, you would be warned of this condition.

Although some of this type of validation can be discovered only by full processing of the command, such as "status rules" checking, being able to specify several status codes and trying each version of the subject in the update commands before the receiving Dictionary is updated can give you an extra level of protection.

To use the additional status code checking, you specify for the STATCHK operand a list of up to 10 status codes. For example, STATCHK=(T,D,P). As each individual command that appears in the input stream is read, the first entry in the
status code list is selected. This status code is substituted in place of the original one in the command. If this is an ADD__RELATIONSHIP command, the status code is substituted in both of the subject names. Then the command data base implications are examined—without an update taking place.

This procedure is repeated for the next status code in the STATCHK list, and so forth, until the list is exhausted. The status codes are not rearranged but used in the order specified on the PREVIEW command. The processing continues with the next command in the input stream. The contents of the records in the DDPREV data set are not altered in any way.

If you do not specify a STATCHK at all, the status code values occurring within subject names in the commands and batch forms are used directly.

Selecting PREVIEW Report Options

The PROCRPT operand on the PREVIEW command controls the type of validation information that will appear in the PREVIEW processing report. You can obtain all available information by omitting the operand, the default is PROCRPT=ALL.

There are several types of validation reporting. Report entries occur in the order in which the input commands are processed. The report entries, with the corresponding PROCRPT option given in parentheses, are:

- Subjects that will be added to the receiving Dictionary or constructed with STATCHK processing as a result of processing the input command stream (NEW)

- Subjects in the input stream, or constructed with STATCHK processing, that match existing subjects in the receiving Dictionary (INDICT)

- Subjects that occur more than once in the input stream, or are referenced after the initial occurrence (INSTRM)

- Commands that have an invalid syntax if the input stream is entered, as specified, into the receiving Dictionary (INVALID)

Two other options are available for documenting the content of the input stream, rather than the effect in the receiving Dictionary:

- The INPUT option requests that a copy of the input stream be included in the processing report.

  Input records are numbered sequentially and these record sequence numbers are used as identifiers in subsequent sections of the processing report. For this reason, you will usually specify this option.

- The UNSUPP option reports on unsupported commands that are present in the input stream. Each such command is listed with its input record number; no further checking or processing takes place.
Interpreting the PREVIEW Processing Report

Figure 125 on page 393 illustrates the types of processing report sections produced by the PREVIEW command:

- Command options
- First input record
- Input command stream
- Previewed commands
- Invalid commands
- Unsupported commands
- Summary of processing

Provided that there are no problems with the PREVIEW command itself, such as improper syntax or status code authorization, you always receive the first two sections ("Command Options" and "First Input Record") and a "Summary Section."

**Command Options and First Input Record**

The first section confirms what options are in effect. An asterisk by the command option indicates that a default was used. The first input record of the PREVIEW input is usually a comment containing the date and time stamp produced by a corresponding EXPORT command. It is extracted at the top of the report to assist you in identifying which data you are validating.

**Input Command Stream**

The "Input Command Stream" section contains the printed images of the records included in the entire DDPREV input data set. To the left of each of these images is an assigned record number, starting at 1. This section of the report is produced if you specify PROCRPT=INPUT or PROCRPT=ALL (the default).

**PREVIEW Results**

The main body of the report starts on a new page. The first entry identifies the category and subject name that appears in the first command in the input stream. Two adjacent columns, headed "PREVIEW RESULT" and "PREVIEW OF," are used to explain the impact on the receiving Dictionary. To the right, the type of command and the record number is given as an aid to identifying which part of the input stream was being processed (under the headings "COMMAND" and "RECORD").

Other entries are included under the initial entry that summarize all other findings that apply to that initial subject. These could be the addition of attribute data, batch form input, participation in a relationship command, or possibly an INSTALL command reference. The entries are in ascending order of record number.
Following the initial subject, a similar summary is included for each subject occurring in the input stream or generated by STATCHK processing.

**Invalid Commands**—Command checking in PREVIEW processing,

The PREVIEW command has restrictions on which commands are selected for processing from the DDPREV input stream. Only the following are processed:

- ADD commands and any accompanying batch form input
- CHANGE__IN commands and any accompanying batch form input
- DELETE__DATA commands to remove text data prior to batch form input
- ADD__RELATIONSHIP commands for aliases or for establishing relationships
- CHANGE__RELATIONSHIP__DATA commands for updating relationship data
- INSTALL commands for installation-defined entities

All other commands are reported as “unsupported commands.”

As each of the above command and batch form inputs is used in PREVIEW processing the outcome could be an error condition. The “Invalid Commands” section lists these commands and includes the Dictionary message that highlights the type of error.

For example, an editing process might have introduced a syntactical error, such as a missing name qualifier, misspelled keyword, or invalid attribute data. If user access control is active, improper authorization for the category access and status code use could cause the command to be listed as invalid.

**Reporting Unsupported Commands**

Because the PREVIEW processing has to keep track of the commands occurring in the input stream, there is a way that you can find out the location of any unsupported commands in the DDPREV input stream. You select the UNSUPP option for the PROCRPT operand.

When you select this option, the PREVIEW processing report includes an “Unsupported Commands” section. Each line of this section of the report identifies a record number and the first record of the command. You can refer to the “Input Command Stream” section to see the location of the record in context. If the unsupported commands span multiple records, only the first record appears in this section.

The unsupported commands section is also produced if you use the default (ALL) for the PROCRPT operand.
Summary Section

At the end of the processing report are some statistics that summarize the PREVIEW processing. You can use the totals as an initial check of complete processing and to detect whether there are errors recorded within the body of the report.

PREVIEW Processing Indicators

The column headed **PREVIEW RESULT** in the processing report contains indicators of the effects of the particular command action on the receiving Dictionary.

**NEW** The subject’s name, primary or alias, or a relationship would be added to the receiving Dictionary. These entries are obtained when **PROCRT=NEW** is specified.

**INDICT** The subject or relationship already exists in the receiving Dictionary when this command is tried. This indicator is not used if the subject in the command was previously encountered in the PREVIEW input stream. These entries are obtained when **PROCRT=INDICT** is specified.

**(blank)** The subject in the command was identical to one previously encountered in the PREVIEW input. These entries are obtained when **PROCRT=INSTRM** is specified.

The columns headed “PREVIEW OF”, “COMMAND”, and “RECORD” help identify the command in which the subject occurred and its location in the command stream.

The **PREVIEW OF** column contains the following entries:

**SUBJECT** To indicate that the command to the right uses the subject.

**PRIMARY** To indicate that the command establishes the subject name as a primary name.

**ALIAS** To indicate that the command establishes the subject name as an alias name.

**REL'SHIP** To indicate that the command establishes a relationship between the subject and another subject.

For example, the five report lines:

<table>
<thead>
<tr>
<th>ELEMENT</th>
<th>TA</th>
<th>EMPADDR</th>
<th>000</th>
<th>INDICT</th>
<th>SUBJECT</th>
<th>CI</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>DD</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>BFORM</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
<td>19</td>
</tr>
</tbody>
</table>
The first line indicates that the subject (T,A,EMPADDR,0) exists as a subject and is updated by the CHANGE)__IN command beginning on record 5 of the input stream. The second line indicates that the current subject is used in a DELETE__DATA command beginning on record 7. The third line shows the current subject is used in batch form input that begins on record 9.

For actions involving relationships, the fourth line shows that the subject is referenced in an ADD__RELATIONSHIP command beginning on record 19 in the input stream. The fifth line indicates that the current subject has a new relationship created by the action of the same ADD__RELATIONSHIP command, beginning on record 19.

**Example of a PREVIEW Processing Report**

Figure 124 on page 390 illustrates the different sections of the PREVIEW command processing report. This is not a complete report, but, rather, parts of each section of the report.

The command used to get this report is:

```
/PREVIEW DEST=L STATCHK=(A,B,C) PROCRPT=ALL
```

A command stream actually created by the EXPORT command was previewed.

**COMMAND OPTIONS and FIRST INPUT RECORD Sections:** The COMMAND OPTIONS section lists any default options and options specified on the PREVIEW command. An * before an option indicates that a default value was taken. This report did not take any default values.

**FIRST INPUT RECORD Section:** Contains the first record of the command stream to be previewed if it is a comment. The command stream produced by EXPORT contains a date and time stamp comment as the first record to assist you in identifying which data you are validating.

**INPUT COMMAND STREAM Section:** The input command stream is listed with the record number and the contents of the record. These records are printed images of the records included in the DDPREV input data set. Numbers are assigned to each record, beginning with record 1.

**PREVIEWED COMMANDS Section:** The PREVIEWED COMMANDS section is organized according to arrival sequence, with all references to a particular category and subject name grouped together.

The RECORD column indicates the record number of the command being previewed.

The COMMAND column indicates what command is being processed.

Under the PREVIEW OF column, an entry of SUBJECT indicates that the command is being previewed. An entry of REL'SHIP indicates that the relationship between the subject and another subject is being previewed.

Under the PREVIEW RESULT column, an entry of NEW means that the subject or relationship would be added to the receiving Dictionary. An entry of INDICT
means that the subject or relationship already exists in the receiving Dictionary. A blank entry means that the subject or relationship has appeared previously in the command stream.

**INVALID COMMANDS Section:** This section would identify which commands were invalid for processing. This report has none.

**UNSUPPORTED COMMANDS Section:** This section would identify which commands were unsupported. This report has none.

**SUMMARY Section:** This section identifies the number of subjects, alias names, or relationships that are new or existing in the receiving Dictionary. The subjects, alias names, or relationships are categorized by the status codes specified by the user in the PREVIEW command (STATCHK). This section also identifies the number of invalid commands, number of unsupported commands encountered, the number of INSTALL commands encountered, the number of commands in the command stream, and the number of records in the command stream.
PREVIEW PROCESSING REPORT 02/17/84 18:15:55
PAGE: 0001

COMMAND OPTIONS:

STATCHK: A, B, C
PROCRPT: NEW, INPUT, INSTRM, INDICT, INVALID, UNSUPP
DEST: L

NOTE: '!' MEANS DEFAULT VALUE TAKEN

FIRST INPUT RECORD:
**** EXPORT OUTPUT GENERATED 02/11/82 AT 11:06

INPUT COMMAND STREAM

<table>
<thead>
<tr>
<th>RECORD</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*** EXPORT OUTPUT GENERATED 02/11/84 AT 11:06</td>
</tr>
<tr>
<td>2</td>
<td>* EXPORT APL (T,,APL411,0) AND DEPENDENTS DETAIL=ALL</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
</tr>
<tr>
<td>4</td>
<td>* APL (T,,APL411,0) EXPORTED - M4: MIXED, DISCONNECTED CATEGORIES</td>
</tr>
<tr>
<td>5</td>
<td>A APL (T,,APL411,0)</td>
</tr>
<tr>
<td>6</td>
<td>A SET (T,,SET406,0)</td>
</tr>
<tr>
<td>7</td>
<td>AR APL (T,,APL411,0)</td>
</tr>
<tr>
<td>8</td>
<td>SET (T,,SET406,0) INCLUDES +</td>
</tr>
<tr>
<td>9</td>
<td>A DTE (T,A,DTE406,0)</td>
</tr>
<tr>
<td>10</td>
<td>AR SET (T,,SET406,0)</td>
</tr>
<tr>
<td>11</td>
<td>DTE (T,A,DTE406,0) CONTAINS +</td>
</tr>
<tr>
<td>12</td>
<td>RELDATE=070182</td>
</tr>
<tr>
<td>13</td>
<td>* EXPORT PSB (T,P,PSB410,0) AND DEPENDENTS DETAIL=ALL</td>
</tr>
<tr>
<td>14</td>
<td>*</td>
</tr>
<tr>
<td>15</td>
<td>* PSB (T,P,PSB410,0) EXPORTED</td>
</tr>
<tr>
<td>16</td>
<td>A PSB (T,P,PSB410,0)</td>
</tr>
<tr>
<td>17</td>
<td>A PCB (T,P,PCB422,1)</td>
</tr>
<tr>
<td>18</td>
<td>AR PSB (T,P,PSB410,0) WITH +</td>
</tr>
<tr>
<td>19</td>
<td>PCB (T,P,PCB422,1)</td>
</tr>
<tr>
<td>20</td>
<td>A DBS (T,P,DBS412A,0)</td>
</tr>
<tr>
<td>21</td>
<td>AR PCB (T,P,PCB422,1) WITH +</td>
</tr>
<tr>
<td>22</td>
<td>DBS (T,P,DBS412A,0)</td>
</tr>
<tr>
<td>23</td>
<td>A SEG (T,A,SEG412AA,0)</td>
</tr>
<tr>
<td>24</td>
<td>AR DBS (T,P,DBS412A,0) WITH +</td>
</tr>
<tr>
<td>25</td>
<td>SEG (T,A,SEG412AA,0)</td>
</tr>
</tbody>
</table>

Figure 124 (Part 1 of 3). Example of PREVIEW Processing Report
<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>SUBJECT</th>
<th>PREVIEW RESULT</th>
<th>PREVIEW OF</th>
<th>COMMAND</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>APL</td>
<td>A</td>
<td>000</td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td>APL</td>
<td>B</td>
<td>000</td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td>SET</td>
<td>A</td>
<td>000</td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td>SET</td>
<td>B</td>
<td>000</td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td>SET</td>
<td>C</td>
<td>000</td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>AA</td>
<td>000</td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>BA</td>
<td>000</td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>CA</td>
<td>000</td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td>PSB</td>
<td>AP</td>
<td>000</td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td>PSB</td>
<td>BP</td>
<td>000</td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td>PSB</td>
<td>CP</td>
<td>000</td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INDICT</td>
<td>SUBJECT</td>
<td>AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NEW</td>
<td>REL'SHIP</td>
<td>AR</td>
</tr>
</tbody>
</table>

Figure 124 (Part 2 of 3). Example of PREVIEW Processing Report
INVALID COMMANDS
NONE

UNSUPPORTED COMMANDS
NONE

SUMMARY:

<table>
<thead>
<tr>
<th></th>
<th>&quot;A&quot;</th>
<th>&quot;B&quot;</th>
<th>&quot;C&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBJETS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- NEW</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>- IN DICTIONARY</td>
<td>53</td>
<td>53</td>
<td>53</td>
</tr>
<tr>
<td>ALIASES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- NEW</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>- IN DICTIONARY</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RELATIONSHIPS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- NEW</td>
<td>41</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>- IN DICTIONARY</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>INVALID COMMANDS</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

NUMBER OF UNSUPPORTED COMMANDS ENCOUNTERED : 0
NUMBER OF INSTALL COMMANDS ENCOUNTERED : 0
NUMBER OF COMMANDS IN COMMAND STREAM : 116
NUMBER OF RECORDS IN COMMAND STREAM : 226

**** END OF PREVIEW PROCESSING REPORT ****
DBD6691 I PREVIEW PROCESSING COMPLETED

Figure 124 (Part 3 of 3). Example of PREVIEW Processing Report

Example of a PREVIEW Processing Report with User-Generated Stream

Figure 125 on page 393 illustrates another PREVIEW command processing report with a user-generated stream. We assume that the command stream has been modified by the user. The PROCRT operand for this example specified a list of all six options.

In this example, an existing ELEMENT subject, (T,A,EMPADDR), has its length altered. A new subfield, (T,A,POBOX,0), and an alias, (T,B,BOX__NUM,0), are added. (T,A,POBOX,0) is related to (T,A,EMPADDR,0).

To discover whether either of the above subjects exists with status code 'D', the STATCHK operand has specified both 'T' and 'D'.

At the end of the command stream input, an exported definition for an installation-defined category (CLIST) is added, followed by a trial INSTALL command. The effect of the STATCHK operand causes the CLIST definition to be tried at status code 'T' and 'D'. The following INSTALL command is
referenced for status code 'T' and 'D'. Note that the INSTALL commands are not executed and neither is installation validation performed nor processing reports produced.

To illustrate the way standard messages appear in the "Invalid Commands" section, the batch form header for the CLIST subject was set to 'N' (New). Usually, the EXPORT command uses a mode of 'O' (Old), which would not be in error because the batch form always follows an ADD (or CI) command.

PREVIEW PROCESSING REPORT 12/29/83 10:19:04
PAGE: 0001

COMMAND OPTIONS:

STATCHK: T, D
PROCRET: NEW, INPUT, INSTRM, INDICT, INVALID, UNSUPP
DEST : L

NOTE: '*' MEANS DEFAULT VALUE TAKEN

FIRST INPUT RECORD:
***** USER GENERATED STREAM 09/29/82 AT 12:59:59

INPUT COMMAND STREAM

RECORD CONTENTS

----- -----------------------------------------------
1   **** USER GENERATED STREAM 09/29/83 AT 12:59:59
2   *
3   *
4   *
5   CI   DTE (T,A,EMPADDR,0) +
6   LEN=80
7   DD   DTE (T,A,EMPADDR,0) +
8   DATA=(DTEDSCR,15)
9   DBDFDSC ODTE TAEMPADDR 000
10  015 FIELD - POBOX CONTAINS EMPLOYEE P.O. BOX NUMBER
11  DBDFEND
12  ADD   DTE (T,A,POBOX,0) +
13  TYPE=C LEN=10
14  DBDFDSC ODTE TAPobox 000
15  001 FIELD CONTAINS P.O. BOX: UP TO 10 CHARACTERS, RIGHT ADJUST
16  DBDFEND
17  AR   DTE (T,A,POBOX,0) HAS_ALIAS +
18  DTE (T,B,BOX_NUM,0)
19  AR   DTE (T,A,EMPADDR,0) CONTAINS +
20  DTE (T,A,POBOX,0) START=71
21  SETSTAT D
22  *
23  * CATEGORY (D,,CLIST,0) EXPORTED
24  ADD   CATEGORY (D,,CLIST,0) +
25  SBJCODE=99 DATE=111182
26  DBDFDSC NCAT D CLIST 000
27  001 CATEGORY USED FOR LANGUAGE PREPROCESSOR COMMAND LISTS
28  DBDFEND
29  *
30  * INSTALL CATEGORY (D,,CLIST,0) EXPORTED
31  INSTALL CATEGORY (D,,CLIST,0) UPDATE=NO

Figure 125 (Part 1 of 3). Example of PREVIEW Processing Report with User-Generated Stream
## PREVIEW PROCESSING REPORT

**PREVIEWED COMMANDS**

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>SUBJECT</th>
<th>COMMAND</th>
<th>RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELEMENT</td>
<td>TA EMPADDR</td>
<td>000 INDICT</td>
<td>SUBJECT CI 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUBJECT DD 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUBJECT BFORM 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUBJECT AR 19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REL'SHIP AR 19</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>TA POBOX</td>
<td>000 NEW</td>
<td>SUBJECT ADD 12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUBJECT BFORM 14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PRIMARY AR 17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REL'SHIP AR 17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUBJECT AR 19</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>DA POBOX</td>
<td>000 NEW</td>
<td>SUBJECT ADD 12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUBJECT BFORM 14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PRIMARY AR 17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REL'SHIP AR 17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUBJECT AR 19</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>TB BOX_NUM</td>
<td>000 NEW</td>
<td>ALIAS AR 17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REL'SHIP AR 17</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>DB BOX_NUM</td>
<td>000 NEW</td>
<td>ALIAS AR 17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REL'SHIP AR 17</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>DA EMPADDR</td>
<td>000 NEW</td>
<td>SUBJECT AR 19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REL'SHIP AR 19</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>T CLIST</td>
<td>000 NEW</td>
<td>SUBJECT ADD 24</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>D CLIST</td>
<td>000 NEW</td>
<td>SUBJECT INSTALL 31</td>
</tr>
</tbody>
</table>

## INVALID COMMANDS

```
5 CI   DTE (T,A,EMPADDR,0) +  
  DBD6641 E SUBJECT DOES NOT EXIST IN STATUS D, COMMAND WILL FAIL
7 DD   DTE (T,A,EMPADDR,0) +  
  DBD6641 E SUBJECT DOES NOT EXIST IN STATUS D, COMMAND WILL FAIL
9 DBFDFSC ODTE TAEMPADDR 000  
  DBD6653 E BATCH FORM 'MODE' IS INCONSISTENT WITH THE EXISTENCE  
  OF THE SUBJECT IN STATUS D
26 DBFDFSC NCAT D CLIST 000  
  DBD6653 E BATCH FORM 'MODE' IS INCONSISTENT WITH THE EXISTENCE  
  OF THE SUBJECT IN STATUS T
```

**Figure 125 (Part 2 of 3). Example of PREVIEW Processing Report with User-Generated Stream**
PREVIEW PROCESSING REPORT 12/29/83 10:19:04
PAGE: 0003

RECORD COMMAND (FIRST LINE ONLY)
-------- ---------------------------------------------
26 DBFDSC NCAT D CLIST 000
DBD6653 E BATCH FORM 'MODE' IS INCONSISTENT WITH THE EXISTENCE
OF THE SUBJECT IN STATUS D

UNSUPPORTED COMMANDS
RECORD COMMAND
-------- ---------------------------------------------
21 SETSTAT D

PREVIEW PROCESSING REPORT 12/29/83 10:19:04
PAGE: 0004

SUMMARY:

- IN STATUS "T" "D"
   ------ ------
SUBJECTS
- NEW 3 4
- IN DICTIONARY 4 0
ALIASES
- NEW 1 1
- IN DICTIONARY 0 0
RELATIONSHIPS
- NEW 1 1
- IN DICTIONARY 0 0
INVALID COMMANDS
1 4

NUMBER OF UNSUPPORTED COMMANDS ENCOUNTERED : 1
NUMBER OF INSTALL COMMANDS ENCOUNTERED : 1
NUMBER OF COMMANDS IN COMMAND STREAM : 11
NUMBER OF RECORDS IN COMMAND STREAM : 31

**** END OF PREVIEW PROCESSING REPORT ****
DBD6691 I PREVIEW PROCESSING COMPLETED

Figure 125 (Part 3 of 3). Example of PREVIEW Processing Report with User-Generated Stream
Completing the Data Transfer

The output stream produced by one or more EXPORT commands becomes the input to the receiving Dictionary. Assuming that you have validated the content of this stream of commands and batch forms, you need to enter this data into a batch version of the receiving Dictionary to complete the data transfer.

This section offers some considerations for this last step in the task of exporting data to a receiving Dictionary.

Editing the Receiving Dictionary's Input Data

In your examination of the EXPORT command-generated output, you may perform an offline edit to alter or delete some of the records. This editing process may be repeated after you have looked at the results from PREVIEW processing.

At the time you generate an output stream, using the EXPORT command, you are able to control what form the output data takes:

- An OS/VS sequential data set
- A member of a partitioned data set
- A set of output records, possibly in card form

To edit the above data, you can use any method available to your installation that accepts the data in one of these formats. Keep in mind that the PREVIEW processing does not alter the input stream in any way, so that any conflicts you discovered will warrant a change in the input stream.

The following is a summary of various editing actions that might need to be done:

- Altering of status codes or other subject qualifiers in the command input
- Changing text data or renumbering of text lines
- Specifying other command options
- Deleting command input that would cause a conflict in data base status or unnecessary duplication

Including EXPORT Command Output in a Batch Stream

Because the output generated by EXPORT processing becomes normal input to a batch Dictionary, you are not limited as to the type of commands that can be present in the input stream. For example, you can add appropriate report commands that will show both new and existing data in the receiving Dictionary in context.
If user access control is active in the receiving Dictionary, you need to precede the input stream with a appropriate SIGN_ON command. One way of doing this is to concatenate a data set containing the appropriate SIGN_ON command to the data set containing the input stream (DDINPUT). The access profile needs appropriate authorization for the commands and any required category and access authority.

You are not limited to the results of a single EXPORT command as input to the receiving Dictionary. Usually, you will combine the effects of several tailored EXPORT actions into one input stream. This has the advantage of localizing the changes to a particular time and reducing the chance that the receiving Dictionary’s content undergo a change during your data transfer activities.

**Verifying the Update of the Receiving Dictionary**

A necessary step following the execution of the batch input stream is to examine the output for diagnostics. You need to discover if any of the expected actions did not occur.

Because the PREVIEW validation did not update the data base, there may be conflicts when some of the subjects or relationships are added. An example would be if your editing actions with the input stream introduced syntactical errors.

**Notifying the Receiving Dictionary’s Users**

You will probably want to notify all those users affected by the transfer of new or updated definitions into the receiving Dictionary.

You can use Dictionary reports to disseminate this information. Although detailed information about the data transfer is contained in both the EXPORT and PREVIEW command processing reports, these reports are considered of most value to those planning and implementing the data transfer rather than the general user. Glossary or STRUCTURE__REPORT command output would be more appropriate for circulation. If your data transfer included installation-defined entities, you need to distribute GUIDE reports or inform users of online information obtainable from EXPLAIN forms.
Appendix A. Establishing Documentation Standards

The first step in establishing documentation standards is to gather data. Organizations just starting to use automated data processing should have no problems with this step; they merely collect the data for the Dictionary as part of the usual process of designing and implementing their application systems.

Organizations that have a large number of automated application systems may extract subject definitions from existing documentation and programs. If they already have good documentation and naming conventions, they can minimize effort by implementing the following techniques:

- Extract the existing descriptions for element groups, physical segments, data sets, and data bases and their relationships from existing common data descriptions (for example, COBOL copy libraries, PL/I %INCLUDE libraries, and DL/I DBD and PSB libraries).

- Complete the definitions from existing documentation sources. This may be necessary because program data definitions usually contain only a portion of the information that is needed in the Dictionary.

For organizations with a large number of automated application systems but without uniform documentation and data-naming standards, the initial data collection becomes a major effort. Here, instead of dealing with a single common data description, you may find the same data element with several different names, the same name used to describe different data elements, and so forth. These inconsistencies make the initial subject definition process and the establishment of relationships between subjects difficult tasks.

Bulk input can be managed by the COBOL__IN, PLI__IN, DBD__IN, and PSB__IN commands and by batch forms. Conflicting entries can be analyzed with the STRUCTURES__OUT, SCAN, and REPORT commands. The conflicts can be resolved by selecting the proper mode of DBD__IN or by assigning aliases with the ADD RELATIONSHIP command. Until the conflicts can be resolved, it may be advisable to allow duplicate user names to exist with multiple occurrence numbers.

If previous documentation is a problem, one approach is to initially include in the Dictionary only new application systems and to add existing application systems as they are scheduled for a major redesign.
The Need for Standards

What is needed is a standard approach to describing data characteristics that will make it possible for a user to find information in the Dictionary when only a general description of the item is known.

For example, a system analyst may need to know whether a data element already exists or is being defined for the first time. If it already exists and has been defined in the Dictionary, the analyst will have to consider present users of the element and choose between using the existing description or allowing a duplicate name (with a different occurrence number). If the duplicate user name is allowed, the analyst may still want to use most of the element description as it was originally defined.

An occasional user may want to know if some data item is part of the organization's data bases, if it really represents what is needed, and if so, under what name is it stored.

The SCAN and REPORT functions of the Dictionary can help answer these questions. These functions are most effective if the data items are described consistently.

In planning for Dictionary installation, it is advisable to set up standards for Dictionary definitions. Planners should decide how subject categories should be used, consider what use should be made of Description and User Data text, and set up minimum requirements for each subject category.

For example, the documentation about a data element could contain information such as:

- Its meaning
- Its standard name for program use
- Who has responsibility for specifications
- Its characteristics (size, type, and so forth)
- How it is obtained (source document, computation, what algorithm)

The documentation for a program could contain information such as:

- Description of the program
- Indication of the supporting resources (for example, data sets, data elements, input transactions, main storage)

In DL/I systems, there are eight names for which conventions should be established:

- DBD name
- DD statement name (ddname)
- Segment name
- PSB/Program name
- Element name
- MOD name for display screens (IMS/VS MFS only)
- Transaction code
- LTERM names

**Naming Conventions**

Standardization helps to simplify the tasks of anyone concerned with programming. By proper use of naming conventions, documentation standards, and the DL/I system libraries, it should be possible for programmers to pick up other programmers' work and understand what they are doing with their programs.

Although 31-byte names are accepted in the Dictionary command language, DL/I names and PSB names should be limited to 8 alphameric bytes. Only 8 bytes are accepted in user names that occur in keyword values, and DL/I names in the Interactive Display Forms facility are limited to 8 bytes.

DBCS users must specify names as uniform strings. They should realize that use of DBCS outside the Dictionary environment is limited to documentation. IMS/VS allows only EBCDIC characters in names, labels, and values in source statements for DBDs, PSBs, and System Definition. Only EBCDIC characters can be used for DDNAMES, dataset names, program names, and member names in JCL or when directly referenced from inside a program. GIS/VS allows only EBCDIC characters in names and values for generation source statements.

The user name can include characters for the system identification, the number of this data item in the system, and variable qualifiers for different subject categories. The important thing is that the names give useful information and that they are used and recognized by all who use your system.

**Documentation Standards**

You should store a free-form text description for every subject. The description should contain at least the common name, meaning, and purpose. For elements that represent codes, a description of each value should be stored. If the code list is extensive, the entry should indicate where the code values are explained (for example, a number of a code standard or data set containing the table).

One approach to standardization is to require that every subject be described by a short identifier constructed from a controlled list of keywords. This description should be the one searched with the SCAN command to examine possible redundancies or inconsistencies in stored elements.
For example, an element that is described as representing "the number of dependents that an employee declares for tax deduction purposes" might have an identifier containing the following keywords:

COUNT, EMPLOYEE, DEPENDENTS, TAX, DEDUCTION

The SCAN function of the Dictionary could provide a report of these keywords as related to stored subject names, so that an analyst looking at a new element and knowing only its general meaning can choose some keyword(s) and use the SCAN command to determine whether the element is already defined.

Another use could be to group data elements with equal or similar sets of keywords (one match, two matches, and so forth) and then to analyze them for redundancy or conflict in meaning.

An occasional user of the organization's Dictionary could apply the same technique to find the description of an element when only the general meaning is known.

In one technique, sometimes called the Designator Technique, a set of descriptive keywords called a designator is assigned to a subject. The keywords in the designator are ordered in a hierarchy from the most general keyword to the most specific keyword. To improve readability, some null words are inserted between keywords. Because the most common null word is the preposition of, this technique is sometimes called the of language. For example,

NAME of FIELD which is KEY of SEGMENT

A few keywords seem to appear in most designators. They are called class words because they provide for a basic classification of data:

NAME (identifier)
CODE
COUNT (quantity)
AMOUNT (currency)
DATE
TEST
FLAG (yes or no)
CONTROL (delimiters, carriage control characters, and so on)
CONSTANT (for example, message)

Consistency and accuracy of the set of keywords are best achieved when designators are assigned centrally (by data administration), instead of letting individual programmers define them.
Maintaining System Standards

This section contains suggested methods for implementing the control functions of the Dictionary in your data base system that are of special interest to the data administrator. The use of the status qualifier and User Data entries is also explained. The following topics are considered:

- Controlling versions of Dictionary definitions
- Controlling DL/I libraries
- Tracking library changes in User Data segments
- Other uses for the User Data segments
- Controlling the use of programming language data structures

Controlling Versions of Dictionary Definitions

The design phase of application development may produce several alternative approaches to a given data base problem. Before putting the new system into production status, central control should be applied in the process of integrating the final designs (integration-level designs) with one another and with the existing production data base. One of the test version status codes should be selected for the purpose of integration; status 0 is one possibility. The completed designs will exist as Dictionary structures in other status groups. The status codes 1 through 9 could be used, or a group of alphabetic codes chosen from A through T, except the production status code P. At the start of integration, the Dictionary administration (or DBA) group should begin supervising the copying and combining of these designs into one final structure in the 0 status. Dictionary subjects with a production (P) status may be included in this integration version, according to the rules of structure formation. This integration level of the application design can then be submitted to a validation procedure, during which output from the REPORT, DBD__OUT, PSB__OUT, and STRUCTURES__OUT commands are examined by all the interested groups, and finally approved. Simultaneously, the DBDs and PSBs produced from this 0-version data base and application design can be used in running DL/I tests of the performance of the new application.

The testing done during this phase should be under the control of the DBA and other interested groups. The testing procedures will vary at each installation, but the DBA should ensure that, whatever they are, the Dictionary output from the 0 integration level is used directly, and that there is no way to bypass the Dictionary at this phase. This will ensure that, when the integration version is finally converted to production (P) status, it reflects the system that was tested at this time.

After testing of a system is complete, when it is approved for use in regular production schedules, the Dictionary should be used to actually convert it into production. The P=Production status in the Dictionary is used for this, and the DBA should be responsible for transferring the completed version-0 Dictionary structure into production status. After it is in this status in the Dictionary, the library control procedures described below should be used to generate the necessary DBD, PSB, and programming language COPY or %INCLUDE library
members directly from the Dictionary. This again is a key point of Dictionary
control over the application; using the Dictionary in this way not only improves the
Dictionary administrator's control over changes to the data bases, but keeps
Dictionary data current.

The remaining status qualifiers might be used to develop and implement changes in
data that do not affect the structure of a data base. A change in the field layout of
a segment might be done in status C, merely by copying the segment from P to C
with the COPY command AFFECTED option. This copies the minimum
Dictionary structure required to implement such a change, while using mostly
existing production subjects (which are not being changed) to show the overall
relationships among the data. After the procedure above has been used to verify
that the change is correct and acceptable, the COPY command with the
DEPENDENTS option, changing the status of the segment from C to P, will
transfer the changes back into the production status. Again, after the change, the
library control procedures described below should be used to build the new DBD,
PSB, and COPY library members.

Controlling DL/I Libraries

The DL/I libraries with which we are concerned here are the DBD, PSB, and ACB
libraries. These control blocks all go into separate libraries. Controlling these
libraries means ensuring that their contents reflect exactly the definitions in the
Dictionary, and keeping track of changes made in both the Dictionary and the
control blocks, so that they can be reconstructed if they are destroyed or
accidentally back leveled.

The first step in controlling these control blocks is to set up a procedure for
updating them that restricts this function to the DBA. The data set password
facility may be used to enforce these regulations. Updating of these control blocks
must be done from the Dictionary directly, without intermediate DBD or PSB
source. There should be only one location for the source form of these control
blocks: the Dictionary. A job stream and associated procedures should be set up
to produce DBD_OUT and PSB_OUT output from the Dictionary, store it in a
temporary file, perform DBDGEN or PSBGEN, perform ACBGEN for the
required blocks, and store tracking information about this updating.

Separate job streams should exist for updating control blocks from production
Dictionary definitions and from test version definitions, and care should be taken to
keep systems that are still to be tested from contaminating or destroying the DBDs
or PSBs that are required for day-to-day production runs. Users should have
separate libraries for these separate versions, corresponding to production level and
integration level systems. Your specific installation procedures will determine
which methods are used to enforce this separation, but it is important that it be
maintained.

Transfer of the system from integration (testing) status to production status should
be done in the Dictionary, as described in the preceding section, and the new
members from the production system libraries should be generated again from the
Dictionary after the transfer is completed. Copying of members from integration
level libraries to production libraries should be prohibited.
Tracking Library Changes in User Data Segments

In a data base system, changes must be managed so that they have minimum impact on users of the production level system. In spite of precautions, something may go wrong:

- An intended change may be unsuccessful.
- An unintended change may somehow occur.
- After a loss resulting from a hardware, software, or operating problem, system resources may be restored to a back level.

If any of these events occur, the invalid status conditions must be quickly identified, and the system restored to its intended status. To expedite handling of such problems, the data base administrator must have complete, up-to-date accounting information that answers such questions as:

- What was the change?
- What was affected by the change?
- Why was it changed?
- Who authorized it?
- Who changed it?
- When was it changed?

Effective control of system libraries requires comprehensive files of current SYSGEN, DBDGEN, PSBGEN, ACBGEN, APAR, and PTF listings. For tracking purposes, however, back-level listings and appropriate documentation of change authorization are of equal importance. Some installations may satisfy these additional requirements with a change journal and a file of change authorizations maintained manually. To ensure that such information is maintained consistently and readily available, it should be included in your data Dictionary.

Two pieces of information are of major importance: the status of the resource and the identity of the person who made the most recent changes in it. This information should be available for each DBD, PSB, and ACB in the production- and integration-level system libraries. It is recommended that the Dictionary User Data entries be used to maintain the tracking information. Several lines of a specified User Data segment in the data base and system subject categories should be assigned for tracking information for DBDs and PSBs, respectively. Each time a change is made in a production- or integration-level DBD or PSB, the data base administrator should update the associated tracking information entries. When the ACBGEN is completed for the PSB or DBD, the associated tracking information should be updated.
Other Uses for the User Data Segments

The Dictionary has available five sets of User Data (each with 999 lines of 80 bytes) for each subject definition, intended for whatever information the user wants to store.

A typical use of the User Data segments might be to store the JCL for a program within a specific User Data segment associated with the subject name for the program. Another entry might be operator instructions for running the program. The User Data segments give a high degree of flexibility in storing installation-dependent information in the Dictionary.

Controlled Use of Language Data Structures

Language data structures are data definitions, in programming language form, for the data segments that can be retrieved from your data base. The control of language data structures is important for the same reasons control of DL/I libraries is important. In addition, the way in which the data structure definitions are managed can provide improved data independence, a primary goal for a data base system.

These data structures should provide programmers with a single consistent interface between the program and the data they use from the data base. All programs should use the same data structure for the same data segment to keep field names consistent throughout programs. Programmers looking at an unfamiliar piece of code should recognize the field names for data base fields from their own programming or at least be able to look them up in the Dictionary. Productivity can be greatly increased through all programming phases by this standardization. The COPY and %INCLUDE features of COBOL and PL/I are the tools that can be used to achieve this standardization, and the Dictionary can be the controlling factor.

The COPYLIB or %INCLUDE libraries that contain the data structures for the data base should be controlled by the DBA in the same manner as the DL/I libraries discussed above, using the Dictionary STRUCTURES__OUT function. (Not all source statement libraries need to be under DBA control; some will contain program segments that do not pertain to the data base system.) Tracking the changes to these structures by means of User Data segments or a manual system is also important.

Developing and maintaining consistent and unique data structures for data base segments is made easier by the use of the Dictionary. Especially while converting from nonstandard data definitions, the Dictionary’s duplicate name and alias features can be very useful for documenting the existing situation and can identify problem areas to address in developing consistent data definitions.
Appendix B. Reference Material for the Program Access Facility

This appendix consists of several sets of reference material useful when writing or debugging a program that accesses the Dictionary data bases (that is, a program that uses the Program Access facility). The sets of reference material are:

- "Appendix B.1. Overall Status of a Program Access CALL Statement" on page 408 describes return codes that can be set, when the Dictionary returns to a user program, to describe the overall status of the request.

- "Appendix B.2. Status of Each Format Table (PADAFT) Entry" on page 412 describes return codes that can be set, when the Dictionary returns to a user program, to describe the status of each item of retrieved data.

- "Appendix B.3. The Control Area (PACA)" on page 414 describes the format and contents of the PACA control block.

- "Appendix B.4. The Format Table (PADAFT)" on page 424 describes the format and contents of the PADAFT control block.

- "Appendix B.5. The COBOL Sample Program" on page 427 describes the COBOL sample program (distributed with the Dictionary) that uses the Program Access facility to retrieve Dictionary data and issue messages to an online user.

- "Appendix B.6. PADAFT Entries for Dictionary Categories" on page 437 describes the contents of the master PADAFT used to retrieve attribute data of subjects in standard categories and relationships.

- "Appendix B.7. Program Access Macro Support" on page 458 describes the macros (distributed with the DB/DC Data Dictionary program product) used to set up an Assembler language user program's PACA and PADAFT control blocks.

- "Appendix B.8. Special Topics for Attribute Retrieval" on page 462 describes items of interest pertinent to special types of attribute retrieval.
Appendix B.1. Overall Status of a Program Access CALL Statement

A return code is set in the binary halfword PARC (or PACA-RC) field of the PACA to indicate the overall status of a retrieval or output request. The return code is also returned in register 15. Assembler language programs could check this register instead of the PARC field.

The programming aids supplied with the Dictionary include initialized definitions of the symbolic return code constants shown in Figure 126. The corresponding COBOL names are specified to the right of the PL/I and Assembler language names. Your program can test for selected values rather than perform a full examination of the return code. In general, severe errors are indicated by return codes of 12 or greater. Your program should record each severe error condition and notify the user with an error message (for example, a line of user program output).

Values that can occur as PACA return codes:

<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0000)</td>
<td>PACOK</td>
<td>PACOK</td>
<td>Retrieval successful. The retrieval action has completed. PADAFT return codes may indicate that individual items were not retrieved. For list-type retrievals, the positioning fields are set to the last retrieved values, ready for a subsequent CALL to retrieve the next available portion of the list.</td>
</tr>
<tr>
<td>4 (0004)</td>
<td>PACEOD</td>
<td>END-OF-DATA</td>
<td>End-of-data condition. This is a successful retrieval of all of the possible data; returned data fields for positioning are set to blank.</td>
</tr>
<tr>
<td>6 (0006)</td>
<td>PACLAST</td>
<td>POSITIONED-ON-LAST</td>
<td>The PADA and the PACA positioning fields have been filled with the last set of data retrieved. (This return code is associated with the specification of PAADOWN.)</td>
</tr>
<tr>
<td>8 (0008)</td>
<td>PACNF</td>
<td>NO-DATA-FOUND</td>
<td>Data not found; no available data satisfies the request.</td>
</tr>
<tr>
<td>12 (000C)</td>
<td>PACPNF</td>
<td>POSIT-NOT-FOUND</td>
<td>The list element identified by the positioning operands cannot be found.</td>
</tr>
<tr>
<td>16 (0010)</td>
<td>PACDATA</td>
<td>NO-DATA-SPEC</td>
<td>No specified data was requested by the CALL. (Your program failed to set a PADAFT operation code to retrieve any PADAFT item.)</td>
</tr>
</tbody>
</table>

Figure 126 (Part 1 of 4). Return Codes for Retrieval and Output Requests
<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 (0014)</td>
<td>PACAUTH</td>
<td>USER-NOT-AUTHORIZED</td>
<td>The request specified retrieval of data from a category or used a status code for which the user did not have access authorization.</td>
</tr>
<tr>
<td>24 (0018)</td>
<td>PACCTRL</td>
<td>BAD-CARRIAGE-CTL</td>
<td>An invalid carriage control character is specified.</td>
</tr>
<tr>
<td>28 (001C)</td>
<td>PACREL</td>
<td>NO-SUCH-RELATION</td>
<td>The requested relationship does not exist between the specified subjects.</td>
</tr>
<tr>
<td>32 (0020)</td>
<td>PACNAME</td>
<td>NO-SUCH-SUBJ</td>
<td>The specified subject does not exist.</td>
</tr>
<tr>
<td>36 (0024)</td>
<td>PACDOWN</td>
<td>BAD-DOWN-COUNT</td>
<td>The down count given as a positioning operand is invalid.</td>
</tr>
<tr>
<td>40 (0028)</td>
<td>PACNREQ</td>
<td>BAD-NUM-OF-ITEMS</td>
<td>The number of data collections specified for the list-type retrieval is invalid.</td>
</tr>
<tr>
<td>44 (002C)</td>
<td>PACINCR</td>
<td>BAD-INCREMENT</td>
<td>The increment (that is, number of bytes) between the start of one data collection and the next is invalid.</td>
</tr>
<tr>
<td>46 (002E)</td>
<td>PACPNAME</td>
<td>NO-SUCH-POSIT-SUBJ</td>
<td>The positioning subject name does not exist.</td>
</tr>
<tr>
<td>48 (0030)</td>
<td>PACPCAT</td>
<td>NO-SUCH-POSIT-CAT</td>
<td>The positioning category name does not exist.</td>
</tr>
<tr>
<td>50 (0032)</td>
<td>PACPLINE</td>
<td>BAD-LINE-NUMBER</td>
<td>The starting line number for a retrieval of text data is invalid or not numeric.</td>
</tr>
<tr>
<td>52 (0034)</td>
<td>PACRKW</td>
<td>BAD-REL-KEYWORD</td>
<td>No relationship-type exists between the specified categories that uses the specified keyword name.</td>
</tr>
<tr>
<td>54 (0036)</td>
<td>POCRNAME</td>
<td>NO-SUCH-REL-SUBJ</td>
<td>The subject specified as being related to the positioning subject does not exist, or the positioning subject for related subjects does not exist.</td>
</tr>
<tr>
<td>56 (0038)</td>
<td>POCRUT</td>
<td>BAD-REL-CAT</td>
<td>The category specified for a related subject does not exist.</td>
</tr>
<tr>
<td>60 (003C)</td>
<td>PACCAT</td>
<td>BAD-PRIN-CAT</td>
<td>The principal category specified does not exist.</td>
</tr>
<tr>
<td>64 (0040)</td>
<td>PACFTU</td>
<td>BAD-PAFT-USAGE</td>
<td>PADAFT usage specified in the PAFTU field of the PACA is invalid.</td>
</tr>
</tbody>
</table>

Figure 126 (Part 2 of 4). Return Codes for Retrieval and Output Requests
<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>68 (0044)</td>
<td>PACFTO</td>
<td>PAFT-AREA-OVERFLOW</td>
<td>PADAFT area overflow—the CALL requested that the master PADAFT be built, but the length of the specified area is less than the length of the indicated master PADAFT. The required size appears in the PAFTLM field.</td>
</tr>
<tr>
<td>72 (0048)</td>
<td>PACFT</td>
<td>BAD-PAFT- SPEC</td>
<td>The PADAFT pointer has not been set in the PACA, or the PADAFT identification, format code, or number of entries is invalid.</td>
</tr>
<tr>
<td>76 (004C)</td>
<td>PACDAOP</td>
<td>BAD-PADA-OPERATION</td>
<td>The operation specified in the PADAOP field of the PACA is invalid.</td>
</tr>
<tr>
<td>80 (0050)</td>
<td>PACDAO</td>
<td>DATA-AREA-OVERFLOW</td>
<td>PADA overflow—one or more fields defined by the PADAFT would, if retrieved, fall partially or completely outside of the area specified for the PADA. The data item is not retrieved. Partial retrieval of requested data might have occurred. If you issued a “list-type” retrieval, this condition may be the result of an attempt to get too many collections, or of using too large an area for each collection.</td>
</tr>
<tr>
<td>84 (0054)</td>
<td>PACDA</td>
<td>BAD-PADA- SPEC</td>
<td>The specification of the PADA pointer or length is invalid.</td>
</tr>
<tr>
<td>88 (0058)</td>
<td>PACACC</td>
<td>BAD-ACCESS-CODE</td>
<td>The specification of the PAACCESS field is invalid.</td>
</tr>
<tr>
<td>92 (005C)</td>
<td>PACTYPE</td>
<td>BAD-REQUEST-TYPE</td>
<td>The type of retrieval or output request is invalid.</td>
</tr>
<tr>
<td>96 (0060)</td>
<td>PACDLI</td>
<td>DLI-ERROR</td>
<td>A DL/I error has been encountered. The Dictionary also issues message DBD0013 to identify the error.</td>
</tr>
<tr>
<td>100 (0064)</td>
<td>PACAINV</td>
<td>BAD-PACA-SPEC</td>
<td>The specification of the PACA has an invalid identification or format code. The Dictionary also issues the message DBD5305.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: Because this could be the result of a bad PACA address, there is no attempt to insert this code into the PARC field.</td>
</tr>
</tbody>
</table>

Figure 126 (Part 3 of 4). Return Codes for Retrieval and Output Requests
<table>
<thead>
<tr>
<th>Return Code (in PARC)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>104 (0068)</td>
<td>PACERR</td>
<td>UNEXPECTED-CONDITION</td>
<td>An unrecognizable error condition was detected by the Dictionary. This condition is probably an error in either the Dictionary data bases or in the Dictionary code.</td>
</tr>
<tr>
<td>108 (006C)</td>
<td>PACSNAME</td>
<td>BAD-STRUC-TYPE</td>
<td>The structure-type definition is not in &quot;installed&quot; status.</td>
</tr>
<tr>
<td>112 (0070)</td>
<td>PACNOSTK</td>
<td>STACK-NOT-DEFINED</td>
<td>The request specified output to a Dictionary stack which had not been properly defined using a SETSTACK command.</td>
</tr>
<tr>
<td>116 (0074)</td>
<td>PACIOSTK</td>
<td>STACK-WRITE-ERROR</td>
<td>The request specified output to a Dictionary stack located in a data set and there was a write error.</td>
</tr>
<tr>
<td>120 (0078)</td>
<td>PACOVSTK</td>
<td>STACK-OVERFLOW</td>
<td>The request to add an entry to a Dictionary stack failed because the stack was full.</td>
</tr>
<tr>
<td>124 (007C)</td>
<td>PACNOSBJ</td>
<td>STACK-SUBJ-MISSING</td>
<td>The request to add an entry to a Dictionary stack failed because the stack location was not found.</td>
</tr>
<tr>
<td>128 (0080)</td>
<td>PACCONFL</td>
<td>STACK-CONFLICT</td>
<td>The request to add an entry to a Dictionary stack failed because the stack processing was already in operation. This could occur if a RUNSTACK command specified EXEC as command verb and one of the programs invoked issued a WSTK call.</td>
</tr>
<tr>
<td>132</td>
<td>PACNORTE</td>
<td>ROUTING-NOT-DEFINED</td>
<td>Indicates an error occurred while trying to route output during a WPRT or WPCH call. This can occur if the destination defined by the current SETPRINT or SETPUNCH command does not exist.</td>
</tr>
</tbody>
</table>

Figure 126 (Part 4 of 4). Return Codes for Retrieval and Output Requests
Appendix B.2. Status of Each Format Table (PADAFT) Entry

Before returning to your program, the Dictionary sets a return code for each PADAFT entry in the binary halfword field named PAFRC (or PAFT-RC) (in addition to setting the PARC value in the PACA). The value of each entry’s return code can be examined by your program. The programming aids define and initialize a symbolic constant for each return code.

The highest value of the return codes returned for the PADAFT entries is put in the PACA’s PART (or HIGHEST-PAFT-RC) field shown in Figure 127. You might want to test this value before analyzing individual PAFRC values.

When a value of 12 or higher is returned in the PAFRC, the corresponding data area (PADA) field has been left unchanged (unless the PACA’s PADAOP field specifies that the PADA field is to be set to blanks).

Values for the PAFRC field:

<table>
<thead>
<tr>
<th>Return Code (PADAFT)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0000)</td>
<td>PAFOK</td>
<td>PAFOK</td>
<td>Data item retrieved successfully.</td>
</tr>
<tr>
<td>4 (0004)</td>
<td>PAFLONG</td>
<td>PADA-FIELD-LONG</td>
<td>Data item retrieved successfully. However, the field length specified in the PADAFT is greater than that specified in the master PADAFT. Nonnumeric fields are padded to the right with blanks; numeric character fields are padded to the left with character zeros.</td>
</tr>
<tr>
<td>8 (0008)</td>
<td>PAFSHORT</td>
<td>PADA-FIELD-SHORT</td>
<td>Data item retrieved successfully. The field length specified in the PADAFT is less than that specified in the master PADAFT. Character fields have blanks truncated on the right; numeric fields have leading zeros truncated on the left.</td>
</tr>
<tr>
<td>12 (000C)</td>
<td>PAFNULL</td>
<td>NO-DATA-FOR-ATTR</td>
<td>Data item is unavailable. The value for the indicated attribute is “null.” If the PADAOP field in the PACA requests that field blanking is to be done, both numeric and character fields in the PADA are set to blank.</td>
</tr>
<tr>
<td>16 (0010)</td>
<td>PAFFOF</td>
<td>PADA-FIELD-OVERFLOW</td>
<td>Significant characters lost for this data item. The length specified for the attribute’s value resulted in the loss of significant characters. As much as would fit in the PADA field was returned.</td>
</tr>
</tbody>
</table>

Figure 127 (Part 1 of 2). Return Codes for PADAFT Entries
<table>
<thead>
<tr>
<th>Return Code (PADAFT)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 (0014)</td>
<td>PAFFUNI</td>
<td>FUNC-CODE-INCORRECT</td>
<td>Data item not retrieved. A value other than “R” or “N” was specified for the corresponding PAFFUNC table entry.</td>
</tr>
<tr>
<td>24 (0018)</td>
<td>PAFDUP</td>
<td>DUPLICATE-ATTR</td>
<td>Data item not retrieved. The specified attribute keyword and instance are defined more than once in the PADAFT. (Data may have been retrieved for the earlier definition.)</td>
</tr>
<tr>
<td>28 (001C)</td>
<td>PAFNF</td>
<td>ATTR-NOT-FOUND</td>
<td>Data item not retrieved. The specified attribute keyword and instance do not match any keyword and instance in the corresponding master PADAFT.</td>
</tr>
<tr>
<td>32 (0020)</td>
<td>PAFDOF</td>
<td>PADA-OVERFLOW</td>
<td>Data item not returned. The offset specified for the PADA field (or offset combined with length) would place it outside the PADA.</td>
</tr>
</tbody>
</table>

Figure 127 (Part 2 of 2). Return Codes for PADAFT Entries
Appendix B.3. The Control Area (PACA)

The symbolic names and values in this section apply to fields and constants in PL/I or Assembler language programs using the control area definition aids provided with the Dictionary. The corresponding names for COBOL are specified to the right of the PL/I and Assembler language names.

Every PACA starts with two identification fields, followed by control fields. When you use one of the programming aids for initialization, the identification fields and many of the other fields have appropriate initial values.

The PACA fields that contain subject names (that is, PANAME, PARNAME, PAPNAME, and PAPRNAME) must be in Dictionary format type I. See “Type of Attributes” in “Appendix B.8. Special Topics for Attribute Retrieval” on page 462 for details about format type I.

The following information is specified in the PACA:

**PACA and Call Identification**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (00)</td>
<td>4</td>
<td>PACAID</td>
<td>IDENTIFIER</td>
<td>Contains the 4-character identification “PACA”.</td>
</tr>
<tr>
<td>4 (04)</td>
<td>1</td>
<td>PACAFM</td>
<td>FORMAT-CODE</td>
<td>Contains the 1-character value “A” or “B”.</td>
</tr>
<tr>
<td>5 (05)</td>
<td>4</td>
<td>PATYPE</td>
<td>REQUEST-TYPE</td>
<td>The type of retrieval request. Specify a 4-character code, described in Figure 128.</td>
</tr>
</tbody>
</table>

**Retrieval Request Names**

<table>
<thead>
<tr>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Character Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATRSA</td>
<td>RETR-SUBJ-ATTRS</td>
<td>“RSA”</td>
<td>Retrieve all attributes for a subject.</td>
</tr>
<tr>
<td>PATRRA</td>
<td>RETR-RELATION-ATTRS</td>
<td>“RRA”</td>
<td>Retrieve all attributes for a relationship.</td>
</tr>
<tr>
<td>PATRDC</td>
<td>RETR-DESCRIPTION</td>
<td>“RDSC”</td>
<td>Retrieve description text.</td>
</tr>
<tr>
<td>PATRP1</td>
<td>RETR-PLI-DATA</td>
<td>“RPLI”</td>
<td>Retrieve PL/I data text.</td>
</tr>
<tr>
<td>PATRUD1</td>
<td>RETR-USERDATA1</td>
<td>“RUD1”</td>
<td>Retrieve text from user data text set 1.</td>
</tr>
<tr>
<td>PATRUD2</td>
<td>RETR-USERDATA2</td>
<td>“RUD2”</td>
<td>Retrieve text from user data text set 2.</td>
</tr>
</tbody>
</table>

Figure 128 (Part 1 of 2). PATYPE: Request-Type Character Codes
<table>
<thead>
<tr>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Character Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATRUD3</td>
<td>RETR-USERDATA3</td>
<td>&quot;RUD3&quot;</td>
<td>Retrieve text from user data text set 3.</td>
</tr>
<tr>
<td>PATRUD4</td>
<td>RETR-USERDATA4</td>
<td>&quot;RUD4&quot;</td>
<td>Retrieve text from user data text set 4.</td>
</tr>
<tr>
<td>PATRUD5</td>
<td>RETR-USERDATA5</td>
<td>&quot;RUD5&quot;</td>
<td>Retrieve text from user data text set 5.</td>
</tr>
<tr>
<td>PATRA</td>
<td>RETR-ALIASES</td>
<td>&quot;RA&quot;</td>
<td>Retrieve a list of the subject’s aliases.</td>
</tr>
<tr>
<td>PATRSN</td>
<td>RETR-SUBJ-NAMES</td>
<td>&quot;RSN&quot;</td>
<td>Retrieve a list of the subject names in a category</td>
</tr>
<tr>
<td>PATRSAN</td>
<td>RETR-NAMES-ALIASES</td>
<td>&quot;RSAN&quot;</td>
<td>Retrieve a list of the subject names (and their aliases) in a category</td>
</tr>
<tr>
<td>PATRRSR</td>
<td>RETR-SUBJ-RELATS</td>
<td>&quot;RSR&quot;</td>
<td>Retrieve a list of all subject names related to a subject.</td>
</tr>
<tr>
<td>PATRSRC</td>
<td>RSR-BY-CAT</td>
<td>&quot;RSRC&quot;</td>
<td>Retrieve a list of the subject names (in a specified category) related to a subject.</td>
</tr>
<tr>
<td>PATRSRK</td>
<td>RSR-BY-KEYWORD</td>
<td>&quot;RSRK&quot;</td>
<td>Retrieve a list of the subject names (in a specified category and with a specified relationship keyword) related to a subject.</td>
</tr>
<tr>
<td>PATRCR</td>
<td>RETR-CAT-RELATS</td>
<td>&quot;RCR&quot;</td>
<td>Retrieve a list of the types of relationships in which a subject in the specified category can participate.</td>
</tr>
<tr>
<td>PATRSTR</td>
<td>RETR-STRUC-RELTYPES</td>
<td>&quot;RSTR&quot;</td>
<td>Retrieve a list of relationship-types related to a structure-type.</td>
</tr>
<tr>
<td>PATWRTM</td>
<td>WRITE-TO-TERMINAL</td>
<td>&quot;WTRM&quot;</td>
<td>Transmit a message to the terminal.</td>
</tr>
<tr>
<td>PATWPRT</td>
<td>WRITE-TO PRINTER</td>
<td>&quot;WPRT&quot;</td>
<td>Transmit a character string to the Dictionary printer.</td>
</tr>
<tr>
<td>PATWPN</td>
<td>WRITE-TO-PUNCH</td>
<td>&quot;WPCH&quot;</td>
<td>Transmit a character string to the Dictionary punch.</td>
</tr>
<tr>
<td>PATWSTK</td>
<td>WRITE-TO-STACK</td>
<td>&quot;WSTK&quot;</td>
<td>Add an entry to a Dictionary stack.</td>
</tr>
</tbody>
</table>

Figure 128 (Part 2 of 2). PATYPE: Request-Type Character Codes
Location of List

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 (09)</td>
<td>1</td>
<td>PAACCESS</td>
<td>ACCESS-CODE</td>
<td>In a list of data collections associated with a “list-type” retrieve, this 1-character code directs where access is to begin (described in Figure 129).</td>
</tr>
</tbody>
</table>

Positioning in a List

<table>
<thead>
<tr>
<th>COBOL, PL/I, and Assembler Name</th>
<th>Character Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAAFIRST</td>
<td>F</td>
<td>Access begins at the first list element</td>
</tr>
<tr>
<td>PAALAST</td>
<td>L</td>
<td>Access the last list element</td>
</tr>
<tr>
<td>PAAHERE</td>
<td>H</td>
<td>Access begins with the list element identified in the positioning data</td>
</tr>
<tr>
<td>PAANEXT</td>
<td>N</td>
<td>Access begins with the list element immediately following the one identified in the positioning data</td>
</tr>
<tr>
<td>PAADOWN</td>
<td>D</td>
<td>Access begins a number of list elements following the one identified in the positioning data (the number is specified in the PADOWN field)</td>
</tr>
</tbody>
</table>

Figure 129. PAACCESS: Character Codes for Positioning in a List

Note: If value F or L is specified, positioning data is ignored. When the positioning fields are set to blank, values H, N, and D are identical to F: Access begins with the first list element.
### Request for Inserted Blanks

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 (0A)</td>
<td>1</td>
<td>PADAOP</td>
<td>PADA-OPERATION</td>
<td>A 1-character code that specifies whether the Dictionary is to insert blanks in PADA fields when the specified field cannot be retrieved (described in Figure 130).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Character Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PADABLNK</td>
<td>PADA-BLANK</td>
<td>B</td>
<td>Attributes requested but not retrieved are to have the PADA field set to blanks.</td>
</tr>
<tr>
<td>PADANOP</td>
<td>PADA-NON-BLANK</td>
<td>N</td>
<td>No PADA blanking occurs.</td>
</tr>
</tbody>
</table>

Figure 130. PADAOP: Character Codes for Blanking Unretrieved Fields
## Specifying PADAFT Usage

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 (0B)</td>
<td>1</td>
<td>PAFTU</td>
<td>PAFT-USAGE</td>
<td>A 1-character code that specifies the PADAFT usage (see Figure 131).</td>
</tr>
</tbody>
</table>

### Choices for PADAFT Processing

<table>
<thead>
<tr>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Character Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAFTBE</td>
<td>BUILD-AND-EXECUTE</td>
<td>B</td>
<td>Build and execute: The master PADAFT is to be loaded into your program's PADAFT area and the data is retrieved.</td>
</tr>
<tr>
<td>PAFTPB</td>
<td>PRE-BUILD</td>
<td>P</td>
<td>Prebuild: The master PADAFT is built and returned to your program's PADAFT area; no data is to be retrieved.</td>
</tr>
<tr>
<td>PAFTVE</td>
<td>VERIFY-AND-EXECUTE</td>
<td>E</td>
<td>Verify and execute: The PADAFT (either a previously obtained master or one built by the program) is already in your program's PADAFT area. The PADAFT is verified and the data is retrieved.</td>
</tr>
</tbody>
</table>

Figure 131. PAFTU: Character Codes for PADAFT Usage
### Format Control for Output Lines

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 (0C)</td>
<td>1</td>
<td>PACCTRL</td>
<td>CARRIAGE-CONTROL</td>
<td>A 1-character code that specifies the format control for output lines (see Figure 132).</td>
</tr>
</tbody>
</table>

### Spacing for Output Lines

<table>
<thead>
<tr>
<th>PL/I and Assembler Name</th>
<th>Control Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASNGL</td>
<td>&quot; &quot;</td>
<td>Single space</td>
</tr>
<tr>
<td>PADUBL</td>
<td>&quot;0&quot;</td>
<td>Double space</td>
</tr>
<tr>
<td>PAPAGE</td>
<td>&quot;1&quot;</td>
<td>Begin on a new page</td>
</tr>
</tbody>
</table>

Figure 132. PACCTRL: Codes for Format Control of Output Lines

*Note:* The PAPAGE value is effective only for printed output. For an online display (PATYPE=PATWTRM) that includes line spaces, program a single-spaced blank line.

### Further PACA Control Fields

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 (0D)</td>
<td>1</td>
<td>PASTATUS</td>
<td>SELECT-STATUS</td>
<td>Retrieve subject names by status. Blank indicates all subject names to be retrieved regardless of status. Valid only when PACAFM=B.</td>
</tr>
<tr>
<td>16 (10)</td>
<td>4</td>
<td>PADAP</td>
<td>PADAP</td>
<td>Address of the data area (PADA).</td>
</tr>
<tr>
<td>20 (14)</td>
<td>4</td>
<td>PADAL</td>
<td>PADA-LENGTH</td>
<td>The size of the data area (PADA) in bytes (a fullword binary value). The value can be from 1 to 255 for output requests. For retrieval requests, no limit is enforced.</td>
</tr>
<tr>
<td>24 (18)</td>
<td>4</td>
<td>PAFTP</td>
<td>PAFTP</td>
<td>Address of the format table (PADAFT), or of the area in your program into which the PADAFT is to be loaded.</td>
</tr>
<tr>
<td>Offset</td>
<td>Length (in Bytes)</td>
<td>PL/1 and Assembler Name</td>
<td>COBOL Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------</td>
<td>--------------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>28 (1C)</td>
<td>4</td>
<td>PAFTL</td>
<td>PAFT-LENGTH</td>
<td>The length of the format table (PADAFT) in your program (a fullword binary value). If the master PADAFT is to be built (or prebuilt), this value is the size of the area into which the master PADAFT is to be loaded.</td>
</tr>
<tr>
<td>32 (20)</td>
<td>4</td>
<td>PAINCR</td>
<td>COLLECTION-INCREMENT</td>
<td>The collection increment for list-type retrievals. That is, the number of bytes (fullword binary) between the start of the first collection and the beginning of the next. The value specified must be greater than zero and less than the length of the PADA.</td>
</tr>
<tr>
<td>36 (24)</td>
<td>4</td>
<td>PADOWN</td>
<td>DOWN-COUNT</td>
<td>The number (a fullword binary value) of items down a list from a starting position at which a retrieval is to commence.</td>
</tr>
<tr>
<td>40 (28)</td>
<td>4</td>
<td>PANREQ</td>
<td>NUM-ITEMS-REQUESTED</td>
<td>The number (a fullword binary value) of data collections requested for a list-type retrieval.</td>
</tr>
<tr>
<td>44 (2C)</td>
<td>4</td>
<td>PANRETR</td>
<td>NUM-ITEMS-RETRIEVED</td>
<td>The number (a fullword binary value) of data collections retrieved for a list-type retrieval.</td>
</tr>
<tr>
<td>48 (30)</td>
<td>4</td>
<td>PAFTLM</td>
<td>MASTER-PAFT-LENGTH</td>
<td>The actual length (a fullword binary value) of the requested master PADAFT.</td>
</tr>
<tr>
<td>56 (38)</td>
<td>2</td>
<td>PARC</td>
<td>PACA-RC</td>
<td>The return code for the current use of the PACA. The return code values and their meanings are given in “Appendix B.1. Overall Status of a Program Access CALL Statement.”</td>
</tr>
<tr>
<td>Offset</td>
<td>Length (in Bytes)</td>
<td>PL/I and Assembler Name</td>
<td>COBOL Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------</td>
<td>-------------------------</td>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>58 (3A)</td>
<td>2</td>
<td>PART</td>
<td>HIGHEST-PAFT-RC</td>
<td>The highest return code recorded for PADAFT items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In the case of a list-retrieve, the PADAFT return code only shows the status of the last collection retrieved. However, the highest return code from prior collections may be returned in this field. This status can be used to check the degree of success achieved by the CALL request, without checking individual PADAFT return codes. The return code values and their meanings are given in “Appendix B.2. Status of Each Format Table (PADAFT) Entry.”</td>
</tr>
</tbody>
</table>

*Note:* If these values are not used or needed for the Program Access facility call, the PARC and PART fields must be set to blanks.
<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>60 (3C)</td>
<td>8</td>
<td>PACAT</td>
<td>PRINC-SUBJ-CATEGORY</td>
<td>The 8-byte name(^1) of the principal subject category.</td>
</tr>
<tr>
<td>68 (44)</td>
<td>36</td>
<td>PANAME</td>
<td>PRINC-SUBJ-NAME</td>
<td>The principal subject name in I-format.</td>
</tr>
<tr>
<td>104 (68)</td>
<td>12</td>
<td>PARKW</td>
<td>RELATION-KEYWORD</td>
<td>The 12-byte name(^1) of the relationship keyword. For standard relationship-types, the keyword in PARKW is either “WITH” or, in the special case of Element-Element relationships, “CONTAINS.” For extensibility relationship-types, the keyword in PARKW is the relationship keyword obtained from a GUIDE report.</td>
</tr>
<tr>
<td>116 (74)</td>
<td>8</td>
<td>PARCAT</td>
<td>RELA-SUBJ-CAT</td>
<td>The 8-byte name(^1) of the related subject category.</td>
</tr>
<tr>
<td>124 (7C)</td>
<td>36</td>
<td>PARNAME</td>
<td>RELA-SUBJ-NAME</td>
<td>The name, in I-format, of the related subject.</td>
</tr>
<tr>
<td>160 (A0)</td>
<td>8</td>
<td>PARSEQ</td>
<td>RELATION-SEQ</td>
<td>The 8-byte value(^1) of the attribute that controls the sequencing of a relationship. It is left blank if the relationship is not sequenced.</td>
</tr>
</tbody>
</table>

\(^1\) Name is left-adjusted.
PACA Positioning Fields

The following fields are positioning fields and can be modified by the Dictionary as a result of processing list-type retrievals. These fields must be set to blanks when positioning is not desired.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>168 (A8)</td>
<td>3</td>
<td>PAPLINE</td>
<td>POSIT-LINE-NUM</td>
<td>The 3-byte line number for the start of text data retrieval. Left blank if retrieval is to start with the first available line. The value must not be greater than 999 (255 for PL/I data).</td>
</tr>
<tr>
<td>171 (AB)</td>
<td>36</td>
<td>PAPNAME</td>
<td>POSIT-SUBJ-NAME</td>
<td>The I-format subject name or alias that specifies the starting position for a list-type retrieval. Left blank if retrieval is to start at the first available name.</td>
</tr>
<tr>
<td>207 (CF)</td>
<td>12</td>
<td>PAPRKW</td>
<td>POSIT-RELATION-KEYWORD</td>
<td>The 12-byte relationship keyword to be used for the retrieval of subjects related to the primary subject.</td>
</tr>
<tr>
<td>219 (DB)</td>
<td>8</td>
<td>PAPRCAT</td>
<td>POSIT-RELATION-CAT</td>
<td>The 8-byte subject category to be used for the retrieval of subjects related to the primary subject.</td>
</tr>
<tr>
<td>227 (E3)</td>
<td>36</td>
<td>PAPRNAME</td>
<td>POSIT-RELATION-SUBJ</td>
<td>The I-format subject name to be used for the retrieval of subjects related to the primary subject.</td>
</tr>
<tr>
<td>263 (0107)</td>
<td>8</td>
<td>PAPRSEQ</td>
<td>POSIT-RELATION-SEQ</td>
<td>The 8-byte value to be used for the sequencing attribute of a relationship between a related subject and the primary subject. This field may also take on the special format for BITSTART described for the PARSEQ field.</td>
</tr>
</tbody>
</table>
Appendix B.4. The Format Table (PADAFT)

The symbolic names and values for the format table (PADAFT) apply to fields and constants provided with the Dictionary. The PADAFT table entries for standard categories are listed in “Appendix B.6. PADAFT Entries for Dictionary Categories” on page 437.

Every format table (PADAFT) must specify:

- Identification fields (PAFTID and PAFTFM)
- The number of table items (PAFTENTS)
- The length of each entry or group of entries (PAFTENTL)
- A set of fields that describes each data item

When you use one of the programming aids for initialization, many of these fields have appropriate initial values.

The following fields are specified once for a PADAFT:

### Identifying and Control Fields for a PADAFT

<table>
<thead>
<tr>
<th>Offset (in Bytes)</th>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (00)</td>
<td>4</td>
<td>PAFTID</td>
<td>IDENTIFIER</td>
<td>Contains the 4-character identification “PAFT”</td>
</tr>
<tr>
<td>4 (04)</td>
<td>1</td>
<td>PAFTFM</td>
<td>FORMAT-CODE</td>
<td>Contains a 1-character value “A” or “B”</td>
</tr>
<tr>
<td>8 (08)</td>
<td>2</td>
<td>PAFTENTS</td>
<td>NUMBER-OF-ENTRIES</td>
<td>The number of entries contained in the Format Table (halfword binary number)</td>
</tr>
<tr>
<td>10 (0A)</td>
<td>2</td>
<td>PAFTENTL</td>
<td>LENGTH-OF-ENTRY</td>
<td>The length of each entry in the Format Table (halfword binary number set to 24)</td>
</tr>
</tbody>
</table>

The fields in the next table define one PADAFT entry, and repeat (as a group) the number of times specified by the PAFTENTS field.

In Assembler language definitions, the names of the fields for each table entry must be unique, or the base register of a DSECT using one set of names must be moved to the start of each new entry. In the PL/I and COBOL programming aids, these field names are subscripted variables.
### Specification of a PADAFT Entry

<table>
<thead>
<tr>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>PAFNAME</td>
<td>ATTRIBUTE-NAME</td>
<td>The keyword identifying an attribute or data item of the requested entity. Valid keyword names appear in the corresponding master PADAFT.</td>
</tr>
<tr>
<td>2</td>
<td>PAFINST</td>
<td>ATTRIBUTE-INSTANCE</td>
<td>Identifies the occurrence (from 1 to 240) to be retrieved. Nonzero for an attribute or relationship data item that can repeat; must be set to zero for nonrepeating attributes. This will be a binary halfword when in B format.</td>
</tr>
<tr>
<td>1</td>
<td>PAFDTYP</td>
<td>DATA-TYPE</td>
<td>Specifies the type of attribute returned in the data area (PADA) for this table item. The possible values are defined in Figure 133.</td>
</tr>
</tbody>
</table>

### Specifying Data-Type for a PADAFT Entry

<table>
<thead>
<tr>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Character Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAFDA</td>
<td>ALPHA-BETIC</td>
<td>A</td>
<td>Alphabetic data. No DBCS allowed.</td>
</tr>
<tr>
<td>PAFDC</td>
<td>ALPHA-NUMERIC</td>
<td>C</td>
<td>Alphabetic data (characters). For DBCS users, must be a uniform string if a name, or a mixed string if an attribute.</td>
</tr>
<tr>
<td>PAFDD</td>
<td>DICT-13-BYTE-NAME</td>
<td>D</td>
<td>Short form of a Dictionary name (13 bytes, with a user name of 8 bytes). For DBCS users, user name must be a uniform string.</td>
</tr>
<tr>
<td>PAFDG</td>
<td>REPEAT-GRP-IDENT</td>
<td>G</td>
<td>Repeating group identifier</td>
</tr>
<tr>
<td>PAFDI</td>
<td>DICT-36-BYTE-NAME</td>
<td>I</td>
<td>Dictionary subject name (36 bytes, with a user name of 31 bytes). For DBCS users, user name must be a uniform string.</td>
</tr>
<tr>
<td>PAFDL</td>
<td>PAREN-LIST</td>
<td>L</td>
<td>List enclosed in EBCDIC parentheses and separated by EBCDIC commas. For DBCS users, may be a mixed string.</td>
</tr>
<tr>
<td>PAFDN</td>
<td>FIRST-CHAR-ALPHA</td>
<td>N</td>
<td>Name format: First character is alphabetic, remainder of the name is alphabetic. For DBCS users, must be a uniform string.</td>
</tr>
<tr>
<td>PAFDQ</td>
<td>QUOTED-STRING</td>
<td>Q</td>
<td>Character string enclosed in EBCDIC quotation marks. For DBCS users, may be a mixed string.</td>
</tr>
</tbody>
</table>

Figure 133 (Part 1 of 2). PAFDTYP: Character Codes for Attribute Types
<table>
<thead>
<tr>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Character Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAFDS</td>
<td>SPECIAL-STRING</td>
<td>S</td>
<td>Special cases of either the Q or C data type depending on how the data was entered into the Dictionary and subsequently edited. For DBCS users, may be a mixed string.</td>
</tr>
<tr>
<td>PAFDX</td>
<td>HEX-DATA</td>
<td>X</td>
<td>Hexadecimal value, in the format X'xx'</td>
</tr>
<tr>
<td>PAFDZ</td>
<td>ZONED-DECIMAL</td>
<td>Z</td>
<td>Numeric characters. No DBCS allowed.</td>
</tr>
</tbody>
</table>

Figure 133 (Part 2 of 2). PAFDTYP: Character Codes for Attribute Types

Making a PADAFT Entry for Retrieval

<table>
<thead>
<tr>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PAFFUNC</td>
<td>FUNCTION-CODE</td>
<td>Marks an item in the PADAFT for retrieval (see Figure 134)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Character Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAFRETR</td>
<td>RETRIEVE</td>
<td>R</td>
<td>Retrieve the data item</td>
</tr>
<tr>
<td>PAFNOP</td>
<td>NO-OP</td>
<td>N</td>
<td>Ignore the data item</td>
</tr>
</tbody>
</table>

Figure 134. PAFFUNC: Retrieve or Ignore Code

Specifying the PADA Location and Status Field for a PADAFT Entry

<table>
<thead>
<tr>
<th>Length (in Bytes)</th>
<th>PL/I and Assembler Name</th>
<th>COBOL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>PAFDATA</td>
<td>PADA-FIELD-OFFSET</td>
<td>The offset from the start of the PADA to the location of the attribute field (up to 65535 bytes).</td>
</tr>
<tr>
<td>2</td>
<td>PAFLENG</td>
<td>PADA-FIELD-LENGTH</td>
<td>The length of the attribute value in the PADA (up to 255 bytes).</td>
</tr>
<tr>
<td>2</td>
<td>PAFRC</td>
<td>PAFT-RC</td>
<td>Return code: status of the retrieval request for the data item. The return code values and their meanings are given in &quot;Appendix B.2. Status of Each Format Table (PADAFT) Entry&quot; on page 412.</td>
</tr>
</tbody>
</table>
Appendix B.5. The COBOL Sample Program

The COBOL sample program in this appendix is annotated to show you how the parts of a user-written program fit together. However, the listed source code has been edited to remove sections and subroutines that are not critical to the description of Program Access usage.

The Identification Division describes the program and specifies its dependencies and restrictions. The last line of the Identification Division specifies the program identifier.

```
IDENTIFICATION DIVISION.
*============================================
* MODULE NAME: DBDWSUPC
* DESCRIPTIVE NAME: SAMPLE PROGRAM ACCESS USER PROGRAM - COBOL
* FUNCTION:
* THIS MODULE SERVES AS AN EXAMPLE OF A USER PROGRAM WHICH
* USES THE DATA DICTIONARY PROGRAM ACCESS FACILITY. IT IS
* WRITTEN IN THE COBOL LANGUAGE AND PRODUCES A REPORT OF
* UPWARD RELATIONSHIPS OF A GIVEN SUBJECT (I.E., A WHERE-
* USED LIST).
* NOTES:
* DEPENDENCIES: MUST BE LINK-EDITED WITH DBDWNKNC,
* AND ENTRY POINT IN RESULTANT LOAD
* MODULE MUST BE DBDWNKNC
* RESTRICTIONS: MUST BE USED WITH COMPILER OPTION
* 'LANGLEVEL(1)' OR THE 4 COPY STATEMENTS
* MUST BE MODIFIED. SEE COMMENTS IN CODE.
* OPTIONS 'LIB' AND 'QUOTE' MUST ALSO
* BE IN EFFECT.
* PROGRAM-ID. USERPGM
```

The Environment Division specifies the type of processor being used (or required for the program).

```
ENVIRONMENT DIVISION.
*------------------------
CONFIGURATION SECTION.
*------------------------
SOURCE-COMPUTER. IBM-370-168.
OBJECT-COMPUTER. IBM-370-168.
```
The Data Division specifies the data areas used by the program. The first part of the Data Division copies the set of initialized constants from the DBDWDVAC member of DBD.MACLIB, which initializes constants needed by a Program Access user program:

```
DATA DIVISION.
*===================

WORKING-STORAGE SECTION.
* -----------------------
*** VALUES FOR PACA, PADAFT AND PAP ***
* FOLLOWING COPY STATEMENT FOR OS LANGLVL(2):
* COPY DBDWDVAC.
* FOLLOWING COPY STATEMENT FOR DOS OR FOR OS LANGLVL(1):
  01 PA-FIELD-VALUE-DEFNS COPY DBDWDVAC.
```

The next part copies the PADAFT definition from the DBDWDFTC member of DBD.FORMAT and defines the number of PADAFT entries to be 4.

```
*** PA DATA AREA FORMAT TABLE ***
* FOLLOWING COPY FOR OS LANGLVL(2):
* COPY DBDWDFTC
* REPLACING WFTPA-ENTRIES BY 4
* DBDWFTPA BY PADAFT.
* FOLLOWING COPY FOR DOS OR FOR OS LANGLVL(1):
  01 PADAFT COPY DBDWDFTC
    REPLACING WFTPA-ENTRIES BY 4
    DBDWFTPA BY PADAFT.
```

The next part initializes a PADA area and defines its fields. Note that there is no Dictionary source segment for the PADA. The format is determined by the user program. In this example, the PADA is designed to hold 20 instances of data retrieved by a Retrieve Related Subjects (RSR) request.

```
*** PA DATA AREA ***
01 PADA.
  03 PADA-LINE OCCURS 20 TIMES.
    05 REL-KEYWORD PIC X(12).
    05 REL-SUBJ-CATG PIC X(8).
    05 SUBJ-STATUS PIC X.
    05 SUBJ-CODE PIC X.
    05 SUBJ-NAME PIC X(31).
    05 OCCURRENCE PIC XXX.
    05 DIRECTION PIC X.
    88 UPWARD VALUE "U".
```

To minimize the number of changes within a PACA from request to request, the program uses two different PACAs for retrieval and output requests. The next part...
copies the PACA definition from the DBDWDCAC member of DBD.FORMAT. The PROCESS-PACA is used for retrieval.

*** PROCESS PA CONTROL AREA ***

* FOLLOWING COPY FOR OS LANGLEVEL(2):
* COPY DBDWDCAC
  * REPLACING DBDWCAPA BY PROCESS-PACA.
* FOLLOWING COPY FOR DOS OR FOR OS LANGLEVEL(1):
  01 PROCESS-PACA COPY DBDWDCAC
  REPLACING DBDWCAPA BY PROCESS-PACA.

---

The next part explicitly defines a PACA for use as the output PACA.

*** OUTPUT PA CONTROL AREA ***

01 OUTPUT-PACA.

*** NOTE THAT DEFINING THIS AREA EXPLICITLY HERE ***

*** PERMITS US TO AVOID EXCESSIVE NAME-QUALIFICATION ***

*** IN THE PROGRAM, BUT REQUIRES THAT THIS DEFINITION ***

*** BE UPDATED FOR ANY CORRESPONDING CHANGE TO DBDWDCAC ***

03 IDENTIFIER-O USAGE DISPLAY PICTURE X(4).
03 FORMAT-CODE-O USAGE DISPLAY PICTURE X.
03 REQUEST-TYPE-O USAGE DISPLAY PICTURE X(4).
03 ACCESS-CODE-O USAGE DISPLAY PICTURE X.
03 PADA-OPERATION-O USAGE DISPLAY PICTURE X.
03 PADA-USAGE-O USAGE DISPLAY PICTURE X.
03 CARRIAGE-CTL-O USAGE DISPLAY PICTURE X.
03 RESERVED1-O USAGE DISPLAY PICTURE X(3).

* THE FOLLOWING FIELD WILL BE SET IN "CBLTDBD".
  03 PADA-ADDRESS-O USAGE COMP PICTURE 9(5).
  03 PADA-LENGTH-O USAGE COMP PICTURE 9(5).

* THE FOLLOWING FIELD WILL BE SET IN "CBLTDBD".

  03 PAFT-ADDRESS-O USAGE COMP PICTURE 9(5).
  03 PAFT-LENGTH-O USAGE COMP PICTURE 9(5).
  03 COLLECTION-INC-R-O USAGE COMP PICTURE 9(5).
  03 DOWN-COUNT-O USAGE COMP PICTURE 9(5).
  03 NUM-ITEMS-REQ-O USAGE COMP PICTURE 9(5).
  03 NUM-ITEMS-RETR-O USAGE COMP PICTURE 9(5).
  03 MASTER-PAFT-LOG-O USAGE COMP PICTURE 9(5).
  03 RESERVED2-O USAGE COMP PICTURE 9(5).
  03 PACA-RC-O USAGE COMP PICTURE 999.

  88 OUT-PACOK
   VALUE 0.
  03 HIGHEST-PAFT-RC-O USAGE COMP PICTURE 999.
  03 PRIN-SUBJ-CATG-O USAGE DISPLAY PICTURE X(8).
  03 PRIN-SUBJ-NAME-O USAGE DISPLAY PICTURE X(36).
  03 RELATION-KEYWORD-O USAGE DISPLAY PICTURE X(12).
  03 RELA-SUBJ-CAT-O USAGE DISPLAY PICTURE X(8).
  03 RELA-SUBJ-NAME-O USAGE DISPLAY PICTURE X(36).
  03 RELATION-SEQ-O USAGE DISPLAY PICTURE X(8).
  03 POSIT-LINE-NUM-O USAGE DISPLAY PICTURE X(36).
  03 POSIT-SUBJ-NAME-O USAGE DISPLAY PICTURE X(36).
  03 POSIT-RELA-KEY-O USAGE DISPLAY PICTURE X(8).
  03 POSIT-RELA-CAT-O USAGE DISPLAY PICTURE X(8).
  03 POSIT-RELA-SUBJ-O USAGE DISPLAY PICTURE X(36).
  03 POSIT-RELA-SEQ-O USAGE DISPLAY PICTURE X(8).
The rest of the Data Division defines and initializes data areas used by the program.

The last part of the Data Division defines the parameter string used to pass data from the EXECUTE command to the program.

```
LINKAGE SECTION.
* --------------------
*** PARAMETER LIST ***
* FOLLOWING COPY FOR OS LANGLEVEL(2):
  * COPY DBDWDUDC.
* FOLLOWING COPY FOR DOS OR FOR OS LANGLEVEL(1):
  01 DBDWPAP COPY DBDWDUDC.
```

The Procedure Division contains the program itself. The first part establishes linkage between the Dictionary and the program:

```
PROCEDURE DIVISION.
*------------------------
*** MAINLINE LOGIC ***
SET-UP.
* -------
ENTER LINKAGE.
ENTRY "DBDWUP" USING DBDWPAP.
ENTER COBOL.
MOVE URC-OK TO USER-RET-CODE.
```

The next part parses the input string and puts the parsed data into an area called LIST-OF-OPTIONS-LINE. The program uses this area to control processing. This area is also used as a title line for the report the program produces.

This part sets up the control area (PACA) used to send output to the user.

```
PROCESS-USER-REQUEST.
* -----------------------
  MOVE LIST-OF-OPTIONS-LINE TO REPORT-TITLE-LINE.
  MOVE 0 TO PACA-RC-0.
  MOVE PACA-ID TO IDENTIFIER-O.
  MOVE PACAFMB TO FORMAT-CODE-O.
  MOVE WRITE-TO-TERMINAL TO REQUEST-TYPE-O.
  MOVE PASNGL TO CARRIAGE-CTL-0.
  MOVE OUT-LINE-LEN TO PADA-LENGTH-O.
```
If EXPLAIN text was not requested, print the report’s column heading.

IF NO-POSITION = "NO"
MOVE REPORT-HEADER-LINE TO REPORT-DETAIL-LINE
PERFORM OUTPUT-SEGMENT

If the subject category and the subject name are not specified by the user, do not attempt to retrieve anything.

IF (CATEGORY-OUT = SPACES) OR (SUBJECT-OUT = SPACES)
  NEXT SENTENCE

Otherwise, initialize the format table (PADAFT), by calling the INITIALIZE-PADAFT subroutine, for a retrieval request (RSR). Initialize the control area (PACA), by calling the INITIALIZE-PACA subroutine, for a retrieval request (RSR). Retrieve all subjects related to the input subject, ignoring any subject whose relationship direction isn’t upward, by calling the RETRIEVE-AND-TEST subroutine.

ELSE
PERFORM INITIALIZE-PADAFT
PERFORM INITIALIZE-PACA
PERFORM RETRIEVE-AND-TEST
  UNTIL (NOT PACOK) OR (NOT OUT-PACOK).

This section writes the end-of-report summary and returns control to the Dictionary (GOBACK).

WRAP-UP.
* ---------
  IF OUT-PACOK
    MOVE UP-REL-COUNT TO UP-REL-OUT
    MOVE SPACES TO REPORT-DETAIL-LINE
    PERFORM OUTPUT-SEGMENT
    IF OUT-PACOK
      MOVE END-OF-REQUEST-LINE TO REPORT-DETAIL-LINE
      PERFORM OUTPUT-SEGMENT.
    GOBACK.
The INITIALIZE-PADAFT subroutine initializes the format table (PADAFT) used in the RSR request. For each relationship found, four fields are retrieved:

RELKWORD  The forward relationship keyword
RELCAT    The name of the related subject category
RELSUBJ   The name of the related subject
DIRECT    The direction of the relationship between the input subject and the retrieved subject

Other fields that could be retrieved are:

SEQVAL    The sequence attribute value
INVKWORD  The inverse relationship keyword

However, this program does not retrieve those fields.

INITIALIZE-PADAFT.
*  ---------------

Number of entries in the PADAFT is 4.

MOVE 4 TO NUMBER-OF-ENTRIES.

The instance of each of the four entries in the PADAFT is 0—as it always is for a list-type retrieval.

MOVE 0 TO ATTRIBUTE-INSTANCE (1), ATTRIBUTE-INSTANCE (2)
       ATTRIBUTE-INSTANCE (3), ATTRIBUTE-INSTANCE (4).

Indicate in each PADAFT entry that the field is to be retrieved, not ignored.

MOVE RETRIEVE TO FUNCTION-CODE (1), FUNCTION-CODE (2)
       FUNCTION-CODE (3), FUNCTION-CODE (4).
Move keyword, length, and offset values into the four PADAFT entries. Because of the method by which PADA-FIELD-OFFSET is set, the data within a list instance will be contiguous in the PADA.

MOVE "RELKWORD" TO ATTRIBUTE-NAME (1).
MOVE 12 TO PADA-FIELD-LENGTH (1).
MOVE 0 TO PADA-FIELD-OFFSET (1).

MOVE "RELCAT" TO ATTRIBUTE-NAME (2).
MOVE 8 TO PADA-FIELD-LENGTH (2).
ADD PADA-FIELD-LENGTH (1) PADA-FIELD-OFFSET (1)
    GIVING PADA-FIELD-OFFSET (2).

MOVE "RELSUBJ" TO ATTRIBUTE-NAME (3).
MOVE 36 TO PADA-FIELD-LENGTH (3).
ADD PADA-FIELD-LENGTH (2) PADA-FIELD-OFFSET (2)
    GIVING PADA-FIELD-OFFSET (3).

MOVE "DIRECT" TO ATTRIBUTE-NAME (4).
MOVE 1 TO PADA-FIELD-LENGTH (4).
ADD PADA-FIELD-LENGTH (3) PADA-FIELD-OFFSET (3)
    GIVING PADA-FIELD-OFFSET (4).

The INITIALIZE-PACA subroutine initializes the control area (PACA) for the RSR retrieval request. This subroutine is called until an error occurs, or until there are no more instances to retrieve.

INITIALIZE-PACA.
* ----------------
* NOTE THAT INITIALIZE-PAFT MUST BE PERFORMED BEFORE THIS
* *

Specify the RSR request type.

MOVE RETR-SUBJ-RELATS TO REQUEST-TYPE.

Start at the beginning of the list.

MOVE PAAFIRST TO ACCESS-CODE.
MOVE PADA-NON-BLANK TO PADA-OPERATION.
MOVE VERIFY-AND-EXECUTE TO PAFT-USAGE.
Calculate the PADAFT length.

MULTIPLY LENGTH-OF-ENTRY BY NUMBER-OF-ENTRIES
GIVING PAFT-LENGTH.
ADD PAFT-HDR-LENGTH TO PAFT-LENGTH.

Calculate the increment for data returned in the PADA. The instances of the list will be contiguous because of how COLLECTION-INCREMENT is set.

ADD PADA-FIELD-LENGTH (4) PADA-FIELD-OFFSET (4)
GIVING COLLECTION-INCREMENT.

Specify the number of relationships per request.

MOVE RSR-NUM-RQSTD TO NUM-ITEMS-REQUESTED.

Calculate the total data area (PADA) length.

MULTIPLY COLLECTION-INCREMENT BY NUM-ITEMS-REQUESTED
GIVING PADA-LENGTH.
MOVE 0 TO PACA-RC.
MOVE 0 TO HIGHEST-PAFT-RC.

The program is not using these fields for positioning.

MOVE SPACES TO POSIT-RELA-KEYWORD.
MOVE SPACES TO POSIT-RELA-CAT.
MOVE SPACES TO POSIT-RELA-SUBJ.
MOVE SPACES TO POSIT-RELA-SEQ.
The RETRIEVE-AND-TEST subroutine issues a CALL to the Dictionary to request an RSR (relationship attribute) retrieval request. When the Dictionary returns to the program, a return code is set in the PACA. If the return code is not OK (that is, if it is neither X'00', X'04' (end of data), nor X'08' (no data found)), subroutine SET-USER-RTN-CODE is called to analyze the return code.

```
RETRIEVE-AND-TEST.
  ---------------------
  CALL "CBLTDBD" USING PROCESS-PACA, PADAFT, PADA.
  IF (NOT PACOK) AND (NOT END-OF-DATA) AND
     (NOT NO-DATA-FOUND)
    PERFORM SET-USER-RTN-CODE
```

At this point, the program analyzes each retrieved relationship instance to detect the ones that are in an upward direction.

```
ELSE
  MOVE 1 TO I
  PERFORM CLASSIFY-RELATION
      UNTIL (I > NUM-ITEMS-RETRIEVED) OR (NOT OUT-PACOK).
```

The next time RETRIEVE-AND-TEST is called, the following statement ensures that retrieval starts where it left off.

```
MOVE PAANEXT TO ACCESS-CODE.
```

The SET-USER-RTN-CODE subroutine analyzes a bad return code and sends an appropriate message to the user who invoked the program.

```
SET-USER-RTN-CODE.
  ----------------
  MOVE URC-ERROR TO USER-RET-CODE.
  IF NO-SUCH-SUBJ
    MOVE NO-SUBJ-FOUND-MSG TO REPORT-DETAIL-LINE
  ELSE
    IF BAD-PRIN-CAT
      MOVE NO-CATG-FOUND-MSG TO REPORT-DETAIL-LINE
    ELSE
      IF USER-NOT-AUTHORIZED
        MOVE NO-AUTHORIZATION-MSG TO REPORT-DETAIL-LINE
      ELSE
        MOVE PACA-RC TO FAILURE-RET-CODE
        MOVE PROG-FAILURE-MSG TO REPORT-DETAIL-LINE
        MOVE URC-SEVERE TO USER-RET-CODE.
    PERFORM OUTPUT-SEGMENT.
```

Appendix B. Reference Material for the Program Access Facility  435
The **CLASSIFY-RELATION** subroutine analyzes a retrieved relationship instance. If it is an upward relationship, the subroutine sends an output line to the report for the user.

```
CLASSIFY-RELATION.
  * ---------------------
    IF UPWARD (I)
      MOVE SPACES TO COLLECTION-OUT
      MOVE CORR PADA-LINE (I) TO COLLECTION-OUT
      MOVE CORR DELIMITERS TO COLLECTION-OUT
      PERFORM OUTPUT-SEGMENT
      ADD 1 TO UP-REL-COUNT.
      ADD 1 TO I.
```

The **OUTPUT-SEGMENT** subroutine issues a CALL to the Dictionary to request a line to be printed or displayed. Note that a dummy PADAFT (called NUL) is used. The Dictionary will ignore the PADAFT specified with an output request, but COBOL requires the correct number of arguments in the CALL statement.

```
OUTPUT-SEGMENT.
  * ---------------------
    IF LINE-COUNT = ZERO
      MOVE PAPAGE TO CARRIAGE-CTL-O
      CALL "CBLTDDB" USING OUTPUT-PACA, NUL, REPORT-TITLE-LINE
      MOVE PASNL TO CARRIAGE-CTL-O
      CALL "CBLTDDB" USING OUTPUT-PACA, NUL, BLANK-LINE
      MOVE LINE-CT-RESET TO LINE-COUNT.
      CALL "CBLTDDB" USING OUTPUT-PACA, NUL, REPORT-DETAIL-LINE.
      SUBTRACT 1 FROM LINE-COUNT.
```
Appendix B.6. PADAFT Entries for Dictionary Categories

This appendix contains reference material for use when you are coding programs to retrieve attribute data from the Dictionary data bases, using the Program Access facility (for attribute-type retrieval requests) as described in Chapter 12, "Writing Programs to Access Dictionary Data" on page 209. The appendix identifies the master PADAFT contents for all Dictionary-supplied categories, and their established relationship-types. It tabulates the attribute names, instance numbers, field lengths, and data types that appear to a user program in the master PADAFT fields. (Master PADAFT contents for extensibility categories and relationship-types are available with GUIDE reports. Master PADAFT contents for list-type retrieval requests are described in Chapter 12, "Writing Programs to Access Dictionary Data" on page 209.)

Each attribute name describing subject data corresponds to a keyword listed in Appendix A of OS/VSE DB/DC Data Dictionary Terminal User's Guide and Command Reference (unless otherwise indicated). The relationship data that is specified in Appendix B of that publication is also identified by attribute keywords of the relationships, with exceptions noted. A summary of keywords for Extensibility categories and their relationship-types is in Appendix C.

Several points should be considered:

- A choice of keywords may be permitted on commands for the same attribute. In each case, one keyword has been selected for use with the Program Access facility, and only that keyword is recognized in retrieval calls.

- Some keywords in the referenced appendixes do not appear in a master PADAFT, and are not accessible as "attributes" through Program Access but are represented in other ways. For example, SRCHOCR, an attribute keyword for the SEGMENT category, is used in commands to identify which of 5 possible instances of the attribute, SRCHNAME, is to be accessed. Using the Program Access facility, this function is performed by the instance number. Consequently, there is no need for the SRCHOCR keyword. Another example is RMNAME in DBS category, which actually represents four other keywords. In this case, the individual (sub-) keywords are used, and RMNAME is not recognized.

- Keywords for Description, User Data, and PL/I Data text are replaced by the special master PADAFTs for text retrieval, as described in the section "Retrieving Text Data (RDSC, RUDn, RPLI Calls)" on page 238.

- Keywords having data type "G" do not appear in the referenced appendixes. These are the headers for repeating attribute groups; refer to "Repeating Groups of Attributes" on page 465.

The tables of attributes are listed in the same order as the tables of keywords contained in Appendixes A and B of OS/VSE DB/DC Data Dictionary Terminal User's Guide and Command Reference.
Standard Dictionary Categories

DATABASE Category

```
CATEGORY: DATABASE
ALIASES: DBS
        D
        FILE
        FIL
        L

ATTRIBUTES: (See Notes 1,2,3)

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBACCS</td>
<td>00</td>
<td>6</td>
<td>A</td>
</tr>
<tr>
<td>INDACCS</td>
<td>00</td>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>DBRMNM</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>ANCH</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>DBMAXRBR</td>
<td>00</td>
<td>8</td>
<td>Z</td>
</tr>
<tr>
<td>DDBYTES</td>
<td>00</td>
<td>8</td>
<td>Z</td>
</tr>
<tr>
<td>PROT</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>DOSCOMP</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>PASSWD</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>QSEC</td>
<td>00</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>USEC</td>
<td>00</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>DATNUM</td>
<td>00</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>DATMOCC</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>DATASET</td>
<td>01</td>
<td>0</td>
<td>G</td>
</tr>
<tr>
<td>DD1</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>DD2</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>OVFLW</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>DEVICE</td>
<td>00</td>
<td>4</td>
<td>Z</td>
</tr>
<tr>
<td>DSGLABEL</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>SCAN</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>BLKSIZE1</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>RCDSIZE1</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>BLKSIZE2</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>RCDSIZE2</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>FRSPCBF</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>FRSPCSP</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>MODEL</td>
<td>00</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>RECFM</td>
<td>00</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>REL</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>SRCFLD</td>
<td>00</td>
<td>8</td>
<td>C</td>
</tr>
<tr>
<td>UOW</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>UOWOFL</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>ROOT</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>ROOTOFL</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
</tbody>
</table>

(See Note 4)

(see Note 5)

Figure 135. DATABASE Category
```
Notes to Figure 135 on page 438:

1. *FRSPC* is represented by two attribute names, *FRSPCBF* and *FRSPCSP*.

2. *RMNAME* is represented by four attribute names, *DBRMNM*, *ANCH*, *DBMAXRB*, and *DBBYTES*.

3. *DATMUSER* is represented by *DATMNUM* and *DATMOCC*.

4. *DSGNMBR* is represented by the instance number on the *PADAFT* attribute name *DATASET*. A new synonym for this attribute is *AREA*.

5. *SIZE* is a new synonym for *BLKSIZE1*. 
**ELEMENT Category**

<table>
<thead>
<tr>
<th>CATEGORY:</th>
<th>ELEMENT</th>
<th>SUBJECT CODE:</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES:</td>
<td>DTE  E FIELD  FLD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Attributes:** (See Notes 1,2,3)

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTES</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>TYPE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>DATE</td>
<td>00</td>
<td>6</td>
<td>C</td>
</tr>
<tr>
<td>DIGITS</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>DECIMALS</td>
<td>00</td>
<td>4</td>
<td>Z</td>
</tr>
<tr>
<td>QSEC</td>
<td>00</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>USEC</td>
<td>00</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>COBPIC</td>
<td>00</td>
<td>30</td>
<td>S</td>
</tr>
<tr>
<td>JUST</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>BLANK</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>USAGE</td>
<td>00</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>SYNCH</td>
<td>00</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>SIGN</td>
<td>00</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>PLIPIC</td>
<td>00</td>
<td>30</td>
<td>S</td>
</tr>
<tr>
<td>PLINIT</td>
<td>00</td>
<td>30</td>
<td>S</td>
</tr>
<tr>
<td>PLIPMT</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>SCALE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>MODE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ALIGNED</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>DIM1L</td>
<td>00</td>
<td>6</td>
<td>Z</td>
</tr>
<tr>
<td>DIM1U</td>
<td>00</td>
<td>6</td>
<td>Z</td>
</tr>
<tr>
<td>DIM2L</td>
<td>00</td>
<td>6</td>
<td>Z</td>
</tr>
<tr>
<td>DIM2U</td>
<td>00</td>
<td>6</td>
<td>Z</td>
</tr>
<tr>
<td>DIM3L</td>
<td>00</td>
<td>6</td>
<td>Z</td>
</tr>
<tr>
<td>DIM3U</td>
<td>00</td>
<td>6</td>
<td>Z</td>
</tr>
<tr>
<td>VALUE</td>
<td>00</td>
<td>120</td>
<td>S</td>
</tr>
<tr>
<td>OCCURS</td>
<td>00</td>
<td>120</td>
<td>S</td>
</tr>
<tr>
<td>EDITMODE</td>
<td>01</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>EDITTYPE</td>
<td>01</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ROUTINE</td>
<td>01</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>TABLE</td>
<td>01</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>EUNIT</td>
<td>01</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>EDITLEN</td>
<td>01</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>ACTION</td>
<td>01</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>EDITNUM</td>
<td>01</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>EDITCC</td>
<td>01</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>EDITMODE</td>
<td>02</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>EDITYPE</td>
<td>02</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ROUTINE</td>
<td>02</td>
<td>8</td>
<td>N</td>
</tr>
</tbody>
</table>

*(See Note 4)*

*(See Note 5)*

---

Figure 136 (Part 1 of 2). ELEMENT Category
Figure 136 (Part 2 of 2). ELEMENT Category

Notes to Figure 136 on page 440:

1. PLIDIM, PLIDIM1, PLIDIM2, and PLIDIM3, and their synonyms, are represented by the attribute names DIM1L, DIM1U, DIM2L, DIM2U, DIM3L, and DIM3U.

2. EDITUSER is represented by EDITNUM and EDITOCC.

3. DTENPREC is represented by DIGITS and DECIMALS.

4. MONTH, DAY, and YEAR are not available as separate attributes.

5. PICTURE is represented by the attribute name COBPIC.
PCB Category

<table>
<thead>
<tr>
<th>CATEGORY:</th>
<th>PCB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES:</td>
<td>P</td>
</tr>
</tbody>
</table>

ATTRIBUTES:

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBTYPE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>POS</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>PROCOPT</td>
<td>00</td>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>KEYLEN</td>
<td>00</td>
<td>4</td>
<td>Z</td>
</tr>
<tr>
<td>PROCSEQ</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>QSEC</td>
<td>00</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>USEC</td>
<td>00</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>TPTYPE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>MODIFY</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>LTERM</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>EXPRESS</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ALTRESP</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>SAMETRM</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

SUBJECT CODE: 4

(See note 1)

(See note 2)

Figure 137. PCB Category

Notes to Figure 137:

1. The attribute name DBTYPE, shown above, corresponds to the first TYPE keyword in the table for PCB subject data.

2. The attribute name TPTYPE, shown above, corresponds to the second TYPE keyword in the table for PCB subject data.
### SEGMENT Category

<table>
<thead>
<tr>
<th>CATEGORY:</th>
<th>SEGMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES:</td>
<td>SEG</td>
</tr>
<tr>
<td>RECORD</td>
<td>REC</td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>

**SUBJECT CODE:** 2

**ATTRIBUTES:** (See notes 1, 2, 3)

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>DATA TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINBYTES</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>MAXBYTES</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>NUMDTES</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>DATE</td>
<td>00</td>
<td>6</td>
<td>C (See note 4)</td>
</tr>
<tr>
<td>SEGALIGN</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>XFLDNAME</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>ISSNAME</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>CONST</td>
<td>00</td>
<td>5</td>
<td>X</td>
</tr>
<tr>
<td>NULLVAL</td>
<td>00</td>
<td>5</td>
<td>X</td>
</tr>
<tr>
<td>EXTRTN</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>SRCHNAME</td>
<td>01</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SRCHNAME</td>
<td>02</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SRCHNAME</td>
<td>03</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SRCHNAME</td>
<td>04</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SRCHNAME</td>
<td>05</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SUBSEQ</td>
<td>01</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SUBSEQ</td>
<td>02</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SUBSEQ</td>
<td>03</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SUBSEQ</td>
<td>04</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SUBSEQ</td>
<td>05</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>DDATA</td>
<td>01</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>DDATA</td>
<td>02</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>DDATA</td>
<td>03</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>DDATA</td>
<td>04</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>DDATA</td>
<td>05</td>
<td>13</td>
<td>D</td>
</tr>
</tbody>
</table>

Figure 138. SEGMENT Category

**Notes to Figure 138:**

1. *The* BYTES, LENGTH, *and* LEN *keywords are represented by* MINBYTES *and MAXBYTES.*

2. *SRCHOCHR is represented by an instance number on SRCHNAME.*

3. *SBSROCHR is represented by an instance number on SUBSEQ* *and DDATA.*

4. *MONTH, DAY, and YEAR are not available as separate attributes.*
SYSTEM Category

| CATEGORY:  | SYSTEM       | SUBJECT CODE: | 5 |
| ALIASES:   | SYS          |               |   |
|           | Y            |               |   |
| ATTRIBUTES:| NONE         |               |   |

Figure 139. SYSTEM Category

PROGRAM Category

<p>| CATEGORY:  | PROGRAM      | SUBJECT CODE: | 7 |
| ALIASES:   | PGM          |               |   |
|           | G            |               |   |
| ATTRIBUTES:|             |               |   |</p>
<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGMLANG</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>PGMTYPE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>SIZE</td>
<td>00</td>
<td>8</td>
<td>Z</td>
</tr>
</tbody>
</table>

Figure 140. PROGRAM Category
**PSB Category**

<table>
<thead>
<tr>
<th>CATEGORY:</th>
<th>PSB</th>
<th>SUBJEC CODE:</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES:</td>
<td>B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ATTRIBUTES:**

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSBLANG</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>IOASIZE</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>SSASIZE</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>MAXQ</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>CMPAT</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>IOEROPN</td>
<td>00</td>
<td>4</td>
<td>Z</td>
</tr>
<tr>
<td>IOWTR</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>PSBTYPE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>OVLY</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>IQF</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>OLIC</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

Figure 141. PSB Category

**TRANSACTION Category**

<table>
<thead>
<tr>
<th>CATEGORY:</th>
<th>TRANSACT</th>
<th>SUBJECT CODE:</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES:</td>
<td>TRN T</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ATTRIBUTES:**

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRTYPE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>LITYPE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>WFI</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>MSGSEG</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>MSGRESP</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>INQTRAN</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>INQRECV</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>MODE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>EDITNAME</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>SPASIZE</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>SEGNO</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>SEGSIZE</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>EDITTRAN</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>SPAFIX</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>MAXRGN</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
</tbody>
</table>

Figure 142. TRANSACTION Category
SYSDEF Category

CATEGORY: SYSDEF
ALIASES: SYSTEM DEFINITION
       SDF
       N
ATTRIBUTES: NONE

SUBJECT CODE: 11

Figure 143. SYSDEF Category

JOB Category

CATEGORY: JOB
ALIASES: J
ATTRIBUTES: NONE

SUBJECT CODE: 9

Figure 144. JOB Category

MODULE Category

CATEGORY: MODULE
ALIASES: MOD
       M
ATTRIBUTES: NONE

SUBJECT CODE: 8

Figure 145. MODULE Category
**DDUSER Category**

<table>
<thead>
<tr>
<th>CATEGORY:</th>
<th>DDUSER</th>
<th>SUBJECT CODE:</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES:</td>
<td>DDU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTES:</td>
<td>(See note 1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECSNBR</td>
<td>00</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>SECLNBR</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>PASSWORD</td>
<td>00</td>
<td>8</td>
<td>C</td>
</tr>
</tbody>
</table>

Figure 146. DDUSER Category

**Notes:**

1. The SECUDPTR keyword is represented by SECSNBR and SECLNBR.
### Database with Segment

**Reltype:** DATABASE/WITH/SEGMENT  
**Aliases:** SEGMENT/WITH/DATABASE

**Attributes:** (See notes 1, 2, 3, 4, 5)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Inst</th>
<th>Space</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHYPAR</td>
<td>00</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>LEVEL</td>
<td>00</td>
<td>2</td>
<td>Z</td>
</tr>
<tr>
<td>DSG</td>
<td>00</td>
<td>2</td>
<td>Z</td>
</tr>
<tr>
<td>PAIREDLC</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>PCPNTRS</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>SECTYPE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>FREQ</td>
<td>00</td>
<td>11</td>
<td>Z</td>
</tr>
<tr>
<td>HP</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>PTP</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>LTP</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>LPF</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>IRULE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>DRULE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>PRULE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>WHRULE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>COMPRTN</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>CCRTNOPT</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>COMPINIT</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>CTR</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>SSPT</td>
<td>00</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>LPDSNAME</td>
<td>00</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>LPSDBNAME</td>
<td>00</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>KEYSTRG</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>SRCSNAME</td>
<td>01</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SRCDNAME</td>
<td>01</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SRCKDOPT</td>
<td>01</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>SRCSNAME</td>
<td>02</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SRCDNAME</td>
<td>02</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>SRCKDOPT</td>
<td>02</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>PCHILD</td>
<td>01</td>
<td>0</td>
<td>G</td>
</tr>
<tr>
<td>PCSGNME</td>
<td>00</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>LCHILD</td>
<td>01</td>
<td>0</td>
<td>G</td>
</tr>
<tr>
<td>LCSGNAME</td>
<td>00</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>LCBNAME</td>
<td>00</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>PAIR</td>
<td>00</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>POINTER</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>INDXFLNM</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>LCIRRULES</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>SECINDEX</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

*Figure 147. DATABASE/WITH/SEGMENT Relationship-Type*
Notes to Figure 147 on page 448:

1. *PARENT* is represented by *PHYPAR, PCPNTRS, and LPARENT* (See note 2)

2. *LPARENT* is represented by *LPSGNAME, KEYSTRG, and LPDBNAME.*

3. *RULES* is represented by *IRULE, DRULE, RRULE, and WHRRULE.*

4. *SRCOCR* is represented by an instance number on *SRCSNAME, SRCDNNAME, and SRCKDOPT.*

5. *SOURCE1 and SOURCE2* are represented by instances 1 and 2, respectively, on the attribute names *SRCSNAME, SRCDNNAME, and SRCKDOPT.*

6. *COMPRTN* represents only the first of the three possible values that can be specified for this keyword. The others are represented by *CCRTNOPT and COMPINIT.*

7. The *LCHILD* attribute differs from the *LCHILD* keyword in the *DATABASE WITH SEGMENT relationship type.* The *LCHILD* attribute name is a group header only, and has no value of its own. Values for the *LCHILD* keyword specification are represented by *LCSGNAME and LCDBNAME.*

8. This is the second of the two *POINTER* keywords, the other being represented by the six attribute names, *HP, PTP, LTP, LPP, CTR, and PAIRDLC.*
### PCB WITH SEGMENT

**RELTYPE:** PCB/WITH/SEGMENT  
**ALIASES:** SEGMENT/WITH/PCB  

**ATTRIBUTES:** (See notes 1 and 2)

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARENT</td>
<td>00</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>PROCPT</td>
<td>00</td>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>SSPTRSEN</td>
<td>00</td>
<td>8</td>
<td>S</td>
</tr>
<tr>
<td>INDCNAME</td>
<td>01</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>INDCNAME</td>
<td>02</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>INDCNAME</td>
<td>03</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDCNAME</td>
<td>32</td>
<td>8</td>
<td>N</td>
</tr>
</tbody>
</table>

**Figure 148.** PCB/WITH/SEGMENT Relationship-Type

**Notes:**

1. *INDEX* is represented by INDCOCR (see Note 2) and INDCNAME.

2. *INDCOCR* is represented by an instance number on INDCNAME.
### SEGMENT WITH ELEMENT

<table>
<thead>
<tr>
<th>RELTYPE:</th>
<th>SEGMENT/WITH/ELEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES:</td>
<td>ELEMENT/WITH/SEGMENT</td>
</tr>
</tbody>
</table>

**ATTRIBUTES:**

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>BITSTART</td>
<td>00</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>FIRSTLEV</td>
<td>00</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>GENFLD</td>
<td>00</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>REPLACE</td>
<td>00</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>SEQNUM</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
</tbody>
</table>

*Figure 149. SEGMENT/WITH/ELEMENT Relationship-Type*

### ELEMENT CONTAINS ELEMENT

<table>
<thead>
<tr>
<th>RELTYPE:</th>
<th>ELEMENT/CONTAINS/ELEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES:</td>
<td>ELEMENT/CONTAINED_IN/ELEMENT</td>
</tr>
<tr>
<td>ALIASES:</td>
<td>ELEMENT/REDEFINES/ELEMENT</td>
</tr>
</tbody>
</table>

**ATTRIBUTES:**

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>BITSTART</td>
<td>00</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>USE</td>
<td>00</td>
<td>1</td>
<td>C</td>
</tr>
</tbody>
</table>

*Figure 150. ELEMENT/CONTAINS/ELEMENT Relationship-Type*
SYSDEF WITH DATABASE

RELTYPE:  SYSDEF/WITH/DATABASE
ALIASES:  DATABASE/WITH/SYSDEF

ATTRIBUTES:

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>DATA TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMBRES</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ACCESS</td>
<td>00</td>
<td>2</td>
<td>A</td>
</tr>
</tbody>
</table>

Figure 151. SYSDEF/WITH/DATABASE Relationship-Type

SYSDEF WITH PSB

RELTYPE:  SYSDEF/WITH/PSB
ALIASES:  PSB/WITH/SYSDEF

ATTRIBUTES:

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>DATA TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSBLOAD</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>CLASS</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>SCHDTYP</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>RSYSID</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>LSYSID</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>FPPROG</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

Figure 152. SYSDEF/WITH/PSB Relationship-Type
**SYSDEF WITH TRANSACTION**

<table>
<thead>
<tr>
<th>RELTYPE</th>
<th>SYSDEF/WITH/TRANSACT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES</td>
<td>TRANSACT/WITH/SYSDEF</td>
</tr>
</tbody>
</table>

**ATTRIBUTES:**

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>DATA TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRTYNORM</td>
<td>00</td>
<td>2</td>
<td>Z</td>
</tr>
<tr>
<td>PROCNUM</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>PROCSECS</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>LIMTCNT</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>PARLIM</td>
<td>00</td>
<td>5</td>
<td>Z</td>
</tr>
<tr>
<td>PRTYLIMIT</td>
<td>00</td>
<td>2</td>
<td>Z</td>
</tr>
<tr>
<td>MSGCLASS</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>SCHD</td>
<td>00</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>SPAKEEP</td>
<td>00</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>ROUTING</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>DCLWA</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>FPTRAN</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

*Figure 153. SYSDEF/WITH/TRANSACTION Relationship-Type*

**TRANSACTION WITH PSB**

<table>
<thead>
<tr>
<th>RELTYPE</th>
<th>TRANSACT/WITH/PSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES</td>
<td>PSB/WITH/TRANSACTION</td>
</tr>
</tbody>
</table>

**ATTRIBUTES:**

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>DATA TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSTTYPE</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>RSYSID</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>LSYSID</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
</tbody>
</table>

*Figure 154. TRANSACTION/WITH/PSB Relationship-Type*
Extensibility Control Information Categories

ATTRTYPE Category

<table>
<thead>
<tr>
<th>CATEGORY:</th>
<th>ATTRTYPE</th>
<th>SUBJECT CODE: 22</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIBUTES:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEYWORD</td>
<td>INST</td>
<td>SPACE</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>DATE</td>
<td>00</td>
<td>6</td>
</tr>
<tr>
<td>DATATYPE</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>DATAMAX</td>
<td>00</td>
<td>3</td>
</tr>
<tr>
<td>DATAMIN</td>
<td>00</td>
<td>3</td>
</tr>
<tr>
<td>VALIDATE</td>
<td>00</td>
<td>4</td>
</tr>
<tr>
<td>VALR TN</td>
<td>00</td>
<td>8</td>
</tr>
<tr>
<td>VALR NGL</td>
<td>00</td>
<td>24</td>
</tr>
<tr>
<td>VALR NH</td>
<td>00</td>
<td>24</td>
</tr>
<tr>
<td>VALLIST</td>
<td>01</td>
<td>24</td>
</tr>
<tr>
<td>VALLIST</td>
<td>02</td>
<td>24</td>
</tr>
<tr>
<td>VALLIST</td>
<td>03</td>
<td>24</td>
</tr>
<tr>
<td>VALLIST</td>
<td>99</td>
<td>24</td>
</tr>
</tbody>
</table>

Figure 155. ATTRTYPE Category

CATEGORY Category

<table>
<thead>
<tr>
<th>CATEGORY:</th>
<th>CATEGORY</th>
<th>SUBJECT CODE: 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES:</td>
<td>CAT</td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTES:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEYWORD</td>
<td>INST</td>
<td>SPACE</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>DATE</td>
<td>00</td>
<td>6</td>
</tr>
<tr>
<td>SBJCODE</td>
<td>00</td>
<td>5</td>
</tr>
<tr>
<td>NAMETYPE</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>MINNAME</td>
<td>00</td>
<td>2</td>
</tr>
<tr>
<td>MAXNAME</td>
<td>00</td>
<td>2</td>
</tr>
<tr>
<td>VALR TN</td>
<td>00</td>
<td>8</td>
</tr>
<tr>
<td>EXPTXT</td>
<td>00</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 156. CATEGORY Category
### RELTYPE Category

<table>
<thead>
<tr>
<th>CATEGORY: RELTYPE</th>
<th>SUBJECT CODE: 21</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIBUTES:</td>
<td></td>
</tr>
<tr>
<td>KEYWORD</td>
<td>INST</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>DATE</td>
<td>00</td>
</tr>
<tr>
<td>LCATNAME</td>
<td>00</td>
</tr>
<tr>
<td>RCATNAME</td>
<td>00</td>
</tr>
<tr>
<td>FORNAME</td>
<td>00</td>
</tr>
<tr>
<td>INVNAME</td>
<td>00</td>
</tr>
<tr>
<td>SEQOPT</td>
<td>00</td>
</tr>
<tr>
<td>SEQATTR</td>
<td>00</td>
</tr>
<tr>
<td>DIRECTED</td>
<td>00</td>
</tr>
<tr>
<td>EXPTEXT</td>
<td>00</td>
</tr>
</tbody>
</table>

Figure 157. RELTYPE Category

### STRTYPE Category

<table>
<thead>
<tr>
<th>CATEGORY: STRTYPE</th>
<th>SUBJECT CODE: 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIBUTES:</td>
<td></td>
</tr>
<tr>
<td>KEYWORD</td>
<td>INST</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>DATE</td>
<td>00</td>
</tr>
</tbody>
</table>

Figure 158. STRTYPE Category
Extensibility Control Information Relationship-Types

CATEGORY HAS ATTRTYPE

<table>
<thead>
<tr>
<th>RELTYPE:</th>
<th>CATEGORY/HAS/ATTRTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES:</td>
<td>ATTRTYPE/APPLIES_TO/CATEGORY</td>
</tr>
</tbody>
</table>

ATTRIBUTES:

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQ</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>KEYWORD</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>REPEAT</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
</tbody>
</table>

Figure 159. CATEGORY/HAS/ATTRTYPE Relationship-Type

RELTYPE HAS ATTRTYPE

<table>
<thead>
<tr>
<th>RELTYPE:</th>
<th>RELTYPE/HAS/ATTRTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASES:</td>
<td>ATTRTYPE/APPLIES_TO/RELTYPE</td>
</tr>
</tbody>
</table>

ATTRIBUTES:

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQ</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>KEYWORD</td>
<td>00</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>REPEAT</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
</tbody>
</table>

Figure 160. RELTYPE/HAS/ATTRTYPE Relationship-Type
STRTYPE CONTAINS RELTYPE

RELTYPE: STRTYPE/CONTAINS/RELTYPE
ALIASES: RELTYPE/CONTAINED_IN/STRTYPE

ATTRIBUTES:

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>INST</th>
<th>SPACE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQ</td>
<td>00</td>
<td>3</td>
<td>Z</td>
</tr>
<tr>
<td>SUPERCAT</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>FOLLOWUP</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>FOLLOWDN</td>
<td>00</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

Figure 161. STRTYPE/CONTAINS/RELTYPE Relationship-Type
Appendix B.7. Program Access Macro Support

If you are coding a user program in Assembler language, macro statements are available for specification of the PACA and PADAFT fields. The macro names are DBDWDCA and DBDWDFT, and they are supplied as members in the macro library.

Examples of the use of these macros can be found in the Assembler language sample program, DBDWSUPA.

The operands you can specify for the macros are given in Figure 162 and Figure 163 on page 459 in tabular form, with the symbolic names of the fields in the PACA or PADAFT to which they correspond. These fields are defined in Chapter 12, "Writing Programs to Access Dictionary Data" on page 209. The notes that follow each table explain:

- Those operands that do not correspond to a symbolic field name
- Those operands that do not have a meaning exactly the same as the symbolic name
- Those operands that require special values to be specified.

The numbers shown refer to the notes that follow.

<table>
<thead>
<tr>
<th>Name of Operand</th>
<th>Symbolic Name of PACA Field</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE=</td>
<td>PATYPE</td>
<td>1</td>
</tr>
<tr>
<td>EXT=</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>PREFIX=</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>ACCESS=</td>
<td>PAACCESS</td>
<td>5</td>
</tr>
<tr>
<td>DAP=</td>
<td>PADAP</td>
<td></td>
</tr>
<tr>
<td>DAL=</td>
<td>PADAL</td>
<td></td>
</tr>
<tr>
<td>DAOP=</td>
<td>PADAOP</td>
<td>6</td>
</tr>
<tr>
<td>FTP=</td>
<td>PAFTP</td>
<td></td>
</tr>
<tr>
<td>FTL=</td>
<td>PAFTL</td>
<td></td>
</tr>
<tr>
<td>FTU=</td>
<td>PAFTU</td>
<td>7</td>
</tr>
<tr>
<td>CAT=</td>
<td>PACAT</td>
<td>8</td>
</tr>
<tr>
<td>NAME=</td>
<td>PANAME</td>
<td>8</td>
</tr>
<tr>
<td>RKW=</td>
<td>PARKW</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 162 (Part 1 of 2). DBDWDCA Operands for PACA Definition
<table>
<thead>
<tr>
<th>Name of Operand</th>
<th>Symbolic Name of PACA Field</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSEQ</td>
<td>PARSEQ</td>
<td>8</td>
</tr>
<tr>
<td>RCAT=</td>
<td>PARCAT</td>
<td>8</td>
</tr>
<tr>
<td>RNAME=</td>
<td>PARNAME</td>
<td>8</td>
</tr>
<tr>
<td>PLINE=</td>
<td>PAPLINE</td>
<td>—</td>
</tr>
<tr>
<td>NREQ=</td>
<td>PANREQ</td>
<td>—</td>
</tr>
<tr>
<td>INCR=</td>
<td>PAINCR</td>
<td>—</td>
</tr>
<tr>
<td>CCTRL=</td>
<td>PACCTRL</td>
<td>9</td>
</tr>
</tbody>
</table>

Figure 162 (Part 2 of 2). DBDWDCA Operands for PACA Definition

<table>
<thead>
<tr>
<th>Name of Operand</th>
<th>Symbolic Name of PADAFT Field</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE=</td>
<td>—</td>
<td>2</td>
</tr>
<tr>
<td>EXT=</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>PREFIX=</td>
<td>—</td>
<td>4</td>
</tr>
<tr>
<td>PADA=</td>
<td>—</td>
<td>10</td>
</tr>
<tr>
<td>ENTRIES=</td>
<td>PAFTENTS</td>
<td>—</td>
</tr>
<tr>
<td>NAME=</td>
<td>PAFNAME</td>
<td>—</td>
</tr>
<tr>
<td>INST=</td>
<td>PAFINST</td>
<td>—</td>
</tr>
<tr>
<td>DATA=</td>
<td>PAFDATA</td>
<td>10</td>
</tr>
<tr>
<td>LENG=</td>
<td>PAFLENG</td>
<td>—</td>
</tr>
<tr>
<td>FUNC=</td>
<td>PAFFUNC</td>
<td>11</td>
</tr>
</tbody>
</table>

Figure 163. DBDWDFT Operands for PADAFT Definition

Notes to Figure 162 on page 458 and Figure 163

1. For the PACA, "TYPE=" is a required operand. In addition to all the call types that can be specified for the PATYPE field in the PACA, the "TYPE=" operand can have two special values:

   - **DSECT**: The PACA is generated as a DSECT, no space is allocated, and no initial values are set.
   - **TCODES**: The definitions of the valid values for PACA fields are generated.

2. For the PADAFT, the "TYPE=" operand is an operand that controls the building of both header and table entry portions, or may be used to request a DSECT to examine the PADAFT structure. The operand values are:
HEADER
To build initialized PADAFT header fields that define the number and length of entries in addition to the identification fields.

ENTRY
To build a single initialized PADAFT table entry. The use of this option requires you to specify a new prefix for each entry. TYPE=ENTRY is the default.

END
To declare that all entries that were previously built should be incorporated into one PADAFT.

DSECT
To request a pair of PADAFT DSECTs be constructed (a header and one entry). No initialization.

3. Specify EXT=YES to cause the generation of an entry statement. This allows PACA or PADAFT fields, or their initialized values, to be accessed from outside the defining module (with a DSECT in the using module). For PADAFT definitions, the EXT= operand is specified in conjunction with TYPE=HEADER and causes an ENTRY statement to be generated for the table header and each table entry. EXT=NO is the default.

4. The “PREFIX=” operand is used to generate uniquely defined names for the fields in a PACA, or in PADAFT table entries. Specify one or two characters as a prefix. The default is PA The prefix you specify replaces the initial letters PA on generated field names. Symbolic names of generated constants retain the PA prefix.

5. This operand specifies the access rule; the allowable values are:

   FIRST  Corresponding to the symbolic value PAAFIRST
   HERE   Corresponding to the symbolic value PAAHERE
   LAST   Corresponding to the symbolic value PAALAST
   NEXT   Corresponding to the symbolic value PAANEXT
   DOWN   Corresponding to the symbolic value PAADOWN

6. This operand specifies the PADA blanking option; the allowable values are:

   DABLNK  To request PADA fields to be blanked
   DANBL   For no PADA blanking

7. This operand specifies the type of PADAFT usage; the allowable values are:

   FTBE    Corresponding to the symbolic value PAFTBE
   FTPB    Corresponding to the symbolic value PAFTPB
   FTVE    Corresponding to the symbolic value PAFTVE
8. These operands specify the names of categories, subjects, or relationship keywords. If any special characters occur in the name (for example, blanks), the value must be enclosed in single quotation marks.

9. This operand specifies the format control for output requests; the allowable values are:

   SNGL        For single spacing
   DUBL        For double spacing
   PAGE        For a page eject

10. The "PADA=" operand enables you to relate a Program Access Data Area (PADA) to a table entry in a PADAFT definition. Specify the symbolic name of the PADA. The effect of specifying a symbolic name as a value for the "DATA=" operand is to cause the difference between the absolute values of the PADA= and DATA= operands to be calculated and used for the offset required in the PAFDATA field of the PADAFT entry. Alternatively, an absolute value may be specified for DATA=, and this value is placed in the PAFDATA field of the PADAFT entry. PADA= need not be specified in this case.

11. This operand indicates whether a PADAFT table item is to be retrieved or not. Specify values:

    PAFRETR  To request retrieval
    PAFNOP   To skip the item
Appendix B.8. Special Topics for Attribute Retrieval

This appendix provides detailed information about attribute-type retrieval. It includes descriptions of:

- Types of attributes
- Sequence values
- Repeating attributes
- Repeating groups of attributes

Types of Attributes

For each data item returned to your program in the PADA the Dictionary sets a code in the PADAFT (the PAFDTYPE value of each entry) to indicate the item’s format. Regardless of the type code set, however, the data is always returned to the PADA as a string of EBCDIC or DBCS characters and blanks. The format of each data type is described below:

<table>
<thead>
<tr>
<th>Format Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alphabetic data. No DBCS allowed.</td>
</tr>
<tr>
<td>C</td>
<td>Alphameric (character) data. For DBCS users, must be a uniform string if a name, or a mixed string if an attribute.</td>
</tr>
<tr>
<td>N</td>
<td>A name similar to an Assembler language data item name. The first character must be alphabetic; the other characters must be alphameric. For DBCS users, must be a uniform string.</td>
</tr>
<tr>
<td>I</td>
<td>Name of a Dictionary subject (36 bytes long):</td>
</tr>
</tbody>
</table>

This format must be used for subject names your program can specify in several control area (PACA) fields.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Subject Code</th>
<th>User Name</th>
<th>Occurrence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>1 byte</td>
<td>31 bytes</td>
<td>3 bytes</td>
</tr>
</tbody>
</table>

There are no intervening blanks or commas between the fields within the name (although the user name portion is padded on the right with blanks up to the occurrence field); the subject name is not enclosed in parentheses.

For names of Extensibility subjects, the subject-code field (that is, the field that identifies the category) is blank.
For DBCS users, user name portion of subject name must be a uniform string.

D
Name of a Dictionary subject (13 bytes long):

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Subject Code</th>
<th>User Name</th>
<th>Occurrence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>1 byte</td>
<td>8 bytes</td>
<td>3 bytes</td>
</tr>
</tbody>
</table>

Although the Dictionary retrieves some names in this shortened format (for example, the name of a segment's parents in a DATABASE-WITH-SEGMENT relationship), you cannot use the shortened name as input (in PACA fields) for subsequent retrieval requests. You must expand the name to its I-format equivalent (that is, pad the 8-byte user name with 23 blanks on the right).

There are no intervening blanks or commas between the fields within the name (although the user name portion is padded on the right with blanks up to the occurrence field); the subject name is not enclosed in parentheses. For DBCS users, the user name must be a uniform string.

Q
A quoted string. DBCS users may enter a mixed string within EBCDIC single quotation marks. EBCDIC apostrophes within the string are doubled. For example, 'BEE''S KNEES' is a 14-byte Q-format character string.

L
A list. Values take the form of a list of one or more character values separated by EBCDIC commas and enclosed in EBCDIC parentheses. For DBCS users, each of the list entries may be a mixed string. For example, (APPLES,ORANGES,CACTUS) is a 23-byte L-format character string.

S
A special string, specifically, the attributes VALUE, OCCURS, PIC, PLI, and PLINIT of subjects in the ELEMENT category. These are special cases of the "Q" and "C" formats. They may appear in either form depending on how they were entered into the Dictionary and subsequently edited. For DBCS users, this data type accepts mixed strings.

When processing this type of attribute, you might want to check for a beginning apostrophe and, if found, edit the string to remove both the beginning and ending apostrophes.

Z
Numeric data (restricted to the characters 0 through 9). No DBCS allowed. Several attributes are stored in the Dictionary data bases in binary format. The Dictionary converts this data to numeric character format before the data is put into the data area (PADA).

As a special case, the retrieved value for the FREQ attribute in the DATABASE-WITH-SEGMENT relationship (although type-Z) can include an embedded decimal point.
X One byte of hexadecimal data, formatted as a 5-byte string \texttt{X'xx'}, where “xx” is the hexadecimal byte. No DBCS allowed.

G A dummy group header for a repeating group of attributes. See “Repeating Groups of Attributes” on page 465 for a detailed explanation.

### Sequence Values

This section applies only to attribute retrieval for \texttt{ELEMENT-CONTAINS-ELEMENT} and \texttt{SEGMENT-WITH-ELEMENT} relationships.

For standard relationships that are sequenced, the relationship sequence attributes are a combination of the \texttt{START} and \texttt{BITSTART} attributes. When the sequence value for these relationships is returned to your program by the Dictionary (for example, in retrieving a list of subject names related to a subject), the sequence value is in the form \texttt{'0ssss/B'}:

- The initial character is zero.
- \texttt{ssss} is the \texttt{START} attribute value.
- \texttt{/} is a literal ‘/’.
- \texttt{B} is the \texttt{BITSTART} value (or 0 if none is defined).

The sequence value must be specified in this format as input (in the PACA field named \texttt{PARSEQ}) if a \texttt{BITSTART} and \texttt{START} value is to be specified to retrieve data about a relationship between two specified subjects, or if \texttt{BITSTART} and \texttt{START} are to be specified as positioning data in the \texttt{PAPSEQ} field of the PACA.

### Repeating Attributes

To find out whether an attribute falls into this category, review “Appendix B.6. \texttt{PADAFT} Entries for Dictionary Categories” on page 437 and examine the master \texttt{PADAFT} containing the attribute.

Many Dictionary attributes are “repeating attributes” (that is, they have more than one value within a given subject or relationship). For example, the \texttt{INDCNAME} operand (in the \texttt{PCB-WITH-Segment} relationship) may repeat up to 32 times. That is, there can be up to 32 \texttt{INDCNAME} values associated with a specific relationship instance. In a \texttt{FORMAT} report, and in the master \texttt{PADAFT} for retrieving attributes of this type of relationship, the name \texttt{INDCNAME} appears 32 times with distinguishing instance numbers. You can retrieve all 32 \texttt{INDCNAME} values with one retrieval request by listing 32 entries in the \texttt{PADAFT} for \texttt{INDCNAME}, each entry having a unique instance number (\texttt{PAFINST} value) from 1 to 32.
Repeating Groups of Attributes

To find out whether an attribute falls into this category, review "Appendix B.6. PADAFT Entries for Dictionary Categories" on page 437 and examine the master PADAFT containing the attribute.

Subjects in some standard categories and relationships can have a group of attributes that repeat as a group. Repeating attribute groups do not occur in Extensibility subjects or relationships. There are two types of repeating attribute groups:

- The first type has all the repetitions of the attribute names present in the master PADAFT. For example, SRCSNAME, SRCDNAME, and SRCKDOPT, in the DATABASE-WITH-SEGMENT relationship, is a group having two instances. Each of these names appears twice in the master PADAFT: once with PAFINST=1 (the first instance), and once with PAFINST=2. Both instances of the group may be retrieved, if desired, with a single retrieval request.

- The second type of attribute group has a "group header" in the form of a dummy attribute with type "G," which controls the repetition of the group of attributes that follow it. There are three type G attribute groups:
  
  - DATASET, in the DATABASE category (the 14 attributes that describe the physical characteristics of a data set)
  
  - PCHILD, in a DATABASE-WITH-SEGMENT relationship
  
  - LCHILD, in a DATABASE-WITH-SEGMENT relationship

An instance number of "#01" appears in the master PADAFT and in FORMAT reports. Your program may alter this number on the type "G" attribute to identify the specific instance to be retrieved. Only one instance of the group can be retrieved with each retrieval request. Usually, you would retrieve the first instance along with other attributes on the first retrieval request. You would then set the flags in the PADAFT to indicate that all attributes except the type "G" attribute is to be ignored. Next, your program loops to retrieve each instance of the attribute group:

1. Increase the type "G" attribute's sequence number.

2. Issue the retrieval request.

3. Continue the loop until the return code indicates end-of-data (PARC=PACEOD or END-OF-DATA).
The following rules apply to retrieving a type G attribute group:

- Group members must follow the group header in the PADAFT. Be careful with this if your program builds the PADAFT instead of using a master PADAFT.

- Instance number fields of the group members must be zero or must match those of the group header. (In a master PADAFT, the group instance is "@01", the member instances are zero, and all are flagged for retrieval.)

- The header must be flagged for retrieval (not "skip") in order for any of the group elements to be retrieved.

- Each individual member to be retrieved must also be flagged. In this way, your program can retrieve only selected attributes within the group.

CAUTION: The Dictionary does not explicitly check for these conditions. Failure to follow these rules can result in an inability to retrieve some or all of the data, or may cause retrieval of data for the wrong instance of one or more group members. Also, it is possible (but not advised) to retrieve instance number one of any group element without specifying the group header.
Appendix C. Format Specifications for EXPORT Command Output Streams

The output from an EXPORT command consists of update commands and batch form input. This command stream is produced as punched output.

To assist in the readability and editing of the output command stream, the format of the records produced follows a set of conventions for the placement of fields in each record. This appendix gives the syntax conventions for each type of record.

Command Formats

The ADD, CHANGE_IN, DELETE_DATA, ADD_RELATIONSHIP, CHANGE_RELATIONSHIP_DATA, and INSTALL commands are arranged in a uniform format so that individual commands may be easily identified.

- The first field of the first record for each command contains a Dictionary command verb (or standard abbreviation). Unless attribute data is continued from a previous record, this field is left blank on continuation records for the command.

- Each command operand is separated by at least one blank space.

- Commands to be continued over several physical records contain the standard Dictionary continuation indicator ("+") on each record except the last.

The format of each command is described in the sections that follow.
ADD, CHANGE__IN, and DELETE__DATA Command Formats

The record formats for the ADD, CHANGE__IN, and DELETE__DATA commands are:

First record:

<table>
<thead>
<tr>
<th>cc</th>
<th>1</th>
<th>10</th>
<th>19</th>
<th>61</th>
<th>79</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADD</td>
<td>category</td>
<td>subject name</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>CI</td>
<td>category</td>
<td>subject name</td>
<td>TO</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>category</td>
<td>subject name</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Second and succeeding records:

<table>
<thead>
<tr>
<th>cc</th>
<th>1</th>
<th>10</th>
<th></th>
<th>79</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>keyword1=data1 keyword2=data2 ... keywordn-1=data1n-1</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>keyword1=data1 keyword2=data2 ... keywordn-1=data1n-1</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATA=((segmentname1),(segmentname2,n1,n2), ...</td>
<td>+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Last record:

<table>
<thead>
<tr>
<th>cc</th>
<th>1</th>
<th>10</th>
<th></th>
<th>79</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>keywordn=data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>keywordn=data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(segmentnamen,n1)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subject names are generated in the form:

(status,code,username,occurrence)

Attribute data is placed on the second record, beginning in column 10. Attribute data consists of one or more keyword-value pairs (for example, BYTES=35). If a keyword-value pair must be split between two (or more) records, a continuation indicator immediately follows the partial keyword-value pair and the remainder of the pair begins in column 1 of the next record. The “keyword” portion of “keyword=data” value is not split between two records. When this is about to occur, the current record is completed with at least one blank space and a
continuation indicator. The keyword-value pair is placed on the next record, beginning in column 10.

A single space is left between keyword-value pairs on the same record. The absence of a continuation indicator denotes the end of the command.

For DELETE__DATA, a single keyword-value pair has one or more values enclosed in parentheses. These are the segment names of text data segments. Each segment name can be followed by one or two line identification numbers. A single value of * means that all lines are deleted. A single value means that one line is deleted. If a segmentname is split, it continues on the next record, starting at column 10.

Note: A Dictionary syntactical rule is that column 80 may not be used.
ADD_RELATIONSHIP and CHANGE_RELATIONSHIP_DATA Command Formats

The record formats for the ADD_RELATIONSHIP and CHANGE_RELATIONSHIP_DATA commands are:

First record:

<table>
<thead>
<tr>
<th>cc</th>
<th>1</th>
<th>10</th>
<th>19</th>
<th>61</th>
<th>79</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AR</td>
<td>category1</td>
<td>subject-name1</td>
<td>relationship-keyword</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>CRD</td>
<td>category1</td>
<td>subject-name1</td>
<td>relationship-keyword</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

Second record:

<table>
<thead>
<tr>
<th>cc</th>
<th>1</th>
<th>10</th>
<th>19</th>
<th>61</th>
<th>79</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>category2</td>
<td>subject-name2</td>
<td>sequence-attribute</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>category2</td>
<td>subject-name2</td>
<td>TO</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

Third and succeeding records:

<table>
<thead>
<tr>
<th>cc</th>
<th>1</th>
<th>10</th>
<th>28</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>keyword1=data1</td>
<td>keyword2=data2</td>
<td>sequence-attribute</td>
<td>keyword1=data1</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>keywordm-1=data1</td>
<td>keywordm-1=data1</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

Last record:

<table>
<thead>
<tr>
<th>cc</th>
<th>1</th>
<th>10</th>
<th></th>
<th></th>
<th>79</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>keywordm=dataam</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

subject names are generated in the form:

(status, code, username, occurrence)

Relationship data is placed on the third record, beginning in column 10. Relationship data consists of one or more keyword-value pairs (for example, START=1). If a relationship keyword-value pair must be split between two (or more) records, a continuation indicator immediately follows the partial keyword-value pair and the remainder of the pair begins in column 1 of the next
record. The "keyword" portion of a "keyword=data" value is not to be split between two records. When this is about to occur, the current record is completed with at least one blank space and a continuation indicator. The keyword-value pair is placed on the next record, beginning in column 10.

A single space is left between keyword-value pairs on the same record. The absence of a continuation indicator denotes the end of the command.

Note: Column 80 may not be used.

**INSTALL Command Format**

INSTALL commands are generated under the control of the INSTALL option on the EXPORT command. The record format for INSTALL commands is shown below. The UPDATE option is set to "YES" or "NO" depending upon the status of the extensibility control information category subject being exported. If it is in "P" status, UPDATE=YES is generated on the INSTALL command; otherwise, UPDATE=NO is generated.

Note: If the extensibility control information category subject being exported is installed (in "*" status), it will be exported in "P" status in the absence of the NEWSTAT operand.

```
First record:

cc  1   10  19   61   79
|   |    |    |    |
INSTAL defn-cat subject-name UPDATE=YES
```

The INSTALL command format is summarized as follows:

```
First record:

columns contents
------  ---------
1-8     'INSTALL'
9       blank
10-17   defn-cat: 'CATEGORY', 'RELTYPE',
       or 'STRTYPE'
18      blank
19-57   subject-name
58      blank
59-68   'UPDATE=YES' or 'UPDATE=NO'
       depending upon status of defn-cat
69-79   blank
```
Subject names are generated in the form:

\[(\text{status}, \text{,username}, 0)\]

Note: Column 80 may not be used.

Batch Forms Formats

The DETAIL operand of the EXPORT command controls the generation of batch forms images, which contain the Description, User Data, or PLIDATA for the subject being exported. These images conform to the batch forms format for text data. Refer to the specifications for the Text-Data Form in Chapter 5 of OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference.

A separate, complete batch form is generated for each type of text data being exported. A complete batch form consists of one Header, one or more Input Text records, and one End record.

The maximum number of complete batch forms that is generated for an exported subject is seven: one for Description, one for PLIDATA, and five for User Data. (Note: PLIDATA is valid only for SEGMENT and ELEMENT subjects.)

The batch forms images immediately follow the update commands for the subject with which they are associated. The “Subject category” and “Subject name” fields on the Header for each batch form is identical to the category and subject name on the associated ADD command.

Header columns 5 through 7 identify the type of text data being added. ’DSC’, ’PLI’, or ’USR’ is specified when the form contains Description, PLIDATA, or User Data, respectively. If ’PLI’ is specified, column 4 of each Input Text record contains ’C’, ’N’, or ’X’ indicating the type of PLIDATA being added. If ’USR’ is specified, Header column 9 contains the User Data number (1-5).

The “MODE” field on the Header is always set to “O” (old) because the batch form will always follows an ADD command for the subject.

The “Originator ID” and “Defaults” fields on the Header and the “Report” field on the End record are not used.

When exporting a subject that has associated user data, a DBCS user may encounter a special situation. When output in batch forms format is generated by the EXPORT command, the User Data text may be split onto two lines (when it is too long to fit on one line). If the data is DBCS data and the split occurs with a blank inserted before the required continuation character, an unintended space may occur between the two DBCS strings. DBCS users should carefully inspect the output in batch forms format before it is executed and correct it, if necessary. A possible solution is to insert a space at the beginning of the line to eliminate the space at the end of the line.
Date and Time-Stamp Record Format

The format below is used for the generated date and time-stamp record. This record is the first record of the EXPORT output file.

```
cc 31 45 79
1 |
**** EXPORT OUTPUT GENERATED mm/dd/yy AT hh:mm:ss
```
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI</td>
<td>American National Standards Institute</td>
</tr>
<tr>
<td>CICS/OS/VS</td>
<td>Customer Information Control System/Operating System/Virtual Storage</td>
</tr>
<tr>
<td>DBCS</td>
<td>Double Byte Character Set</td>
</tr>
<tr>
<td>DB/DC</td>
<td>Data Base/Data Communication</td>
</tr>
<tr>
<td>DBD</td>
<td>Data Base Description</td>
</tr>
<tr>
<td>DDT</td>
<td>Data Definition Table</td>
</tr>
<tr>
<td>DEEB</td>
<td>Data Entry Data Base</td>
</tr>
<tr>
<td>DL/I</td>
<td>Data Language/I</td>
</tr>
<tr>
<td>DSG</td>
<td>Data Set Group</td>
</tr>
<tr>
<td>GIS/VS</td>
<td>Generalized Information System/Virtual Storage</td>
</tr>
<tr>
<td>GSAM</td>
<td>Generalized Sequential Access Method</td>
</tr>
<tr>
<td>IMS/VS</td>
<td>Information Management System/Virtual Storage</td>
</tr>
<tr>
<td>ISR</td>
<td>Indirect Subject Reference</td>
</tr>
<tr>
<td>MSDB</td>
<td>Main Storage Data Base</td>
</tr>
<tr>
<td>PCB</td>
<td>Program Communication Block</td>
</tr>
<tr>
<td>PIP</td>
<td>Primary Index Pointer</td>
</tr>
<tr>
<td>PSB</td>
<td>Program Specification Block</td>
</tr>
<tr>
<td>SIP</td>
<td>Secondary Index Pointer</td>
</tr>
<tr>
<td>SYSDEF</td>
<td>IMS/VS system definition</td>
</tr>
<tr>
<td>VSAM</td>
<td>Virtual Storage Access Method</td>
</tr>
</tbody>
</table>
Glossary

This glossary defines OS/VS DB/DC Data Dictionary terms used in this book. As a convenience, a number of DL/I terms helpful in understanding the Dictionary are included; for further explanation of these and other DL/I concepts, see the General Information Manual for IMS/VS. If you do not find the term you are looking for here, refer to the index or to the IBM Data Processing Glossary, GC20-1699. Abbreviations used in this book are defined in the list immediately following this glossary.

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the American National Standard Vocabulary for Information Processing (Copyright 1970 by American National Standards Institute, Incorporated), which was prepared by Subcommittee X3K5 on Terminology and Glossary of the American National Standards Committee X3.

ANSI definitions are preceded by an asterisk (*). The symbol “(SC1)” at the beginning of a definition indicates that it has been agreed upon at meetings of the International Organization for Standardization Technical Committee 97/Subcommittee 1, and has also been approved by ANSI and included in the American National Standard Vocabulary for Information Processing.

**application system.** In DL/I, a combination of data bases and application programs that satisfies a complete area of information-processing requirements, for example, a payroll system or a Dictionary system.

**attribute.** In the OS/VS DB/DC Data Dictionary, an attribute is a property or characteristic of a subject or a relationship (for example, the data length or type). In the broader context of programming languages, an attribute is a specification or qualifier that is part of a data definition.

**ATTRTYPE.** A Dictionary category that is used to store occurrences of Dictionary attribute-type definitions.

**byte.** A sequence of eight adjacent binary digits that are operated upon as a unit and that constitute the smallest addressable unit in the system. One byte is used to store a single EBCDIC character. Two bytes are used to store a single DBCS character.

**character.** An external representation of any symbol, letter, or digit that is used to represent data. The internal representation of a single character may not necessarily be restricted to a single byte.

**CLIST.** See command list

**command list.** A list of commands.

- In the OS/VS DB/DC Data Dictionary, an unordered list of Dictionary commands stored in the Dictionary so that they can be invoked as a group.
- In the TSO environment, a data set or a member of a partitioned data set containing TSO commands to be performed sequentially by the EXEC command.

Abbreviated CLIST.

**control statement.** A program statement that identifies the function to be performed when processing the output definition. A control statement consists of a control verb identifying the function to be performed and optional operands that control various aspects of processing of that function.

**control verb.** Part of a control statement that identifies the function to be performed for each control statement. Each control verb begins with a period (.).

**current destination.** Destination of RFG output that has been specified by the .DEST control statement or is the default destination.

**current subject.** The Dictionary subject that is currently being processed according to the program statements in the output definition.

**data base.** (1) A collection of data fundamental to an enterprise. (2) A nonredundant collection of interrelated data items processable by one or more applications. (3) In DL/I, a collection of interrelated data items arranged in a hierarchic or tree structure that can be processed by one or more applications. (See also logical data base, physical data base.)

**data base administrator.** A person (or persons) with responsibility for the operational data in a data base.
system, and particularly for defining the rules by which
data is accessed and stored.

**data base description.** In DL/I, a set of statements
describing the hierarchic structure, DL/I organization,
device type, segment length, content, and retrieval
sequences of a DL/I data base. Abbreviated DBD.

**data definition.** A program statement that describes the
features of, specifies relationships of, or establishes
context of data.

**data dictionary.** A central collection of definitions of the
data resources of an enterprise and their use in application
Systems.

**Data Entry Data Base.** A data base used to provide a high
level of availability for, and efficient access to, large
volumes of detailed data. Each DE DB can be partitioned,
or divided into multiple areas, with each area containing a
different collection of data base records, grouped together
by a randomizing algorithm. The DE DB area concept
makes it possible for DE DBs to provide a high level of
data availability, and to support large data bases.
Abbreviated DE DB.

**data hierarchy.** A data structure consisting of sets and
subsets such that every subset of a set is of lower rank
than the data of the set.

**Data Language/I.** The data management facility in
IMS/VS that allows the user to specify interrelationships
between the user application and the host system: to
define data bases, relate them to the application, and load
and reorganize them. Abbreviated DL/I.

**data set.** The major unit of physical storage of data in a
system without a data base structure.

**data structure, programming language.** (See programming
language data structure.)

**DBCS.** See Double Byte Character Set.

**DE DB.** See Data Entry Data Base.

**default destination.** Destination of RFG output that has
been established through one of the following:

- DEST execution-time option
- DEST attribute
- Default DEST (specified in the defaults module)

**definition.** In the OS/VS DB/DC Data Dictionary, a
collection of entries in one or more of the dictionary
subject categories that name and define a data resource.

**dependent segment.** In DL/I, a segment that relies on a
higher level segment for its full meaning; that is, all
segments other than the root segment in a hierarchic data
base.

**dialog.** Any application designed to be run under the
ISPF dialog manager. A dialog is composed of program
and data elements, which together create an orderly
interaction between the computer and the user of the
application. This interaction consists of a series of
interrelated inquiries and responses, similar to a
conversation between two people.

**display field.** On display stations, an area in the display
buffer, and on the screen, that contains a set of characters
manipulated or operated on as a unit.

**display form.** In the OS/VS DB/DC Data Dictionary, one
of a set of standard display images presented by the
interactive display forms facility, each with a fixed format
of labeled fields or columns in which users enter
information to be stored in the dictionary or review
information already stored.

**DL/I.** See Data Language/I.

**Double Byte Character Set.** A character set whose
representation uses 2 bytes per character. Abbreviated
DBCS.

**EBCDIC.** See Extended Binary Coded Decimal
Interchange Code.

**element.** In the OS/VS DB/DC Data Dictionary, the
smallest unit of data for which a separate subject category
is provided. Synonymous with: field. Subelements are
included in element definitions. Elements are grouped to
form segments; segments are grouped in hierarchic levels
to form data bases.

**entity.** (See subject.)

**entry.** In the OS/VS DB/DC Data Dictionary,
information entered through commands or the interactive
display forms facility to name and define dictionary
subjects and define the relationships between them.

**execution-time options.** Optional operands of the PARM
string operand of the Dictionary EXECUTE command
that enable the user to tailor each specific execution of an
output definition.

**expedited message handling.** A process whereby Fast Path
messages bypass IMS/VS message queue processing,
reducing the time that a message must wait to be
processed. In addition, Fast Path application programs
remain in virtual storage and wait for the next message,
reducing the time required by the application program to
process the next message.
Extended Binary Coded Decimal Interchange Code. A single byte character set. Abbreviated EBCDIC.

external file. A sequential data set or member of a partitioned data set created by the RFG. It can be created in addition to or in lieu of files created via the Dictionary destination specification terminal, printer, or punch.

Fast Path facility. In IMS/VS, a facility for applications that require good response characteristics and that may have large transaction volumes. Terminals have rapid access to main storage data bases to the field level and to direct access data entry data bases. Processing is grouped for load-balancing and is synchronized for data integrity and recovery. (See also Data Entry Data Base, Main Storage Data Base.)

field. *(1) In a record, a specified area used for a particular category of data, for example, a group of card columns used to represent a wage rate or a set of bit locations in a computer word used to express the address of the operand. (2) In the OS/VS DB/DC Data Dictionary, field and element are synonymous.

field, display. (See display field.)

field level sensitivity. The ability of an application program to access a subset of a segment's fields, to be denied access to the rest of the fields, and to specify if a field can be updated.

file. A sequential data set or member of a partitioned data set created by the RFG. It can be created in addition to or in lieu of files created via the Dictionary destination specification terminal, printer, or punch.

following relationships. A Dictionary process using a relationship as a path from one subject to another (related) subject. If a relationship is followed, the related subject as well as the relationship itself is processed in some way. If a relationship is not followed, only the relationship is processed.

forms entry. In the OS/VS DB/DC Data Dictionary, creation of entries in the dictionary by filling in blanks on display forms represented by the interactive display form facility.

group. A grouping of logically related elements and/or groups (also called grouped data items) such as a structured definition of a logical record in a COBOL program.

help panel. A type of panel used for reference or instructional purposes. Generally, however, these panels are presented to a user at specific points in a dialog upon request for more information about what to do at that step. This term also refers to a complete set of panels (a tutorial) specifically used for instructional purposes.

highlight. In computer graphics, the higher of two contrasting light levels, used to emphasize a portion of a display image.

*job. A set of data that completely defines a unit of work for a computer. A job usually includes all necessary computer programs, linkages, files, and instructions to the operating system.

key form. In the OS/VS DB/DC Data Dictionary, a display form that may be selected from the subject categories offered on the header form of the interactive display forms facility.

*level. The degree of subordination of an item in a hierarchic arrangement.

logical data base. In DL/I, a data base that interrelates segments of one or more physical data bases in hierarchic order corresponding to their use in an application.

logical segment. A SEGMENT subject that describes a segment of a logical data base.

loop. A group of program statements processed repetitively. Loops are initiated by:

- The .LOOP control statement that is used to control operations performed while interpreting the program statements in the output definition.
- Most data retrieval control statements.

lowlight. In computer graphics, the lower of two contrasting light levels, used for the less emphasized portion of the display image.

Main Storage Data Base. A data base used to store and provide access to an installation's most frequently used data. The data in an MSDB is stored in segments. Each segment can be available to all terminals, or assigned to a specific terminal. To provide fast access and allow frequent update to this data, MSDBs reside in virtual storage during execution. MSDBs may not be shared. Abbreviated MSDB.

mixed string. A character string that may contain both single-byte characters (EBCDIC) and double-byte characters (DBCS). The DBCS portion of the string must be enclosed in shift-out and shift-in characters.

*module. A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from, an assembler, linkage editor, or executive routine.

MSDB. See Main Storage Data Base.

name rules. The data type validation rules applied to uniform strings to verify the validity of the string.
**national character.** Deprecated term for the characters #, @, and $. Some terminals and printers support national symbols other than #, @, and $. When entering data in the Dictionary, the national characters must be specified according to the correct hexadecimal representation: X'7B' for #, X'7C' for @, and X'5B' for $.

**no-role segment.** A SEGMENT subject that describes (a) a segment not currently related to a data base, or (b) a segment that has not been assigned a role in a data base definition.

**output definition.** Describes a report or output file to be generated by an execution of RFG. It consists of two parts: (1) information about the RFG execution which is stored as attributes of an RFG subject, and (2) program statements that describe the data to be processed and the format of output to be generated. Program statements are stored as User Data text of the same RFG subject.

**output description code.** A two EBCDIC character code in the first two nonblank positions of an output statement that specifies the format, purpose, or destination of the statement.

**output routing.** A Dictionary facility for directing individual user output to a data set or an IMS/VS logical terminal. The output data can be preceded by other output records to direct the use of the data, for example, JCL to invoke a procedure or input to an MVS internal reader.

**output statement.** A program statement that identifies the format image of an output line or output record in the output definition. It is distinguished from the control statement by the presence of an output description code instead of a control verb.

**physical data base.** In DL/I, a data base in which segments are physically stored in an order corresponding to the hierarchic order of their use in an application.

**physical segment.** A SEGMENT subject that describes a segment of a physical data base.

**primary-index pointer (PIP) segment.** A SEGMENT subject that describes a segment of a primary index data base.

**program communication block.** In DL/I, a collection of information relating to a program and used by DL/I to determine which data base or terminal is referred to and to return the results of the input/output operation to the program. Abbreviated PCB.

**program specification block.** In DL/I, a set of statements naming the required data bases, segments to be accessed, and data base modification options for a given program. It contains a program communication block (PCB) for each data base named, in the sequence used by the program. Abbreviated PSB.

**program statement.** Controls the flow and format of the desired output. Program statements are stored as lines of text in one set of User Data for each subject in the RFG category. Program statements consist of either control statements or output statements, or both.

**programming language data structure.** In the OS/VS DB/DC Data Dictionary, a sequence of input or output data that defines the logical structure of the data with respect to a particular programming language; that is, a PL/I declare statement, a COBOL data description, or an assembler language data definition.

**real segment.** A SEGMENT subject that describes a segment and is related to a view segment with the HAS VIEW or VIEW OF keyword. When generating a PSB that contains a PCB related to a real segment, only segment-level sensitivity is assumed for the PCB.

**record.** In the OS/VS DB/DC Data Dictionary, a collection of related data fields treated as a unit corresponding to a DL/I segment; that is, as defined in the segment subject category.

**record description.** In COBOL, the total set of data description entities associated with a particular logical record.

**relationship data.** In the OS/VS DB/DC Data Dictionary, data defining the logical connection between two subjects defined in the dictionary.

**relationship-type.** A form of logical connection that can exist between two subjects. A Dictionary relationship-type possesses attributes (for example, an offset) and might include some descriptive text that describes the relationship-type. Relationships are identified by two names (for example, A and B) separated by a verb that implies direction:

- Forward: the primary meaning of a relationship (for example, A OWNS B, A AUDITS B)
- Inverse: the converse of a primary relationship that exists between two subjects (for example, B OWNED _ BY A, B AUDITED _ BY A)

**RELTYPe.** A Dictionary category used to store occurrences of Dictionary relationship-type definitions.

**report.** In the OS/VS DB/DC Data Dictionary, information presented at a user's request, in the form of a printed listing or displayed image, summarizing some aspect of the definitions in the dictionary.
The name of the Program Access facility application program that processes output definitions.

**retrieve identifier variable.** A variable on a control statement used to match the start and end of data retrieval loops and blocks.

**RFG.** See Report and File Generator.

**RFG category.** The term used throughout the documentation to denote the extensibility category that contains output definitions. This name can be changed by the installation if desired.

**RFG defaults.** Values specified in the defaults module for the RFG. The default values include defaults for some RFG attributes and defaults for controlling RFG processing.

**root segment.** In DL/I, the unique, highest level segment in a hierarchic data base, on which each subordinate segment depends for its complete meaning.

**scan.** To examine sequentially, part by part.

**secondary-index pointer (SIP) segment.** A SEGMENT subject that describes a segment of a secondary index data base.

**segment.** In DL/I, the basic unit of data stored and retrieved by an application program. A segment may be defined as either fixed or variable in length and may contain one or more logically related data fields.

**shift-in character.** One single-byte character of a pair of characters, used to enclose a DBCS string. A shift-in character is the trailing delimiter of a DBCS string. The representation of a shift-in character used in the Dictionary publications is a >. The hexadecimal representation of a shift-in character is X'0F1'. Abbreviated si.

**shift-out character.** One single-byte character of a pair of characters, used to enclose a DBCS string. A shift-out character is the leading delimiter of a DBCS string. The representation of a shift-out character used in the Dictionary publications is a <. The hexadecimal representation of a shift-out character is X'0E1'. Abbreviated so.

**si.** See shift-in character.

**so.** See shift-out character.

**special characters.** (1) * (ISO) A graphic character in a character set that is not a letter, not a digit, and not a space character. (2) In COBOL, a character that is neither numeric nor alphabetic. Special characters in COBOL include the space ( ), the period (.), as well as the following: + - * / = $ ; " ) ( 

**stack subject.** A Dictionary subject whose User Data contains an ordered list of category subject-name pairs to be substituted in a given Dictionary command that is repeatedly executed.

**STRTYPE.** A Dictionary category used to store occurrences of Dictionary structure-type definitions.

**structure-type.** A network of relationships defined as a list of categories with connecting relationship-types that represents a distinct Dictionary structure and describes the manner in which those types of relationships are to be processed.

**structure reports.** Dictionary reports that extract occurrences of subjects and instances of their relationships to other subjects according to selection criteria defined by a STRTYPE subject or by a path following a sequence of categories and relationship keywords.

**subject.** In the OS/VS DB/DC Data Dictionary, a unit of data or a system component that is defined in a dictionary entry; for example, a data base, segment, or program.

**subject categories.** In the OS/VS DB/DC Data Dictionary, the classes of data resource subjects in which user entries of subject names and attributes are anticipated; for example, data base, segment, element. In addition to the IBM-supplied subject categories, the data base administrator can define up to 200 customized subject categories and relationship-types.

**subject name.** In the OS/VS DB/DC Data Dictionary, the complete four-part name by which subjects are identified in the dictionary (user name plus three subject name qualifiers).

**system.** A combination of jobs that satisfies a complete area of information processing requirements, such as an inventory system. (See also application system.)

**text rules.** The data type validation rules applied to mixed strings to verify the validity of the string.

**transaction.** (1) In batch or remote batch entry, a job or job step. (2) In systems with time sharing, an exchange between a terminal and another device that accomplishes a particular action or result; for example, the entry of a customer's deposit and the updating of the customer's balance.

**tree structure.** (See data hierarchy.)

**uniform string.** A character string that contains either double-byte characters (DBCS) or single-byte characters (EBCDIC), but not both. A DBCS string must be enclosed in shift-out and shift-in characters.
**user data.** In the OS/VS DB/DC Data Dictionary, definitions of some data resource subjects not covered by the preestablished dictionary subject categories, stored in special user data segments reserved for that purpose.

**user name.** In the OS/VS DB/DC Data Dictionary, the third part of the four-part subject name; the name assigned to the subject by the user.

**variable.** A value used for saving data, counting, loop control, and from which values may be substituted into program statements. Variables are supplied with the RFG, and may also be defined by the developer of the output definition.

**view segment.** A SEGMENT subject that describes a subset, or "view," of the fields in a real segment, and is related to the real segment. A view segment can consist of:

- A subset of a segment’s fields
- All of a segment’s fields, arranged in a different order
- A subset (or all) of a segment’s fields with different application accessing authorization (that is, either read-only or update)

**window.** In computer graphics, a bounded area within a display image that contains a subset of the displayable data.
Index

Special Characters

*S Y O, R P T command access control check  19
%INCLUDE statement       333

access authorization checking for user programs  214
access control
   in DBCS environment  6
access profiles  13, 31-39
   authorizing  45
   definition with display forms  39-45
   establishing  31
   examples  38-39
   format  35-37
   keywords  31, 35
   keywords for  32
   model profile  43
   syntax rules  35
   verifying  45
   why used  14
accessing Dictionary data  209
   alias retrieval  243
   examples of  230
   output requests  271
   overview of  209
   relationship attribute data retrieval  235
   relationship-type retrieval  262, 266
   requesting  214
   retrieving data  215
   retrieving relationships for a subject  250
   sending output messages  216
   subject attribute data retrieval  232
   subject name and alias retrieval  246
   text data retrieval  238
   with user programs  70
ADD command
   example of for STRTYPE subject  208
   for ATTTRTYPE subjects  79, 81
   for CATEGORY subjects  91
   for DDUSER subjects  30-31
   for RELTYPE subjects  103
   for STRTYPE subjects  185
   syntax for  30
ADD_RELATIONSHIP command
   example of for STRTYPE  191
   example of for STRTYPE subject  208
   for ATTTRTYPE relationships  81
   for CATEGORY relationships  91, 92
   for RELTYPE relationships  104-105
   for STRTYPE definition  185
   for STRTYPE relationships  186
   for STRTYPE-RELTYPE relationships  186
   aids, programming for Program Access  73
   alias
      added with AR command  91
      added with AR command for STRTYPE  191
      changing for attribute-type definition  81
      display form  168
      display form restriction after installed  171, 173
      example of use  91
      for default structure-type  203
      not deleted by DELETE command  104
      relationship  52
   retrieving subject aliases for Program Access  243
   application system (definition)  477
   area for retrieved data  218
   Assembler
      accessing Dictionary data  70
      coding program to access Dictionary data  283
      COPY statement for  335
      example for Language Preprocessor  344
      ICTL statement for  335
      inclusion statements for  296
      parameter area for user exit routine  320
      validation routine entry point  178
      validation routine for  59, 176
   assigning the sequence of attributes  80
   attribute data
      changing for relationship-type  140
      changing for subject categories  137
      changing validation specifications for  144
      retrieving  71
      stored in segments  128-129
   attribute retrieval
      for a relationship  235
      for a subject  232
      overview of  224
   attribute-type
      as a general property  54
      components of definition  55
      defining  75
      relating to CATEGORY subjects  92
      relating to RELTYPE subjects  104
   attribute-type category  75
      definition operands  76
      for installation-defined entities  55
      operands for defining  76
   attribute-type definitions
      defining with commands  81
      definition operands  79
      example of report  82
      operands for  76
      procedure for changing  82

Index  483
relating to CATEGORY subjects 79
relating to relationship-type 104
relating to RELTYPE subjects 79
reporting on 82
subject-specific report 82
validation for 117
attribute-types as sequencing attributes 105
required criteria 105
attributes
of a subject or relationship 54
repeating 54, 80
values stored in segments 128-129
ATTRTYPE category
defining 75
defining with commands 81
element of report 82
for installation-defined entities 55
operands for defining 76
procedure for changing 82
relating to category 79
relating to relationship-type 79
reporting on 82
subject names 75
ATTRTYPE definition
validation for 117
ATTRTYPE form
description of 173
element of 173
authorization 14
for Dictionary users 45
modifying DDUSER subjects 46

B

batch forms
use for DDUSER subjects 47
blank lines, not allowed in access profile 35
build and execute option (of the PADAFT) 219

C

CALL statement
for a COBOL program 275
for a PL/I program 280
for an Assembler language program 285
to access Dictionary data 214
to communicate with the Dictionary 210
to send output messages 216
CATEGORY category
for installation-defined entities 55
category definition
adding and updating 91
alias name assigned 96
allowed characters in name 89
changing operands 135
copying 91
definition operands 88-90
deleting 92
explain text 90
procedure for changing 93
relating attribute-types 92
reporting development 95-97
restrictions after installed 57
subject-specific reports 82
CATEGORY form for extensibility subject
description of 168
element of 168
category name and number pairs 15
category number checking 36, 46
in access profiles 34-37
in Sign-On Header form 44
on USER DATA form 41, 43
summary of values 15
verifying 46
category records 35
syntax for 36
syntax rules 35
category records, for access profiles 35-37
format 35
CATEGORY report
element of 96
category subject
adding 91
defining 87
defining subject name 87
definition reports 95
deleting 92
element of GUIDE report 124
element of subject-specific report 96
for installation-defined entities 55
FORMAT reports 130
GUIDE report for 121
procedure for changing 93
relating attribute-types to 92
reporting on 95
specifying operands for 88
updating 91
using commands to define 91
validation for 115
CATEGORY-ATTRIBUTES form
example of 169
CATEGORY-RELTYPES form
description of 170
element of 170
category's relationship-types, retrieval of 262, 266
CHANGE IN command
example for CATEGORY subject 91
example for STRTYPE subject 185
for attribute-type definitions 81
for RELTYPE subjects 103
used for access control maintenance 46
CHANGE NAME command
for attribute-type definition 81
index 485

- assigning commands 27
- Dictionary-supplied 27
- syntax for records 37
- command lists 325
- concatenating status codes 325
- for Language Preprocessor 298
- commands
  - category access requirements 23
  - communicating
    - with the Dictionary 210
  - communicating with
    - user programs 72
  - validation routines 176, 177
- communication area
  - for name validation routines 177
  - for value validation routines 176
- completion of user program 214
- concatenating status codes 323
  - in the command list 325
  - on the SETCLIST command 326
- control area (see PADA) 217
- COPY command
  - access authorization checks 19
  - example for DDUSER subjects 47
  - for attribute-type subjects 81
  - options for ATTRTYPE subjects 81
  - options for STRTYPE subjects 185
  - use with CATEGORY subjects 91
  - use with relationship-types 104
- COPY statement
  - syntax for Assembler 335
  - syntax for COBOL 331
- customized relationship-types 51
- customizing Header forms 44

- data
  - retrieving through user-written programs 224
  - transferring through EXPORT command 349
- data area (see PADA) 218
- data areas for Dictionary-user program
  - communication 216
- data retrieval 215
  - examples of 230
- data transfer
  - planning for 350
  - using the EXPORT command 357
- DATATYPE 76
- DBCS
  - access control 6
  - character strings 7
  - currency symbols 6
  - DATATYPE 76
  - default value 6
  - defined 6
  - mixed string 7

- restriction against change of status 103
- CHANGE_RELATIONSHIP_DATA command
  - for ATTRTYPE relationships 81
  - for CATEGORY-ATTRTYPE relationships 92
  - for category/attribute relationship 93
  - for RELTYPE-ATTRTYPE relationships 104
  - for STRTYPE-RELTYPE relationships 186
  - to delete attributes 105
- character set, Dictionary
  - allowed for attribute values 77
  - allowed for category names 89
- characteristics of attributes 55
- COBOL
  - accessing Dictionary data 70
  - coding program to access Dictionary data 274
  - example of Language Preprocessor segment 336
  - inclusion statements for 296
  - parameter area for user exit routine 318
  - syntax of inclusion statement 331
  - validation routine entry point 178
  - validation routines for 59, 176
- coding a CALL for retrieval or output
  - for a COBOL program 275
  - for a PL/I program 280
  - for an Assembler language program 285
- coding a COBOL user-written program 274
  - CALL statement for 275
  - documenting your program 287
  - entry point for a 274
  - example program 277
  - link-editing your program 277
  - parameter string area for 274
  - program access control blocks for 275
  - returning to the Dictionary 276
- coding a PL/I user-written program 278
  - CALL statement for 280
  - documenting your program 287
  - entry point for a 278
  - example program 282
  - link-editing your program 281
  - parameter string area for 278
  - program access control blocks for 279
  - returning to the Dictionary 280
- coding an Assembler language user-written program 283
  - CALL statement for 285
  - documenting your program 287
  - entry point for a 283
  - example program 287
  - link-editing your program 286
  - parameter string area for 283
  - program access control blocks for 284
  - returning to the Dictionary 285
- coding options for the PADAFT
  - build and execute 219
  - prebuild 220
  - verify and execute 219
- coding your own PADAFT 219
- combining several types of retrieval 230
- command groups
  - abbreviations used in 26
| Name Rules | 7 |
| NAMETYPE | 89 |
| shift characters | 6 |
| status rules | 6 |
| Text Rules | 7 |
| uniform string | 7 |
| DBCS support | viii |
| DBD.MACLIB library | 73 |
| DBDGFALT module | 25 |
| setting defaults in | 23 |
| DBDGSGRP macro | |
| abbreviations required by | 26 |
| list of command abbreviations | 26 |
| result if you omit | 28 |
| DBDWLANA module | 286 |
| DBDLWNLK module | 277 |
| DBDLWNK module | 281 |
| DDSPEXT data base | 127 |
| DDUSER category | 13 |
| access control status code | 14, 45 |
| category access | 15 |
| subject names | 29 |
| DDUSER display form | 39-41 |
| DDUSER subject | contains access descriptions | 31 |
| DDUSER subject reports | 47 |
| Default Options Table | for Language Preprocessor | 307 |
| list of options | 308 |
| default qualifiers | setting in default module | 23 |
| default structure-type | 68 |
| for standard categories | 68, 203 |
| GUIDE report for | 206 |
| modifying | 206 |
| procedure for changing | 207 |
| Defaults Options Processor | for Language Preprocessor | 307 |
| how to use | 315 |
| operands for | 310 |
| options initialized with | 310 |
| DEFAULTS record | syntax for | 37 |
| defining a Dictionary access request | 217 |
| definitions, modifying | for attribute types | 78 |
| for installation-defined categories | 91 |
| for installation-defined relationship-types | 103 |
| for installation-defined structure-types | 185 |
| DELETE command | for attribute-type subjects | 81 |
| for STRTYPE definitions | 186 |
| use with CATEGORY subject | 92 |
| use with relationship-types | 104 |
| DELETE_RELATIONSHIP command | for category/attribute relationships | 93 |
| for relationship/attribute relationships | 104 |
| for STRTYPE-RELTYPE relationships | 186 |
| to delete STRTYPE relationships | 187 |
| DELETE_RELATIONSHIP_DATA command | restriction with | 105 |
| DELETE_STRUCTURE command | access authorization checks | 19 |
| for STRTYPE definitions | 186 |
| use with CATEGORY subject | 92 |
| deleting relationships to ATTRTYPE | on CATEGORY-ATTRIBUTES form | 169 |
| with commands | 93, 104 |
| deleting relationships to STRTYPE | with commands | 187 |
| DESC | text entries for extensibility subjects | 79 |
| description | for extensibility subjects | 55 |
| in display forms | 79 |
| description data text retrieval | 238 |
| DESCRIPTION display form | 165 |
| description of | description of retrieval requests | 231 |
| DETAIL option, used with RPT command | 82, 95, 108, | 192 |
| diagnostic output | for Language Preprocessor | 302 |
| Dictionaries | multiple | 10 |
| Dictionary | accessing data from | 209 |
| administration tasks | 3 |
| data bases | 49 |
| installation planning | 1 |
| messages for validation routines | 179 |
| overview of administration | 1 |
| receiving | 10, 349 |
| requesting access to | 214 |
| source | 10, 349 |
| directed relationship-type | defined | 62 |
| directed relationships | 53, 101 |
| direction in a structure-type | 62, 63 |
| display forms | |
| ALIAS form | 171 |
| for RELTYPE subjects | 171 |
| CATEGORY-RELTYPES form | 170 |
| considerations for access control | 20 |
| DDUSER form | 39-41 |
| DESCRIPTION form | 165 |
| EXTENSIBILITY EXPLANATION form | 165 |
| EXTENSIBILITY SUBJECT form | for ATTRTYPE subjects | 173 |
| for CATEGORY subjects | 168 |
| for defining STRTYPE | 200 |
| overview for extensibility subjects | 167 |
| RELATIONSHIP DATA form | 165 |
| RELTYPE-ATTRIBUTES form | 172 |
| STRTYPE-RELTYPES form | 202 |
| SUBJECT-RELTYPES forms | 174 |
| DL/I parent of a segment, locating | 230 |
documentation for STRTYPE subjects 184
documentation for users 289
documenting a Program Access program 287-289
double byte character set 6
see also DBCS
DOWN accessing (list-type retrieval) 227
DOWN count 227

E

element-element relationships 52
end users
control of Dictionary access control 47
GUIDE report update if new relationship types 121
informing of validation outcome 179
name of relationship-type on GUIDE report 125
relationship-type name not generally visible 100
use of CATEGORY aliases 96
use of CATEGORY GUIDE report 95
use of RELTYPE GUIDE report 108
use of STRTYPE GUIDE report 192
use of structure-type aliases 192
view of attribute keyword name 80
view of attribute-type name 76
entering definitions on display forms 165-174
EXTENSIBILITY SUBJECT form 165
entry point, for a user-written program for a COBOL program 274
for a PL/I program 278
for an Assembler language program 283
entry points
to validation routines 178
entry statements
for validation routines 178
error return codes for alias retrieval 245
output request 273
relationship (RSR) retrieval 254
relationship (RSRC) retrieval 257
relationship (RSRK) retrieval 259
relationship attribute data retrieval 237
subject attribute data retrieval 234
subject name and alias retrieval 249
subject name retrieval 249
text data retrieval 242
validation routines 179
establishing Dictionary user access control 24
recommended procedure 24
example
COPY command for DDUSER subjects 47
example of
ADD command for STRTYPE subject 208
ADD_RELATIONSHIP command for STRTYPE subject 208
ADD_RELATIONSHIP command for STRTYPE-RELTYPE relationships 186
adding alias name for STRTYPE definition 185
adding RELTYPE category 103
adding STRTYPE subject 185
Assembler program to access Dictionary data 287
Assembler segment 344
changing STRTYPE definition 185
COBOL program 277
command user access profile 39
data retrieval 230
developer access profile 38
EXPORT processing report 379
FORMAT report with optimized storage 134
generated commands from EXPORT 375
GUIDE report 159, 164
INSTALL command for STRTYPE subject 208
installation processing 157
installation processing report 163
Language Preprocessor profile 39
Language Preprocessor segment 336
Language Preprocessor SYSPRINT output 304
of ATTTRTYE subject-specific report 85
PL/I program to access Dictionary data 282
PL/I segment 341
PREVIEW processing report 395
Program Access control block usage 221
Program Access programs 215
relationship-type FORMAT report 133
relationship-type GUIDE report 127
RELTYPE report 109
report to install a relationship-type 120
report to install a subject category 119
security administration access profile 38
STRTYPE GUIDE report 195
STRTYPE installation report 199
structure-type definition 190
subject category FORMAT report 132
subject category GUIDE report 124
subject-specific category report 96
subject-specific STRTYPE report 193
User Access Profiles 38
example of a Program Access program written in Assembler language 287
written in COBOL 277
written in PL/I 282
EXECUTE command operands for 212
to invoke a user program 72
to invoke user-written program 209
EXPLANATION display form 165
explanation text
for category definitions 90
for relationship-types 102
EXPORT command 357-361
exporting data
through EXPORT command 349
extensibility control information
definition operands for attribute-types 79
extensibility control information categories 55
category numbers 55
defining 55
definition operands for attribute-types 76
definition operands for categories 88
definition operands for relationship-types 101-103
definition operands for structure-types 184
deleting 57
establishing for Language Preprocessor 328
in access profiles 15
installing 57
modifying 58
Extensibility facility 50
what it is 49
extensibility subject
display form for ATTRTYPE subject 173
display form for CATEGORY subject 168
display form for CATEGORY-ATTRIBUTES 169
display form for CATEGORY-RELTYPES 170
display form for RELTYPE subject 171
display form for RELTYPE-ATTRIBUTES 172
overview of display forms for 167

fields
defined for validation routines 176, 177
finding a segment's DL/I parent 230
finding a subject among interrelated subjects 230
FIRST accessing (list-type retrieval) 227
FOLLOW DOWN processing
defining 189
explained 65
FOLLOW UP processing
defining 189
explained 65
FORMAT reports 127, 130-133
extample for CATEGORY subject 131
example for RELTYPE subject 132
example of for subject category 132
for CATEGORY subjects 130-132
for RELTYPE subjects 130-131, 133
use for optimizing storage 133-134
format table (see PADAFT) 218
forward and inverse names 51, 101

GIS/V5
use of 130
glossary 477
glossary report
for DDUSER subject 47
GUIDE reports 121-127, 194-195
example for CATEGORY subject 122-124
example for RELTYPE subject 126-127
example for STRTYPE subject 194-195
for CATEGORY subjects 121

for default structure-type 206
for RELTYPE subjects 125

H

Header form 35, 44
(see also Sign-On Header form)
HERE accessing (list-type retrieval) 227
hierarchy of display forms 166
how a user-written program operates 210

I

IBM-supplied default structure-type 68, 203
ICL statement
syntax for Assembler 335
identification of Dictionary user 13, 29
ignoring retrieved data 218
INCLUDE statement
syntax for 333
inclusion statement
how they work 296
syntax for Assembler 335
syntax for COBOL 331
syntax for PL/I 333
indefinite status codes
use of by Language Preprocessor 323, 325
INSTALL command 111-116, 117
example of for STRTYPE subject 208
format 111
installation processing reports 117-120, 199
for a relationship-type 119
for a structure-type 199
for a subject category 118
using 112
validation performed
for attribute-type definitions 117
for category definitions 115
for relationship-type definitions 116
validations performed by 198
INSTALL STRTYPE command 197-198
installation
planning for 1
installation-defined categories
adding and updating 91, 185
as Dictionary extensions 50
categories for 55
change procedure 93
definition operands 88-90, 184
deleting 92, 186
modifying 58
procedure for changing 200
relating attribute-types 92, 186
reporting development 95-97
restrictions for deleting 57
installation-defined keywords 54, 79
installation-defined relationship keywords 51, 101
installation-defined relationships
adding and updating 103
as Dictionary extensions 51-54
change procedure 105
definition operands 101-103
deleting 104
relating attribute-types 104
reporting development 108-109
installation-defined structure-types
reporting development 192-195
installing extensibility definitions
as a prerequisite 50, 111
categories 111
relationship-types 111
using the INSTALL command 112
interface, program with Dictionary 213
inverse names 51, 101

K

keywords
used in access profiles 32

L

Language Preprocessor
administration of 291
advantages of 292
authorizing use of 321
command lists for 298
Default Options Table 307, 308
Defaults Options Processor 307, 315
diagnostic output 302
establishing an extensibility category for 328
example for COBOL 336
example of Assembler segment 344
example of PL/I segment 341
example of profile 39
example of SYSPRINT output 304
how data structures are selected 298
how Dictionary processing is interpreted 299
how it works 293
how output is requested 299
inclusion statements for 296
listing output 303
options for 307
options initialized for 310
output for 301

processing environment 293
testing command list readiness 329
use of status codes 325
user exits 300, 317
using indefinite status codes 323
LAST accessing (list-type retrieval) 227
link-edit
validation routines 180
link-editing validation routines 181
for a COBOL program 180
for a PL/I program 181
for an assembler program 181
link-editing your program
for a COBOL program 277
for a PL/I program 281
for an Assembler language program 286
list-type retrieval
for alias retrieval 243
for relationship-type retrieval 262, 266
for subject name and alias retrieval 246
for subject name retrieval 246
for text data retrieval 238
locating data in the PADA 228
overview of 225
PADAFT for a 226
positioning 227
relationships for a subject 250
lists, on display forms
moving entries within 169
of attribute-types for a relationship-type 172
of installed relationship-types 170
locating data in the PADA 218
locating list data in the PADA 228

M

messages
for validation routines 179
issued by validation routines (batch output) 179
issued by validation routines (display forms) 179
sending for user-written programs 216
mixed string
defined 7
Text Rules 7
MODE (new or old)
RELTYPE-ATTRIBUTES (relationship) form 172
modifying a master PADAFT before retrieval 220
alias retrieval 244
description data text retrieval 239
PL/I data text retrieval 240
relationship retrieval 261
relationship-type retrieval 263, 267
subject name and alias retrieval 247
user data text retrieval 240
NAME in RPT command
  for attribute-type  82
  for category       95
  for DDUSER subjects 47
  for relationship-types 108
  for structure-type  192
NAMETYPE  89
naming conventions   8
  for relationship-types 100
navigation through a network  230
NEXT accessing (list-type retrieval)  227

operation of a user-written program  210
operator identification, on CSSN transaction  29
options for the PADAFT
  build and execute  219
  prebuild          220
  verify and execute 219
output messages
  sending for user-written programs   216
output of REPORT command
  for reporting DDUSER subjects   47
output requests 72, 271
output routing
  with EXPORT command   354, 355
output via the Dictionary
  overview of          211

PACA (control area)  217
example of           221
list of required fields for
  alias retrieval   243
output request       271
relationship (RMR) retrieval 251
relationship (RMRK) retrieval 255
relationship (RSMK) retrieval 258
relationship attribute data retrieval 235
relationship-type retrieval 262, 266
subject name and alias retrieval 246
text data retrieval  238
list of returned-data fields for
  alias retrieval    244
output request       272
relationship (RSM) retrieval 253
relationship (RSMK) retrieval 256
relationship attribute data retrieval 236
relationship-type retrieval 264, 269
subject name and alias retrieval 248
text data retrieval  241
using the       217
PADA (data area)  218
example of        223
locating list data in 228
using the         218
PADAFT (format table)  218
and retrieval requests 228
coding options for 218
  build and execute 219
  prebuild          220
  verify and execute 219
example of          221
for a list-type request 226
reusing            229
using the         218
parameter string area
  for a COBOL program 274
  for a PL/I program 278
  for an Assembler language program 283
passing parameters to a user program 212
PL/I
  accessing Dictionary data  70
  coding program to access Dictionary data 278
data text retrieval  238
example for Language Preprocessor   341
inclusion statements for  296
parameter area for user exit routine 319
syntax of inclusion statement 333
validation routine entry point 178
validation routines for   59, 176
Portability facility
  used to transfer Dictionary data 349
  what it is          10
positioning data for retrieval
  positioning data for subsequent retrieval
    relationship (RMR) retrieval 252
    relationship (RSM) retrieval 254
  relationship (RSMK) retrieval 256
  relationship (RSMK) retrieval 259
  relationship-type retrieval 264, 267
positioning within a list         227
prebuild option (of the PADAFT)   220
PREVIEW command 381-382
processing action for structure-types  64
production status, use of   75, 183
profile, access (see access profiles)
program access control blocks
  for a COBOL program 275
  for a PL/I program 279
  for an Assembler language program 284
Program Access facility 9, 209-289
  Dictionary-supplied programming aids for 73
getting attribute data    71
planning for            70
program output         72
summary of support      73
Program Access facility (see also user-written programs)
what it is 49
program completion 214
Program Entry Points 178
program name
for user programs 72
for validation routines 180
specified for name validation 89-90
specified for value validation 78
program-with-Dictionary interface 213
data areas 216
example of 221
for retrieval 215
for sending output 216
relationship of 217
programming aids, Program Access
sample programs 73
programming restrictions
user programs 72
validation routines 72

relationship-type subjects
relating with STRTYPE subjects 186
relationship-types
concepts 51-54
relationship-types for a category, retrieval of 262, 266
relationship, alias 52
RELATIONSHIPS action, on display forms 174
relationships for a subject, retrieval of 250
relationships possible between categories 51-54, 99
RELTYPE definition
adding 103
category name restriction 102
copying 104
defining 99
deleting 103, 104
example of adding 103
for installation-defined entities 55
FORMAT reports for 130
GUIDE report for 127
modifying 105
operands to define 101
relating to attribute-type 104
reports on 108
requirements for sequencing attribute 105
subject names for 99
suggested naming convention 100
update commands 103
updating 103
using commands with 103
validation for 116
RELTYPE form
description of 171
every example of 171
RELTYPE subjects
relating with STRTYPE subjects 186
RELTYPE-ATTRIBUTES form
description of 172
every example of 172
REPORT and SCAN commands
for attribute-type definitions 82
for category subjects 95
for relationship-type definitions 108
for reporting Dictionary users 47
for structure-type definitions 192
REPORT command
display form equivalent reports 47
subject-specific reports 82-85, 108
reports
CATEGORY FORMAT 130
example of EXPORT processing 379
every example of FORMAT with optimized storage 134
example of GUIDE 159, 164
every example of PREVIEW processing 395
example of relationship-type FORMAT 133
example of STRTYPE GUIDE 195
every example of STRTYPE installation 199
example of subject category FORMAT 132
example of subject-specific ATTRTYPE 82
example of subject-specific CATEGORY 96
example of subject-specific STRTYPE report 193
EXPORT processing 375
for DDUSER subjects 47
for installing a subject category 119
for installing STRTYPE 199
FORMAT 127
GUIDE for default structure-type 206
GUIDE for STRTYPE definitions 192
installation processing 117, 157, 163
PREVIEW processing 384
relationship-type definition 108
RELTYPE FORMAT 130
subject category definition 95
with output routing 355
representing structure-types 62
requesting Dictionary access 214
required PACA fields for
alias retrieval 243
list of returned-data fields for
subject attribute data retrieval 233
output request 271
relationship (RSR) retrieval 251
relationship (RSRC) retrieval 255
relationship (RSRK) retrieval 258
relationship attribute data retrieval 235
relationship-type retrieval 262, 266
subject attribute data retrieval 232
subject name and alias retrieval 246
text data retrieval 238
requirements for a user-written program 212
restrictions
for deleting installation-defined categories 57
for updating attribute-type definitions 81
on category access 16, 23
on DDUSER subject names 29
on relationship-types 102
on set of characters for attribute values 76
on the status of related attributes 93, 187
on user programs 72, 214
on validated values 77
retrieval requests
combining several types 230
for a relationship's attribute data 235
for a subject's attribute data 232
for a subject's relationships 250
for alias retrieval 243
for relationship-type retrieval 262, 266
for subject name and alias retrieval 246
for subject name retrieval 246
for text data retrieval 238
how request-types are described 231
PADAFT for 228
alias retrieval 244
description text retrieval 239
PL/I data text retrieval 240
relationship retrieval 261
relationship-type retrieval 263, 267
subject name and alias retrieval 247
user data text retrieval 240
reference sources for
summary of 228
retrieve subject relationships (RSR) 251
retrieve subject relationships by related category (RSRC) 255
retrieve subject relationships, by related category and keyword (RSRK) 258
retrieving a relationship's attribute data 235
retrieving a subject's aliases 243
retrieving a subject's attribute data 232
list of required fields for
subject attribute data retrieval 232
retrieving attributes of subjects and relationships
alias retrieval 243
overview of 224
relationship attribute data retrieval 235
relationship-type retrieval 262, 266
relationships for a subject 250
subject attribute data retrieval 232
subject name and alias retrieval 246
text data retrieval 238
retrieving Dictionary data 209, 215
ignoring some data items 218
overview of 209, 210
requests for 228
retrieving lists of data
locating data in the PADA 228
overview of 225
PADAFT for a 226
positioning 227
retrieving subject names and their aliases in a
category 246
retrieving subject names in a category 246
retrieving text data 238
retrieving the text-portion of a line 239
return codes
for validation routines 70, 179
set for user-written programs 214
set in PACA 216
setting for user-written programs 214
return codes set for a
alias retrieval 245
output request 273
relationship (RSR) retrieval 254
relationship (RSRC) retrieval 257
relationship (RSRK) retrieval 259
relationship attribute data retrieval 237
subject attribute data retrieval 234
subject name and alias retrieval 249
subject name retrieval 249
text data retrieval 242
returned-data PACA fields for
alias retrieval 244
output request 272
positioning data for subsequent retrieval
relationship (RSRC) retrieval 256
relationship (RSRK) retrieval 259
relationship-type retrieval 264, 267
relationship (RSR) retrieval 253
relationship (RSRC) retrieval 256
relationship attribute data retrieval 236
relationship-type retrieval 264, 269
subject attribute data retrieval 233
subject name and alias retrieval 248
text data retrieval 241
returning to the Dictionary
for a COBOL program 276
for a PL/I program 280
for an Assembler language program 285
reusing the PADAFT 229
routines, validation
communication area for attribute values 176
communication area for name validation 177
general requirements of 175
installing 180
link-edit requirements 180
managing validation libraries 182
planning for 58
planning for attribute value 60
programming requirements 176-182
required operand
for ATTRTYPE relationships 79
requirements for validation 176-182
restrictions for programming 72
return codes and messages 179
specifying 78
specifying for CATEGORY subject 89
specifying names of 77, 89-90
validation routines 180
RSR request 251
RSRC request 255
RSRK request 258

S

sample Program Access program
written in Assembler language 287
written in COBOL 277
written in PL/I 282
sample user programs 73
segment names
for extensibility data 128
for validation data 129
SEQ keyword
required operand
for CATEGORY relationships 92
for RELTYPE relationships 104
for STRTYPE relationships 186
sequence of display forms 166
sequenced relationships
attribute-type criteria 105
concepts 53
for installation-defined categories 53
for installation-defined relationship-types 53
relationship-type definition operands 104
representation in GUIDE reports 125-128
SETCLIST command 326
concatenating status codes 326
use of by Language Preprocessor 322
shift-in character (>) 6
shift-out character (<) 6
si 6
see also shift-in character
Sign-On Header form
messages 45
sample form 44, 45
SIGN_ON command 46
so 6
see also shift-out character
specifying a parameter string area
for a COBOL program 274
for a PL/I program 278
for an Assembler language program 283
specifying attributes
as attribute-types 76-80
for installation-defined categories 92
for installation-defined relationship-types 104
for installation-defined structure-types 186
to have validation 77
specifying program access control blocks
for a COBOL program 275
for a PL/I program 279
for an Assembler language program 284
starting a user program 212
status code specification
for access profiles 32
status codes
concatenating for Language Preprocessor 325
for access profiles 14, 32
use of for Language Preprocessor 323, 325
status codes, indefinite
use of by Language Preprocessor 323, 325
storage
of attribute values 128
of validation values 129
optimizing attribute data 133
STRTYPE category
adding 185
defining subject name 183
definition operands 184
updating 185
using commands with 185
STRTYPE definitions
copying 185
defining processing options for 188
deleting 186
display forms for defining 200
example of 190
example of adding alias name for 185
equivalent of installation report 199
example of subject-specific subject report 193
GUIDE reports for 192
INSTALL command for 197
installing 195
modifying DEFAULT 206
procedure for changing 200
procedure for changing DEFAULT 207
relating with relationship-types 186
used for structure-type definition 185
user access control
access profiles 31-39
access rules 16
category name and number pairs 14, 15, 34-37
command considerations 18
command requirements 23
DDUSER display form 39-41
defining for status code 14
defining for subject category 15
display form considerations 20
enabling 24
examples of access profiles 38-39
for user-written programs 214
general information 13-14
Header form 35, 44, 45
messages 45
model profile 43, 44
relationship requirements for commands 16
specifying for relationships 17
USER DATA form 41-44
user identification 29
using the ADD command 30
view or update authorization 14, 32
view or update status 32
user data
adding for extensibility subjects 79
as explain text 56, 90, 102
for access profiles 31-37, 41-44
user data text retrieval 238
user exit routine
entry points and returns for 319
in Language Preprocessor 300
link-editing 320
parameter area for Assembler 320
parameter area for COBOL 318
PL/I parameter area for 319
user-program output 271
user-written program
access authorization checking 214
completion 214
data retrieval 215
documenting your program 287
examples of 215
executing a 212
how to write 212
interface with Dictionary 213
invoking a 212
overview of 209
passing parameters to 212
requirements 212
restrictions 214
retrieving subject aliases 243
sending output 216
written in Assembler language 283
written in COBOL 274
written in PL/I 278
using a master PADAFT with retrieval requests 219
using the EXPORT command 351

V
validation routines
communication area for attribute values 176
communication area for name validation 177
general requirements of 175
installing 180
link-edit requirements 180
managing validation libraries 182
planning for 58
planning for attribute value 60
requirements for 176
return codes and messages 179
specifying 78
specifying for CATEGORY subject 89
validation table
use of 129
verify and execute option (of the PADAFT) 219

W
writing programs to access Dictionary data 209, 212
overview of 209
programming languages 212
requirements 212
This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you wish a reply, give your name, company, mailing address, and telephone number.

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL ____________________________

Previous TNL ____________________________

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Thank you for your cooperation.
Reader's Comment Form

Fold and tape  Please do not staple  Fold and tape

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.
POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape  Please do not staple  Fold and tape
This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author’s department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you wish a reply, give your name, company, mailing address, and telephone number.

______________________________

______________________________

______________________________

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL ______________________

Previous TNL __________________

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.) Thank you for your cooperation.