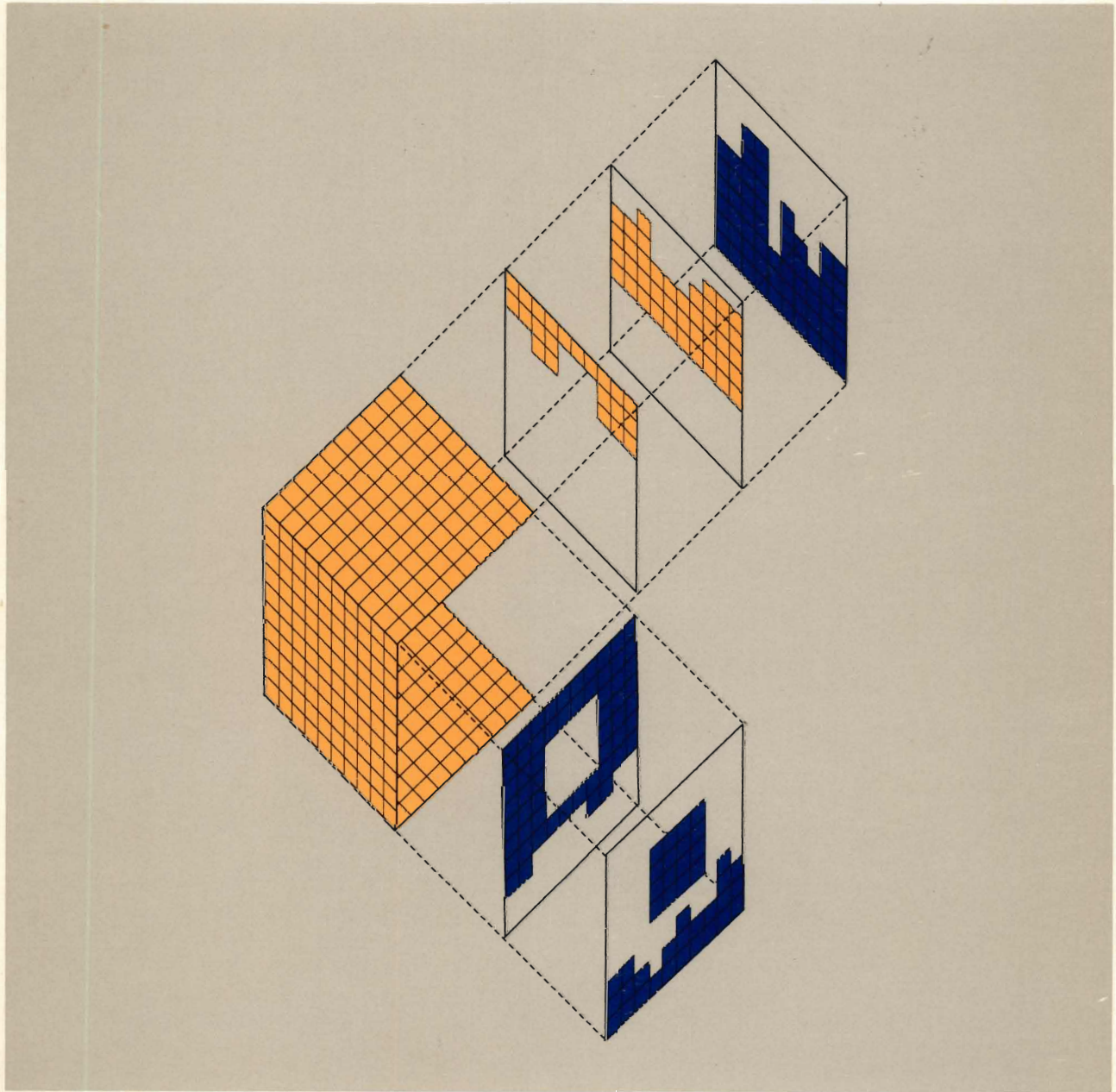


Cross System Product/Application Development

External Source Format Reference

Version 3 Release 3







Cross System Product/Application Development

External Source Format Reference

Version 3 Release 3



Second Edition (May 1990)

This edition applies to Version 3 Release 3 of the licensed programs Cross System Product/Application Development (CSP/AD) for CICS/VS, VM CMS, and MVS/TSO, program number 5668-813, and Cross System Product/Application Execution (CSP/AE) for CICS/VS, VM CMS, and MVS/TSO, program number 5668-814, Cross System Product/370 Runtime Services for IMS/VS and IMS/ESA, program number 5688-150, and to Service Level Update SLU-0200/Fix Package FP-0202 of the licensed program Cross System Product/Application Execution for DPPX/SP, program number 5660-285.

This edition applies to subsequent releases and modifications of this program until otherwise indicated. The licensed programs and related licensed material described herein are provided by IBM under the Agreement for IBM Licensed Programs.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. **This disclaimer does not apply in the United Kingdom or elsewhere if inconsistent with local law.**

This publication may include inaccuracies or errors. IBM may change this publication and/or the product described herein. Textual changes are indicated by a vertical line to the left of the change.

This book is not intended for production use and is furnished as is. IBM assumes no responsibility for the use of the functions as described in this book in any production manner.

IBM may have patents or pending patent applications covering subject matter described herein. This document neither grants nor implies any license or immunity under any IBM or third-party patents, patents applications, trademarks, copyrights, or other similar rights, or any right to refer to IBM in any marketing activities. Other than responsibilities assumed via the Agreement for Purchase of IBM Machines and the Agreement for IBM Licensed Programs, IBM assumes no responsibility for any infringement of third-party rights that may result from use of the subject matter disclosed in this document or from the manufacture, use, lease, or sale of machines or programs described herein.

Licenses under IBM's utility patents are available on reasonable and nondiscriminatory terms. IBM does not grant licenses under its appearance design patents. Direct licensing inquiries in writing to the IBM Director of Commercial Relations, International Business Machines Corporation, Armonk, New York, 10504.

References in this publication to IBM products or services do not imply that they will be available everywhere IBM operates, nor that only IBM's products or services may be used.

Publications are not stocked at the address below. Request IBM publications from your IBM representative or branch office.

A form for comments is provided at the back of this publication. Or you may address comments to: IBM Corporation, Department T45, P.O. Box 60000, Cary, North Carolina 27512-9968. IBM may use and distribute information you supply without obligation to you.


Note to U.S. Government users - Documentation is related to restricted rights; use, duplication, or disclosure is subject to restrictions set forth in the GSA ADP Schedule Contract with IBM Corp.



Special Notices

Before using this manual, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, *IBM System/370 and 4300 Processors Bibliography of Industry Systems and Application Programs*, GC20-0370, *IBM 8100 Information System Bibliography*, GC20-8100, or the IBM* *AS/400 Information Directory*, GC21-9678 for applicable and current editions.

The following names, used in this manual, are trademarks of the IBM Corporation in the United States and/or other countries:



CMS
DB2
International Business Machines
IBM
MVS
Operating System/2
OS/2
Operating System/400
OS/400
SQL
VM



An asterisk is used to identify the first time a trademark name is used in the text.



* A trademark of IBM

Preface

This manual is a reference manual for Cross System Product/Application Development (CSP/AD) external source format. This publication supplements the *CSP/AD Reference* manual by describing the external source format syntax. It describes the CSP/AD structures only as they relate to the external source format. For additional information on these structures, refer to the *CSP/AD Reference* manual.

This manual is intended to help you convert applications developed with non-Cross System Product design tools into CSP/AD. It primarily contains general-use programming interfaces, which allow the customer to write programs that use the services of the Cross System Product set.

However, this manual also provides the following types of information, which are explicitly identified where they occur:

Other Product Information

Other product information, such as information about commands and messages is provided to allow the customer to use this product. This information should never be used as programming interface information.

This manual also includes various examples of code and data definition sequences. They are intended only as general guidance and do not define a programming interface. For a complete summary of the Cross System Product programming interfaces, refer to "Directory of Programming Interfaces for Customers" in the *CSP/AD Reference* manual.

Who Should Use this Manual

This manual is intended for the application programmer, application developer, or personnel with strong technical skills who develop programs and applications.

How this Manual is Organized

This manual contains the following chapters:

- Chapter 1, "Introduction," provides an overview and sample formats of the external source format.
- Chapter 2, "Application Structures," describes the external source format tags and attributes that define an application structure in CSP/AD.
- Chapter 3, "Process Structures," describes the external source format tags and attributes that define a process structure in CSP/AD.
- Chapter 4, "Statement Group Structures," describes the external source format tags and attributes that define a statement group structure in CSP/AD.
- Chapter 5, "Record Structures," describes the external source format tags and attributes that define records in CSP/AD.
- Chapter 6, "Table Structures," describes the external source format tags and attributes that define tables in CSP/AD.
- Chapter 7, "Data Item Structures," describes the external source format tags and attributes that define data items in CSP/AD.

- Chapter 8, “Program Specification Block Structures,” describes the external source format tags and attributes that define a program specification block (PSB) structure in CSP/AD.
- Chapter 9, “Map Structures,” describes the external source format tags and attributes that define map structures in CSP/AD.
- Chapter 10, “Map Group Structures,” describes the external source format tags and attributes that define map group structures in CSP/AD.
- Appendix A, “External Source Format Functions with CSP/AD and CSP/AE Commands,” describes the CSP/AD and CSP/AE commands that control external source format functions.
- Appendix B, “DBCS Support,” describes CSP/AD support for DBCS data.
- Appendix C, “Report Formats,” describes the formats for printed messages.

Prerequisite Publications

Before reading this manual, you should read the *Cross System Product Set General Information* manual, which contains an overview of the CSP/AD and CSP/AE products.

Application programmers using this manual to import or export MSL members should refer to the *CSP/AD Reference* manual and the *CSP/AD Developing Applications* manual.

Related Publications

If you are a DPPX/370 user, you should be familiar with the following publications before using this manual:

- *CSP/AD for DPPX/370 User's Guide*, SC23-0684
- *CSP/AE for DPPX/370 User's Guide*, SC23-0668.

For additional information on CSP/AD applications generated as COBOL programs, refer to the information supplied with the Cross System Product/370 Runtime Services Systems product.

For additional information on CSP/AD applications defined on a CSP/AD Programmable Workstation Feature, refer to the information supplied with the workstation product.

Cross System Product Information Library

The Cross System Product library is an important part of the Cross System Product set. The books explain how to use the product to build applications, manage the object libraries, and interface with the database. In addition, online help, samples, tutorials, and reference material are available.

When you order any of the Cross System Products, you receive a complete set of the books. You can order additional copies of these books separately using the order numbers found in the following descriptions, or as a set, using the order number **SBOF-2260**. The Licensed Program Specifications are shipped only with the product tape as part of the original set of books.

The following descriptions summarize the books in the Cross System Product library.

Cross System Product

General Information

Order Number: GH23-0500-4

This book outlines the Cross System Product set and helps you evaluate and plan for the Cross System Product set.

Cross System Product/Application Development ***Cross System Product/Application Execution***

Administering CSP/AD and CSP/AE on MVS

Order Number: SH20-6765-0

Administering CSP/AD and CSP/AE on VM

Order Number: SH20-6766-0

Administering CSP/AD and CSP/AE on VSE

Order Number: SH20-6767-0

Administering CSP/AD and CSP/AE on DPPX

Order Number: SH20-6768-0

Administering CSP/AD and CSP/AE on OS/2 and IBM DOS

Order Number: SH20-6769-0

These books describe the administrative tasks involved in managing application libraries and other system level tasks for the various environments.

Cross System Product/Application Development

Defining Applications on the System/370

Order Number: SH20-6434-0

This book describes the processes involved in defining CSP/AD applications specifically on the System/370.

Cross System Product/370 Runtime Services

Generating and Running IMS Applications

Order Number: SH23-0514-0

This book describes how to generate VS COBOL II applications from applications developed using CSP/AD.

Cross System Product/Application Development

Developing Applications

Order Number: SH20-6435-0

This book describes the processes involved in designing, testing, and generating CSP/AD applications for multiple environments.

Cross System Product/Application Development

Cross System Product/Application Execution

Messages, Codes, and Problem Determination

Order Number: SH23-0505-4

This book lists the messages, return codes, and error codes issued by the Cross System Product. It also includes EZ-PREP and EZ-RUN messages, CSP/AD Programmable Workstation Feature messages, and a discussion of how to diagnose problems. Each message contains an explanation and the appropriate user response.

Cross System Product/Application Development

Reference

Order Number: GH23-0515-3

This book describes how to write Cross System Product applications having the same function on all supported systems.

Cross System Product/Application Development

External Source Format Reference

Order Number: SH20-6433-1

This book describes how to use the CSP/AD external source format. Using this reference, you can convert applications developed with non-Cross System Product design tools to CSP/AD applications.

Cross System Product/Application Development

Cross System Product/Application Execution

Planning

Order Number: SH20-6770-0

This book helps you plan for the installation and implementation of the Cross System Product set.

Binders

Order Number: SX80-0258-0

Inserts and Labels for Binders

Order Number: GH23-0513-2

Cross System Product/Application Development

Reference Summary

Order Number: SH23-0519-1

This reference summary provides logic diagrams of processing statements and process options. It also lists highlighting defaults, special function words, command area and line edit commands, and PF key functions.

Cross System Product/Application Development

External Source Format Reference Summary

Order Number: SH20-0520-0

This reference summary provides diagrams of the external source format tags used to create CSP/AD applications.

The following legal documents contain information concerning the hardware and software requirements for the Cross System Product set. They also serve as a warranty.

***Licensed Program Specifications
Cross System Product/Application Development for S/370***

Order Number: GH23-0510-5

***Licensed Program Specifications
Cross System Product/Application Execution***

Order Number: GH23-0511-5

***Licensed Program Specifications
Cross System Product/Application Execution for Distributed Processing
Programming Executive/System Product***

Order Number: GH20-0955-7

***Licensed Program Specifications
Cross System Product/370 Runtime Services***

Order Number: GH20-6428-0



Contents

Chapter 1. Introduction	1-1
External Source Format Tags	1-2
Format Example	1-5
Chapter 2. Application Structures	2-1
Application Definition	2-1
Main Process Definition	2-5
Table and Additional Records List Definition	2-6
Parameter Specification	2-7
Application Prolog Definition	2-9
Application Generation Option Specification	2-10
Table Generation Specification	2-14
Target System Specification	2-15
Internal Resource Name Specification	2-16
DPPX File Information Specification	2-19
Application Definition Syntax Example	2-21
Chapter 3. Process Structures	3-1
Process Member Definition	3-1
Logic Definition Before Process Option	3-4
Logic Definition After Process Option	3-5
SQL Selection Condition Specification	3-6
DL/I Call Definition	3-8
Segment Search Argument Definition	3-9
Qualification Statement Definition for the Segment Search Argument	3-11
Process Definition Syntax Example	3-13
Chapter 4. Statement Group Structures	4-1
Statement Group Definition	4-1
Statement Group Processing Definition	4-3
Statement Group Definition Syntax Example	4-4
Chapter 5. Record Structures	5-1
Record Definition	5-1
SQL Table Name Definition	5-6
Default Selection Criteria Definition	5-8
Record Prolog Definition	5-9
Record Item Definition	5-10
Record Definition Syntax Example	5-14
Chapter 6. Table Structures	6-1
Table Definition	6-1
Table Generation Option Specification	6-3
Table Prolog Description	6-4
Table Column Definition	6-5
Data Content Attribute Identification	6-8
Row Specification	6-10
Table Definition Syntax Example	6-11
Chapter 7. Data Item Structures	7-1
Data Item Definition	7-1
Map Edit Characteristic Definition	7-3

Data Item Message Definition	7-7
Data Item Definition Syntax Example	7-9
Chapter 8. Program Specification Block Structures	8-1
PSB Definition	8-1
Program Communication Block Specification	8-2
Segment Sensitivity Specification	8-3
PSB Definition Syntax Example	8-5
Chapter 9. Map Structures	9-1
Map Definition	9-1
Presentation Information Definition	9-4
Constant Field Definition	9-6
Constant Field Attribute Definition	9-7
Variable Field Definition	9-10
Map Edit Characteristic Definition	9-12
Field Edit Message Definition	9-16
Variable Field Attribute Definition	9-17
Map Structure Syntax Example	9-20
Chapter 10. Map Group Structures	10-1
Floating Area Definition	10-2
Map Group Structure Example	10-3
Appendix A. External Source Format Functions with CSP/AD and CSP/AE	
Commands	A-1
Import and Export of Member Specification Library Members	A-1
Files Used For Importing and Exporting	A-2
Error Processing	A-4
CSP/AE Batch Commands	A-5
Appendix B. DBCS Support	B-1
Appendix C. Report Formats	C-1
Glossary	X-1
Index	X-20

Summary of Enhancements

3.3 Enhancements

The enhancements of the Cross System Product set Version 3 Release 3 help improve productivity, portability, and usability:

- CSP/AD Programmable Workstation Feature
- IMS application definition
- Long names support
- Local data items support
- SQL * extensions
- VM/XA * exploitation
- External source format interactive support
- Facility transfer commands support
- Batch generation options support
- New publications.

The CSP/AD Programmable Workstation Feature

The new CSP/AD Programmable Workstation Feature improves the productivity of application developer by enhancing error prevention and keystroke elimination. You can create new Cross System Product applications and modify existing applications using the CSP/AD Programmable Workstation Feature.

Using the OS/2 * Presentation Manager *, the interface for the CSP/AD Programmable Workstation Feature conforms to Common User Access (CUA) and includes a comprehensive set of help windows. The help windows provide information for all fields, tasks, windows, and messages.

The graphical representation of the application structure improves productivity compared to non-graphical alternatives. In the CSP/AD Programmable Workstation Feature, an interface to a graphical editor with icons displays application definitions either in a horizontal or vertical tree-structured format.

You can view or edit an application definition in either format. When editing large application diagrams, you can choose to view or edit the entire diagram or a portion of it. Also, main processes can be manipulated and processes and statement groups can be expanded or collapsed.

IMS Application Definition

CSP/AD now supports the definition of applications for execution in the IMS environment. Both interactive and batch IMS applications can be defined using CSP/AD in the TSO, VM *, or CICS/VS environments or the CSP/AD Programmable Workstation Feature. Existing database applications that run in segmented mode in CICS/VS or as MVS * batch jobs can be retargeted for generation and execution in the IMS environment within normal environmental restrictions.

* A trademark of IBM

Cross System Product/370 Runtime Services generates COBOL applications from CSP/AD application definitions and supports execution in the IMS/VS, IMS/ESA, and MVS batch environments.

CSP/AD supports the definition of message processing programs (MPP), batch message programs (BMP), and DL/I batch programs. IMS applications can support:

- DL/I
- DB2
- Both types of Fast Path databases
 - Data entry databases (DEDBs)
 - Main storage databases (MSDBs)
- Generalized Sequential Access Method (GSAM) files.

CSP/AD applications defined for IMS can be tested through the CSP/AD test facility in the TSO, VM, and CICS/VS environments with some restrictions.

Long Names Support

The support of long names increases flexibility when naming members used in the development and administration of CSP/AD applications. Because the Cross System Product set supports long names, multi-line statements are also supported.

Local Data Items

Allowing data items to be defined or scoped as local data items provides further flexibility in naming data items. This allows the application developer to use common names for working storage variables or other noncritical data items.

SQL Extensions

The enhanced support for Structured Query Language (SQL) includes:

- SQL multi-column indexes
- Database switching
- Concurrent DL/I and DB2 support in MVS batch applications
- The dynamic WHERE clause
- Single-row SELECT
- Modification of FOR UPDATE OF and INSERT INTO clauses in SQL Statements built by CSP/AD
- The ability to generate a model SQL statement for the SQLEXEC process option.

VM/XA Exploitation

CSP/AD and CSP/AE can run in the VM/XA environment in extended address mode making available the performance benefits of this environment.

* A trademark of IBM



External Source Format Interactive Support

CSP/AD utilities now support the export and import of CSP/AD members in external source format in an interactive environment. The addition of online support for this function provides you with greater flexibility and productivity when using external source format files.

Facility Transfer Commands

With the addition of facility transfer commands, the application developer can go to any CSP/AD facility from any CSP/AD panel, with the option of filing or cancelling changes that you made in the current facility. This provides quick and easy movement through CSP/AD functions without having to move up and down the CSP/AD panel hierarchy.

Batch Generation Options

CSP/AD batch generation now allows you to select all options that can be specified in interactive generation.



New Publications

- *Administering CSP/AD and CSP/AE on MVS*
- *Administering CSP/AD and CSP/AE on VM*
- *Administering CSP/AD and CSP/AE on VSE*
- *Administering CSP/AD and CSP/AE on DPPX*
- *Administering CSP/AD and CSP/AE on OS/2 and IBM DOS*
- *Defining Applications on the System/370**
- *Developing Applications*
- *Planning*
- *External Source Format Reference Summary*
- *Generating and Running IMS Applications.*



* A trademark of IBM



Chapter 1. Introduction

The information in this chapter provides no programming interfaces for customers. For detailed information about programming interface information, see the statement about programming interfaces in the Preface of this manual.

The external source format feature of CSP/AD lets you define Member Specification Library (MSL) members in a readable format. Before Version 3 Release 2 Modification 2, MSL members were defined and stored in an internal, hexadecimal representation that could neither be read nor edited outside of the Cross System Product library.

External source format provides the following capabilities to make the job of application development easier:

- Accepts application definitions into CSP/AD that were developed using non-CSP/AD design tools
- Lets you export CSP/AD definitions to other systems
- Provides readable application definitions within CSP/AD
- Provides an external format of CSP/AD definitions that can be used for centralized control and archiving
- Lets you access CSP/AD source code for analysis.

Figure 1-1 shows an overview of the CSP/AD application definition process using the external source format.

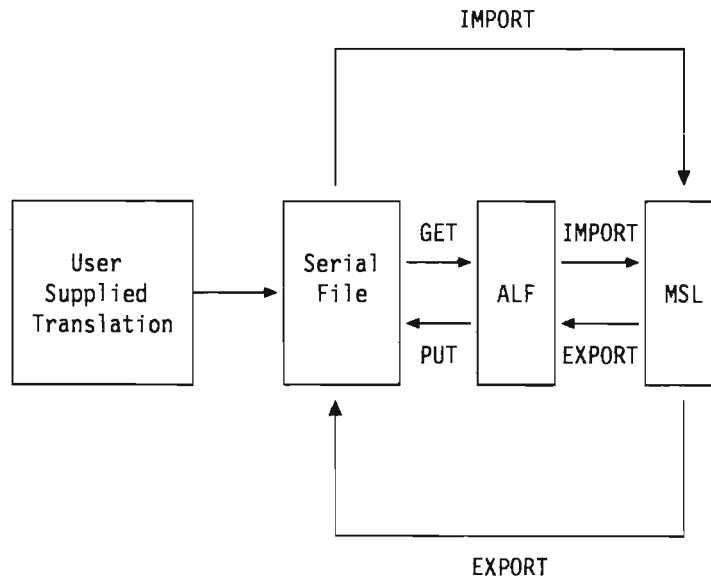


Figure 1-1. External Source Format

After you define an application with a non-CSP/AD design tool, you must translate the definition into external source format. CSP/AD does not support this translation; you have to put your definition into the external source format syntax, or use a product that is designed to translate applications into external source format.

The external source format feature works within CSP/AD as a formatting option specified when exporting MSL members. See Appendix A, "External Source Format Functions with CSP/AD and CSP/AE Commands," for information on using the external source format with the Cross System Product set.

External Source Format Tags

The external source format uses a set of mark-up language elements, called tags, to define each of the members found in an MSL. These tags identify the type of member and the attributes and values assigned to that member. A unique tag delimits each member type.

Figure 1-2 shows a sample of the tag syntax:

```
:tagname
      keyword = value.
      tag-content
:etagname.
```

Figure 1-2. Sample Tag Syntax

The *tagname* specifies the name of the tag. Tag names begin with the start tag open delimiter, the colon, which must be in column 1. Tags can be specified in uppercase and lowercase. Examples of tag names used in external source format are: MAP, ITEM, VFIELD, and PSB. It is often necessary to mark the end of an element explicitly. To do so, you mark the end with the same tag you began the element with, but you precede it with the end tag open delimiter, colon e (:e). Both start and end tags close with the tag close delimiter, the period.

Note: You must use the tag close delimiter only when you have text following the tag. However, it is a good practice to always include it.

An external source format tag in column 1 marks the end of the preceding tag and the beginning of the next tag.

The second item in the example, *keyword = value*, is an attribute of the tag. In this example, *keyword* is the name of the attribute and *value* is the variable attribute data. In some cases, the attribute value can be a list of numbers or alphabetic characters separated by one or more blanks.

You must enclose attribute values within single quotes or double quotes when the value contains a blank. If you need quotes inside the value, you must use either two single quotes or two double quotes, as in the following example:

```
:tagname keyword='value containing blanks, 'single quoted text,' and
"double quoted text."'
```

When an attribute value is a number that contains a decimal point, place at least one digit to the right of the decimal point to distinguish it from the tag close delimiter.

Attributes should follow the tag on the same line or on the following lines. Multiple attributes can appear on the same line, or an attribute and its value may span multiple lines. Attributes can be specified in lowercase and uppercase. You can specify attributes only for the tags you specify.

The *tag-content* is the text associated with that tag. The tag-content, which is subject to national language translation, follows the period. The format shown in Figure 1-3 on page 1-5 is used throughout this manual to describe the tag layouts.

The following information is provided for each tag.

Name

The name of the tag.

Description

A short description including all the required and optional tags.

Syntax

The tag syntax always appears in the left hand column. The tag being defined is always listed in bold typeface.

Note: You must specify tags in the order that they are listed under "Syntax." However, you do not have to specify the attributes in the order that they are listed under "Attributes."

Attribute Table

The table to the right of the syntax lists all the required and optional attributes and values for the current tag. The tables list attributes in uppercase letters. You can specify attributes in external source format in uppercase and lowercase. These tables have three columns:

Attributes Lists all possible attributes.

Values Lists all possible attribute values.

The tables use several styles of highlighting to make the values more readable:

- *Italic* — Italic typeface indicates a value name. When you specify this attribute, you give this value a quantity particular to the MSL member you are defining.
- **Bold** — Bold typeface indicates a default value. If you do not specify this attribute, CSP/AD automatically uses this value.
- **CAPS** — Words in all capital letters indicate the choices you can specify for this attribute.

Uses Lists possible uses.

Brackets ([]) surround attributes and values that are optional for the current tag. Vertical bars (|) separate value options that are valid for an attribute. Braces ({}) surround attribute value options that are mutually exclusive (you can specify only one of them).

Some tags do not require keyword attributes. Tags such as PROL, JOINCON, and STMTS have value content only. The syntax tables for these tags have only two columns, **Value** and **Usage**.

Attribute Description

A brief attribute description follows the syntax. This description contains two kinds of information:

- Quick reference
- Usage.

Attribute Quick Reference: The attribute quick reference information appears inside labeled boxes. The boxes list all the attribute values, using the same highlighting that appears in the attribute tables:

- *Italic* indicates a value name.
- **Bold** indicates a default value.
- CAPS indicate a value choice.

Attribute Usage: A description of how to use an attribute follows the quick reference. In some cases, you might require a more detailed explanation of how to use the attribute. In these cases, refer to the *CSP/AD Reference* manual or the *CSP/AD Developing Applications* manual.

Format Example

Variable Field Definition

The VFIELD tag specifies information about a variable field on the map. The VFIELD, MAPEDITS, MESSAGES, and VATTR tags are a set of tags that describe a single variable field and must be repeated for each variable field on the map. The MAPEDITS, MESSAGES, and VATTR tags are always optional. The EVFIELD tag closes the definition and is required for each VFIELD.

Syntax	Attributes		
:map			
⋮			
:vfield	:VFIELD		
⋮	Attributes	Values	Uses
:mapedits	BYTES =	<i>field length in bytes</i>	Specifies the number of positions that the variable field occupies
⋮	COLUMN =	<i>column number</i>	Specifies the column of the variable field marker
:messages	(DECIMALS =	<i>0[decimal places]</i>	Specifies the number of positions to the right of a decimal point in numeric items
⋮	(DESC =	<i>'field description'</i>	Describes what the field represents
:vattr	(EDITORDR =	<i>number</i>	Specifies the map edit sequence
⋮	(INDEX =	<i>index value</i>	Specifies the index value when the field is an array
:evfield	(NAME =	<i>field name</i>	Identifies a map variable field
⋮	ROW =	<i>row number</i>	Specifies the row of the variable field marker
:emap	(TYPE =	CHA DBCS MIX NUM	Specifies the data type
	[<i>initial value</i>]		Specifies the initial value of the field
	:EVFIELD[.]		

BYTES -	
<i>field length in bytes</i>	Specifies the number of positions that the variable field occupies

Decimal places can be specified only for numeric data. The maximum number of decimal positions is 18 or the number of digits defined for the field, whichever is smaller.

BYTES does not include the field marker.

DESC -	
<i>'field description'</i>	Describes what the variable field represents

COLUMN -	
<i>column number</i>	Specifies the column of the variable field marker

DESC is a text description of what the variable field represents. The text can be up to 30 characters.

DECIMALS -	
0	Specifies 0 decimal places (default value)
<i>decimal places</i>	Specifies a number of decimal places

EDITORDR -	
<i>number</i>	Specifies the map edit sequence

EDITORDR allows you to modify the editing sequence in which variable fields on the map are checked on input to an application.

DECIMALS is the number of positions to the right of an implied decimal point in numeric items.

Figure 1-3. Variable Field Definition, Format Example



Chapter 2. Application Structures

This chapter describes the external source format tags and attributes that define an application structure in CSP/AD.

Application Definition

The APPL tag defines attributes for individual applications. The APPL tag defines a CSP/AD application that consists of various CSP/AD objects, its parameter list, the type of program, and text describing the application. The MAINPRC, TABREC, CALLPARM, PROL, GENOPTS, GENTABLE, TARGSYS, GENFILE, and HTFFILE tags further define aspects of an application and can have attributes. The EAPPL tag closes the definition and is required.

Syntax	Attributes		
:appl	:APPL		
:			
:mainprc	Attributes	Values	Uses
:	[BYPKEY =	<i>number list</i>	Specifies PF keys used to bypass map edits and map edit groups
:emainprc	[DATE =	<i>'modification date'</i>	Specifies the date the application member was last modified
:	[FIRSTMAP =	<i>map name</i>	Specifies the initial map to be shown prior to running the first process
:tabrec	[HELPGRP =	<i>help map group name</i>	Specifies the map group that contains user-defined help maps
:	[HELPKEY =	<i>number</i>	Specifies a PF key to use for requesting help
:callparm	[IMPLICIT =	<i>Y N</i>	Specifies whether generation and test are to create implicit data items
:	[MAPGROUP =	<i>main map group name</i>	Specifies the mapgroup that contains the maps to be specified as process objects or as received parameters in the called parameter list
:prol	[MSGFILE =	<i>file name</i>	Specifies a user-created message file
:	NAME =	<i>name</i>	Identifies the application being developed
:eprol	[PFEQUATE =	<i>Y N</i>	Makes pressing PF13-24 equal to pressing PF1-12
:	[PSB =	<i>PSB name</i>	Specifies a program specification block definition in the MSL
:genopts	[TIME =	<i>'modification time'</i>	Specifies the time the application member was last modified
:	[TYPE =	MAIN MAINBATCH CALLED CALLBATCH	Defines the method of processing to be used for an application
:gentable	[WORKSTOR =	<i>record name</i>	Specifies a record with working storage organization
:targsys	[.]		
:	:EAPPL[.]		
:genfile			
:			
:htffile			
:			
:eappl			

BYPKEY =
number list Specifies up to five PF keys that allow the user to bypass map edits and map edit groups

Specify BYPKEY keys as integers from 1 to 24. Separate multiple keys with blanks. No default bypass edit PF keys are designated.

Note: Specifying the bypass edit PF keys for a map overrides the application specification bypass PF keys for that map.

DATE =
'*modification date*' Specifies the date the application member was last modified

DATE format is mm/dd/yy. If the external source format file was produced by CSP/AD export, this date shows when the application member was last modified in the MSL. The date is ignored on import.

FIRSTMAP =
map name Specifies the initial map to be shown prior to running the first process.

FIRSTMAP is the initial map on which you enter data prior to the execution of the first process. FIRSTMAP is valid only when the application is a main transaction.

HELPGRP =
help map group name Specifies the map group that contains the user-defined help maps

HELPGRP can be the same map group as the one specified in the map group attribute. If so, the help map group attribute does not need to be specified.

HELPKEY =
number Specifies a PF key to use for requesting help from application maps

Specify HELPKEY as an integer ranging from 1 to 24.

IMPLICIT =
N Specifies that all processing involved in creating implicits is bypassed
Y Creates implicits for all unqualified data items used in the application that are not defined in any map, record, or table used by the application (default value)

IMPLICIT specifies whether implicit data item definitions are created. If IMPLICIT=N, test and generation issue error messages for any undefined items.

MAPGROUP =
main map group name Specifies the mapgroup that contains the maps to be specified as process objects or as received parameters in the called parameter list

MAPGROUP is the name of the map group that contains the maps to be specified as process objects, as received parameters in the called parameter list, or as the first map for the application. A called application can use a different map group than the calling application unless a map is a parameter passed to the called application. In this case, the same map group must be used.

MSGFILE =
file name Specifies a user-created message file

MSGFILE has a maximum length of 4 characters. The first three characters identify the data set that contains the message file. A message header always precedes the message text; the first four characters of this message header are the same as the message file name.

NAME =	
<i>name</i>	Identifies the application being developed

NAME identifies the application being developed. It consists of 1 to 7 characters and must meet the following conventions:

- The first character must be alphabetic (A-Z).
- The remaining characters must be alphanumeric (A-Z, 0-9).
- The name cannot contain blanks or have an EZE prefix.

PFEQUATE =	
Y	Makes PF13-24 equal to PF1-12 (default value)
N	Makes PF13-24 not equal to PF1-12

PFEQUATE makes pressing PF13-24 equal to PF1-12. If PFEQUATE = Y, you can code PF key checks into the application as if there are only 12 PF keys. Statements checking for a single PF key, such as PF3, are treated as true if either PF3 or PF15 is pressed.

PSB =	
<i>PSB name</i>	Specifies a program specification block definition in the MSL

PSB is used to generate default DL/I calls within the application.

TIME =	
<i>'modification time'</i>	Specifies the time the application member was last modified

TIME format is hh:mm:ss. If the external source format file was produced by CSP/AD export, this time shows when the member was last modified in the MSL. The time is ignored on import.

TYPE =	
MAIN	Specifies main transaction processing (default value)
MAINBATCH	Specifies main batch processing
CALLED	Specifies called transaction processing
CALLBATCH	Specifies called batch processing

TYPE defines the method of processing to be used for an application.

MAIN: Specify this for an application that interacts with a user at a display device. The application is started via a transfer from the system, a non-CSP/AD program, or a Cross System Product application. A block of working storage data can be passed to the application upon transfer from another non-CSP/AD program or Cross System Product application.

MAINBATCH: Specify this for an application that processes without interacting with a user at a terminal. The application is started via a transfer from the system, a non-CSP/AD program, or a Cross System Product application. A block of working storage data can be passed to the application on transfer from another non-CSP/AD program or Cross System Product application.

CALLED: Specify this for an application that is called by and returns to another Cross System Product application or non-CSP/AD program. The application interacts with a user at a display device. Parameters can be passed between the calling and the called Cross System Product applications or non-CSP/AD programs.

CALLBATCH: Specify this for an application that is called by and returns to another Cross System Product application or non-CSP/AD program. The application does not interact with a user at a display device. Parameters can be passed between the calling and called Cross System Product applications or non-CSP/AD programs.

WORKSTOR =

<i>record name</i>	Specifies a record with working storage organization
--------------------	--

WORKSTOR defines a temporary data area, in addition to the defined maps and records, that the application might need for processing. You can specify only one record, but you can specify additional working storage records in the Table and Additional Records List.



Main Process Definition

The MAINPRC tag is an optional tag that, with its attributes, further describes the APPL and EAPPL tag set. If repeated, the order that the processes are named determines the order that they appear in the main process list. The EMAINPRC tag is required and closes each specification of MAINPRC.

Syntax	Attributes		
:appl	:MAINPRC		
:	Attributes	Values	Uses
:mainprc	NAME =	<i>name</i>	Identifies a process member
:	[. <i>flow statements</i>]		Specifies the flow statements associated with a process
:emainprc	:EMAINPRC[.]		
:			
:eappl			

NAME =	
<i>name</i>	Identifies a process member

flow statements	
<i>flow statements</i>	Specifies the flow statements, if any, associated with a process

NAME consists of 1 to 18 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The name cannot contain blanks or have an EZE prefix.

The name can be a DBCS name up to 8 DBCS characters long with no embedded blanks.

Flow statements can consist of multiple lines of text. If a statement line is longer than 71 characters, place a continuation character in column 72. Any character in column 72 is a continuation character. The continuation character causes concatenation of the two lines during import. The two lines concatenate with no blanks between the last character on the first line and the first character on the second line. The concatenated line length can never exceed 73 bytes.

Table and Additional Records List Definition

The TABREC tag specifies the following:

- Each table used in the application, including edit tables for data items on a map. The list of tables is used to verify references to tables by processing statements and to assure that the tables are available when the application runs.
- Each record used as an additional working storage area, an argument passed on a call, or a record redefinition referred to in the application.

This optional tag with its attributes further describes the APPL and EAPPL tag set.

Syntax	Attributes												
:appl : :tabrec : :eappl	<p>:TABREC</p> <table border="1"> <thead> <tr> <th>Attributes</th> <th>Values</th> <th>Uses</th> </tr> </thead> <tbody> <tr> <td>NAME=</td> <td><i>name</i></td> <td>Specifies the name of the additional table or record</td> </tr> <tr> <td>TYPE=</td> <td>RECORD TABLE</td> <td>Specifies the type of member</td> </tr> <tr> <td colspan="3">[.]</td> </tr> </tbody> </table>	Attributes	Values	Uses	NAME=	<i>name</i>	Specifies the name of the additional table or record	TYPE=	RECORD TABLE	Specifies the type of member	[.]		
Attributes	Values	Uses											
NAME=	<i>name</i>	Specifies the name of the additional table or record											
TYPE=	RECORD TABLE	Specifies the type of member											
[.]													

NAME =

name Specifies the name of the additional table or record

The record name can be a DBCS name up to 8 DBCS characters long with no embedded blanks.

NAME depends on the TYPE specification. TYPE can be one of the following values:

TABLE NAME is the name of the table member in the MSL. It consists of 1 to 7 characters and must meet the following conventions:

RECORD NAME is the name of the record member in the MSL. It consists of 1 to 18 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The record name cannot contain blanks or have an EZE prefix.

- The first character must be alphabetic (A-Z).
- The remaining characters must be alphanumeric (A-Z, 0-9).
- The table name cannot contain blanks or have an EZE prefix.

Table names cannot end in 0.

TYPE =

RECORD	Specifies that the additional member is a record
TABLE	Specifies that the additional member is a table

Parameter Specification

The CALLPARM tag specifies the name and type of each parameter received by a called application. This optional tag with its attributes further describes the APPL and EAPPL tag set.

The maximum number of parameters is 30. Parameters can be maps, records, or data items. List the parameters in the same order as the arguments are listed in the application's CALL statement. The number of parameters must equal the number of arguments.

The parameter definitions must be the same as, or compatible with, the definitions of the call arguments. If data types are not compatible or lengths are not the same, errors might occur during execution. A data item parameter cannot be a data item in a record, table, or map used by the called application. You can define the data item using data item definition.

Syntax	Attributes	Values	Uses
:appl : :callparm : :eappl	:CALLPARM NAME= [TYPE= [.]	<i>name</i> RECORD ITEM MAP]	Specifies the name of the parameter Specifies the type of parameter

NAME -

name Specifies the name of the parameter

NAME depends on the TYPE specification. TYPE can be one of the following values:

RECORD NAME is the name of the record member in the MSL. It consists of 1 to 18 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The record name cannot contain blanks or have an EZE prefix.

The record name can be a DBCS name up to 8 DBCS characters long with no embedded blanks.

ITEM NAME is a unique identification for a data item. It consists of 1 to 32 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The item name cannot contain blanks or have an EZE prefix.

The item name can be a DBCS name up to 15 DBCS characters long with no imbedded blanks.

Note: EZEDLPSB can be specified as an item name.

MAP NAME is the name of the map member in the MSL. It consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters (A-Z, 0-9, \$, #, @).
- The map name cannot contain blanks or have an EZE prefix.

TYPE =

RECORD	Specifies that the parameter is a record
ITEM	Specifies that the parameter is a data item
MAP	Specifies that the parameter is a map

TYPE must be RECORD, ITEM, or MAP. Although TYPE is not required, the application is not generated unless you specify the type through external source format or through on-line application definition.

Application Prolog Definition

The PROL tag is a text description of a defined application. This optional tag further describes the APPL and EAPPL tag set. The EPROL tag closes the description, and if you use the PROL tag, the EPROL tag is required.

Syntax	Usage
:appl :	:PROL
:prol :	Value
:eprol :	Usage
:eappl	[. <i>prolog lines</i>] Describes the application :EPROL[.]

Only the MSL size and machine size limit the number of lines that you can specify. Each line can contain up to 60 characters. You can use both uppercase and lowercase in the prolog text. The

text is saved as entered. When the text is longer than 60 characters, it is split into two lines during import.

Application Generation Option Specification

The GENOPTS tag specifies the saved generation options associated with the application. This optional tag with its attributes further describes the APPL and EAPPL tag set.

The external source format makes it possible to delete the saved generation options in an application. To do so, complete the following steps:

1. Export the application using the external source format.
2. Delete the GENOPTS, GENTABLE, TARGSYS, GENFILE, and HTFFILE tags from the external source format file.
3. Import the application into the MSL specifying the REPLACE option.

Syntax	Attributes																																															
<pre> :appl : :genopts : :eappl </pre>	<p>:GENOPTS</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Attributes</th> <th style="text-align: left;">Values</th> <th style="text-align: left;">Uses</th> </tr> </thead> <tbody> <tr> <td>[ANSIGEN =</td> <td>Y N]</td> <td>Specifies whether the ANSI static module will be generated</td> </tr> <tr> <td>[ANSIMOD =</td> <td><i>ANSI static module name</i>]</td> <td>Specifies the ANSI static module</td> </tr> <tr> <td>[CLIST =</td> <td>Y N]</td> <td>Specifies whether CLISTs are to be produced when the target system is DPPX</td> </tr> <tr> <td>[CREATREF =</td> <td>Y N]</td> <td>Specifies whether to create a reference file for use with the application during execution</td> </tr> <tr> <td>[DB2GEN =</td> <td>Y N]</td> <td>Specifies whether the DB2 static module will be generated</td> </tr> <tr> <td>[DB2MOD =</td> <td><i>DB2 module name</i>]</td> <td>Specifies the static assembler module generated for use with this application</td> </tr> <tr> <td>[DDSGEN =</td> <td>Y N]</td> <td>Specifies whether the DDS file is to be generated when generating for OS/400*</td> </tr> <tr> <td>[EXECMODE =</td> <td>NONSEGMENTED SEGMENTED EITHER SINGLESEG]</td> <td>Specifies the execution mode</td> </tr> <tr> <td>[FOLD =</td> <td>Y N]</td> <td>Specifies whether lowercase alphabetic characters in the listing are folded (converted) to uppercase</td> </tr> <tr> <td>[GENGRP1 =</td> <td>Y N]</td> <td>Specifies whether the map group named in MAPGRP1 is to be generated</td> </tr> <tr> <td>[GENGRP2 =</td> <td>Y N]</td> <td>Specifies whether the map group named in MAPGRP2 is to be generated</td> </tr> <tr> <td>[LOADLIB =</td> <td><i>name</i>]</td> <td>Specifies the library to which the generated modules are written</td> </tr> <tr> <td>[MAPGRP1 =</td> <td><i>map group name</i>]</td> <td>Specifies the name of a map group associated with the application</td> </tr> <tr> <td>[MAPGRP2 =</td> <td><i>map group name</i>]</td> <td>Specifies the name of a map group associated with the application</td> </tr> </tbody> </table>			Attributes	Values	Uses	[ANSIGEN =	Y N]	Specifies whether the ANSI static module will be generated	[ANSIMOD =	<i>ANSI static module name</i>]	Specifies the ANSI static module	[CLIST =	Y N]	Specifies whether CLISTs are to be produced when the target system is DPPX	[CREATREF =	Y N]	Specifies whether to create a reference file for use with the application during execution	[DB2GEN =	Y N]	Specifies whether the DB2 static module will be generated	[DB2MOD =	<i>DB2 module name</i>]	Specifies the static assembler module generated for use with this application	[DDSGEN =	Y N]	Specifies whether the DDS file is to be generated when generating for OS/400*	[EXECMODE =	NONSEGMENTED SEGMENTED EITHER SINGLESEG]	Specifies the execution mode	[FOLD =	Y N]	Specifies whether lowercase alphabetic characters in the listing are folded (converted) to uppercase	[GENGRP1 =	Y N]	Specifies whether the map group named in MAPGRP1 is to be generated	[GENGRP2 =	Y N]	Specifies whether the map group named in MAPGRP2 is to be generated	[LOADLIB =	<i>name</i>]	Specifies the library to which the generated modules are written	[MAPGRP1 =	<i>map group name</i>]	Specifies the name of a map group associated with the application	[MAPGRP2 =	<i>map group name</i>]	Specifies the name of a map group associated with the application
Attributes	Values	Uses																																														
[ANSIGEN =	Y N]	Specifies whether the ANSI static module will be generated																																														
[ANSIMOD =	<i>ANSI static module name</i>]	Specifies the ANSI static module																																														
[CLIST =	Y N]	Specifies whether CLISTs are to be produced when the target system is DPPX																																														
[CREATREF =	Y N]	Specifies whether to create a reference file for use with the application during execution																																														
[DB2GEN =	Y N]	Specifies whether the DB2 static module will be generated																																														
[DB2MOD =	<i>DB2 module name</i>]	Specifies the static assembler module generated for use with this application																																														
[DDSGEN =	Y N]	Specifies whether the DDS file is to be generated when generating for OS/400*																																														
[EXECMODE =	NONSEGMENTED SEGMENTED EITHER SINGLESEG]	Specifies the execution mode																																														
[FOLD =	Y N]	Specifies whether lowercase alphabetic characters in the listing are folded (converted) to uppercase																																														
[GENGRP1 =	Y N]	Specifies whether the map group named in MAPGRP1 is to be generated																																														
[GENGRP2 =	Y N]	Specifies whether the map group named in MAPGRP2 is to be generated																																														
[LOADLIB =	<i>name</i>]	Specifies the library to which the generated modules are written																																														
[MAPGRP1 =	<i>map group name</i>]	Specifies the name of a map group associated with the application																																														
[MAPGRP2 =	<i>map group name</i>]	Specifies the name of a map group associated with the application																																														

* A trademark of IBM

:GENOPTS (continued)		
Attributes	Values	Uses
[PRINT =	Y N]	Specifies whether to print during generation
[SEGTRAN =	'transaction ID']	Specifies the transaction used for the second and subsequent segments of a main transaction application running in segmented execution mode
[SQLBLOCK =	Y N]	Specifies the BLOCK option on the SQL CREATE PROGRAM statement
[SQLGEN =	Y N]	Specifies whether the SQL/DS access module is to be generated
[SQLID =	'SQL user ID']	Identifies the SQL/DS user who is listed as the owner of the application access module being created
[SQLMOD =	SQL access module name]	Specifies the SQL/DS access module
[VALIDLOC =	Y N]	Specifies whether LOCAL data items are to be validated against GLOBAL data item definitions at generation
[VALIDSQL =	Y N]	Specifies whether SQL statements are to be validated and if host variable compatibility is to be checked
[.]		

ANSIGEN =	
Y	Specifies that the ANSI static module for the application is generated
N	Specifies that the ANSI static module for the application is not generated (default value)

CREATREF =	
Y	Specifies to create a reference file for use with the application during execution
N	Specifies not to create a reference file for use with the application during execution

ANSIMOD =	
<i>ANSI static module name</i>	Specifies the name of the ANSI static module

DB2GEN =	
Y	Specifies that the DB2 static module for the application is generated (default value)
N	Specifies that the DB2 static module for the application is not generated

Use this attribute to supply the name of the ANSI static module if the application has SQL processes and the target system supports the ANSI Standard SQL interface.

CLIST =	
Y	Specifies that CLISTs will be produced when the target system is DPPX (default value)
N	Specifies that CLISTs will not be produced when the target system is DPPX

DB2MOD =	
<i>DB2 module name</i>	Specifies the name of the static assembler module generated for use with this application

DB2MOD is valid only if the application has SQL processes and the target system supports DB2 as the database manager. The default name is the application name.

DDSGEN =	
Y	Specifies that the DDS file is generated when generating for OS/400 (default value)
N	Specifies that the DDS file is not generated when generating for OS/400

GENGRP1 =	
Y	Specifies that the map group named in MAPGRP1 is generated (default value)
N	Specifies that the map group named in MAPGRP1 is not generated

EXECMODE =	
NONSEGMENTED	Specifies that the application runs in conversational mode (default value)
SEGMENTED	Specifies that the application runs in segmented (DTMS) or pseudoconversational (CICS/VS) mode
EITHER	Specifies that the application can run in either segmented or nonsegmented mode (defaults to NONSEGMENTED)
SINGLESEG	Specifies that the application runs in single-segment mode and is valid for IMS/VS only

GENGRP2 =	
Y	Specifies that the map group named in MAPGRP2 is generated (default value)
N	Specifies that the map group named in MAPGRP2 is not generated

LOADLIB =	
<i>load library name</i>	Specifies the library to which the generated modules are written

EXECMODE defines the mode in which applications run in the transactional processing environments: IMS/VS, CICS/VS, and DPPX DTMS. Use the EZESEGM special function word in the application to dynamically change the execution mode.

NONSEGMENTED holds input/output locks, position, and main storage resources across each CONVERSE process option.

SEGMENTED saves application information across each CONVERSE process option. All main storage resources are freed while waiting for a terminal.

EITHER allows the user to specify whether the application is segmented or nonsegmented when starting the application.

SINGLESEG specifies that the application runs in single-segment mode. It is valid for IMS/VS only.

FOLD =	
Y	Specifies that lowercase characters in the listing are folded to uppercase characters before they are printed
N	Specifies that lowercase characters in the listing are not folded before they are printed (default value)

The load library is the application load file (ALF), a VSAM data set, used by CSP/AE during generation. The default is FZERSAM.

MAPGRP1 =	
<i>map group name</i>	Specifies a map group associated with the application

MAPGRP1 can be either the main map group or the help map group.

MAPGRP2 =	
<i>map group name</i>	Specifies a map group associated with the application

MAPGRP2 can be either the main map group or the help map group.

PRINT =	
Y	Specifies that a listing is to be printed during generation
N	Specifies that a listing is not printed during generation (default value)

SEGTRAN =

'*transaction ID*' Specifies the transaction name used for the second and subsequent segments of a main transaction application running in segmented execution mode

The default name is XSPS in CICS/VS and DZGNEXT in DPPX.

The application can override the segmented transaction name by moving a new name into EZSEGTR prior to the CONVERSE.

When receiving control from an application doing a DXFR, the application receiving control uses the segmented transaction name defined for the application that was initially invoked.

SQLBLOCK =

Y Specifies that the BLOCK option is used on the SQL CREATE PROGRAM statement
N Specifies that the BLOCK option is not used on the SQL CREATE PROGRAM statement (default value)

SQLGEN =

Y Specifies that the SQL access module for the application is generated (default value)
N Specifies that the SQL access module for the application is not generated

SQLID =

'*SQL user ID*' identifies the SQL/DS user who is listed as the owner of the application access module being created

SQLMOD =

SQL access module name
Specifies the name of the SQL/DS access module

SQLMOD is valid only if the application has SQL processes and the target system supports SQL/DS as the database manager. The default name is the application name.

VALIDLOC =

Y Specifies that LOCAL data items are validated against GLOBAL data item definitions at generation
N Specifies that LOCAL data items are not validated against GLOBAL data item definitions at generation

VALIDLOC specifies whether the preprocessor compares LOCAL data item definitions to GLOBAL data item definitions found in the MSL.

Data item definitions in maps are not validated because certain data item characteristics are supported on maps and field edits can change item length.

VALIDSQL =

Y Specifies that SQL statements will be validated and that host variable compatibility will be checked
N Validates nothing (default value)

VALIDSQL specifies that the preprocessor is to use the SQL dynamic PREPARE interface to check the syntax of the SQL statement defined for the application. In addition, VALIDSQL specifies that compatibility checking between host variables in the INTO clause of the SELECT statements and the columns selected by the SQL statement is to be performed.

This attribute is valid only if the application accesses a relational table.

Table Generation Specification

The GENTABLE tag specifies the saved generation options for tables used in the application. This optional tag with its attributes further describes the APPL and EAPPL tag set.

Syntax	Attributes		
:appl : :gentable : :eappl	:GENTABLE		
	Attributes	Values	Uses
	[KEEP =	Y N]	Specifies whether the table is to be deleted at the end of application execution
	NAME =	<i>table name</i>	Specifies the table member in the MSL
	[TABLEGEN =	Y N]	Specifies whether the table is to be generated with the application
	[.]		

KEEP =

Y	Specifies that the table is deleted at the end of application execution (default value)
N	Specifies that the table is deleted at the end of the main process that caused the table to be loaded

- The name cannot contain blanks or have an EZE prefix.
- Table names cannot end in 0.

NAME =

name Specifies the table member in the MSL

TABLEGEN =

Y	Specifies that the table is generated with the application (default value)
N	Specifies that the table is not generated with the application

NAME consists of 1 to 7 characters and must meet the following conventions:

- The first character must be alphabetic (A-Z).
- The remaining characters must be alphanumeric (A-Z, 0-9).

Target System Specification

The TARGSYS tag specifies the saved generation target system on which the application is to be executed. This optional tag further describes the APPL and EAPPL tag set.

Syntax	Attributes	Values	Uses
:appl : :targsys : :eappl	:TARGSYS Attributes SYSTEM =	system	Specifies the system on which the application is to be executed; the default is the system on which you are importing
	[.]		

When you import members using the CSP/AD Programmable Workstation Feature, TARGSYS is a required attribute. When importing on all host systems, TARGSYS is optional. If you do not specify TARGSYS, the default is the host system on which you are importing.

System =	
CSP/AE Systems	
CICS/DOS	Specifies that the application will be generated to run in CICS/DOS or in VSE Batch
CICS/OS	Specifies that the application will be generated to run in CICS/OS or in MVS Batch
CMS*	Specifies that the application will be generated to run in CMS
TSO	Specifies that the application will be generated to run in TSO
OS/400	Specifies that the application will be generated to run in OS/400
ANY	Specifies that the application will be generated to run on any CSP/AE system except DPPX
DPPX	Specifies that the application will be generated to run in DPPX

System =	
CSP/370 Runtime Services Systems	
IMS/VS	Specifies that the application will be generated to run in IMS/VS
IMSBMP	Specifies that the application will be generated to run as an IMS BMP program
MVSBATCH	Specifies that the application will be generated to run in MVS Batch

* A trademark of IBM

Internal Resource Name Specification

The GENFILE tag specifies the saved generation internal names of resources used by an application. This optional tag with its attributes further describes the APPL and EAPPL tag set.

You can specify the GENFILE tag multiple times.

Syntax	Attributes		
:appl : :genfile : :eappl	:GENFILE		
	Attributes	Values	Uses
	FILENAME =	<i>file name</i>	Specifies a file name for a file used by the application
	[FILETYPE =	{TRANSIENT VSAMCICS} {CMS VSAMCMS} {DATABASE FILE} {MSGQ MMSGQ} SEQ VSAM GSAM}}	Specifies the type of file that was named by FILENAME and SYSNAME
	[PCBNO =	<i>pcb number</i>]	Specifies the number of the PCB in the PSB member
	[SYSNAME =	' <i>system resource name</i> ']	Specifies the real name of the resource specified in FILENAME
	[SYSTEM =	<i>system</i>]	Specifies the system on which the specified file resource name is to be used
	[.]		

FILENAME =	
<i>file name</i>	Specifies a file name for a file used by the application

Specify FILENAME for serial, relative, and indexed files.

FILENAME consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic (A-Z).
- The remaining characters must be alphanumeric (A-Z, 0-9).
- The file name cannot contain blanks or have an EZE prefix except EZEPRINT.

A message file name cannot be used as a file name.

Records sharing the same file name are associated with the same physical file.

EZEPRINT is the internal file name for printer maps. This name cannot be changed, but you can give it an external association with the SYSNAME attribute.

FILETYPE =

TRANSIENT	Specifies that the file is a transient data queue on CICS/VS
VSMCICS	Specifies that the file is a VSAM file on CICS/VS
CMS	Specifies that the file is a CMS file on VM/SP
VSMCMS	Specifies that the file is a VSAM file on VM/SP
DATABASE	Specifies that the file is a database on DPPX
FILE	Specifies that a resource is a file on DPPX
SMSGQ	Specifies that the file is a single-segment message queue in IMS/VS or IMS BMP (SMSGQ is the default for IMS/VS)
MMSGQ	Specifies that the file is a multisegment message queue in IMS/VS or IMS BMP
SEQ	Specifies that the file is an OS sequential file in IMS BMP or MVS Batch (SEQ is the default for IMS BMP and MVS Batch)
VSAM	Specifies that the file is a VSAM ESDS data set in IMS BMP or MVS Batch
GSAM	Specifies that the file is a GSAM database in IMS BMP or MVS Batch

The default FILETYPE value depends on the target system specified. The MMSGQ and VSAM filetypes cannot be specified for EZEPRINT.

PCBNO =

<i>pcbno</i>	Specifies the number of the PCB in the PSB member
--------------	---

PCBNO is the number of the Program Communication Block (PCB) in the PSB member that represents the message queue or GSAM database for the file. The number is an integer from zero to the number of PCBs in the PSB. PCBNO is valid only if the file type is SMSGQ, MMSGQ, or GSAM. If you do not specify a value for PCBNO, the default number is determined during application generation from the PSB definition and from the use of the file in the application, as follows:

File	Default
SMSGQ input	0
MMSGQ input	0

SMSGQ output	1
MMSGQ output	1
GSAM	First GSAM PCB

SYSNAME =

<i>'system resource name'</i>	Specifies the real name of the resource specified in FILENAME. SYSNAME defaults to the name specified for FILENAME.
-------------------------------	---

The default value of SYSNAME is the value specified for FILENAME, except when the target system is CICS and the FILENAME is EZEPRINT. In that case, the default is EZEPR. For DPPX, the default name is SYSNET01.PRT1.

For SMSGQ and MMSGQ files, the name is a 1- to 8-character IMS terminal or transaction identifier. For SEQ, VSAM, or GSAM files, the name is an MVS data set name.

System =**CSP/AE Systems**

CICS/DOS	Specifies that the application will be generated to run in CICS/DOS or in VSE Batch
CICS/OS	Specifies that the application will be generated to run in CICS/OS or in MVS Batch
CMS	Specifies that the application will be generated to run in CMS
TSO	Specifies that the application will be generated to run in TSO
OS/400	Specifies that the application will be generated to run in OS/400
ANY	Specifies that the application will be generated to run on any CSP/AE system except DPPX
DPPX	Specifies that the application will be generated to run in DPPX

System =**CSP/370 Runtime Services Systems**

IMS/VS	Specifies that the application will be generated to run in IMS/VS
IMSBMP	Specifies that the application will be generated to run as an IMS BMP program
MVSBATCH	Specifies that the application will be generated to run in MVS Batch

| The default value of SYSTEM is the value specified | the default is the system on which you are
| on the TARGSYS tag. If TARGSYS is not specified, | importing.



DPPX File Information Specification

The HTFFILE tag specifies the saved generation DPPX file information. This optional tag with its attributes further describes the APPL and EAPPL tag set.

You can specify the HTFFILE tag multiple times, once for each DPPX file.

Syntax	Attributes																	
<pre>:appl : :htffile : :eappl</pre>	<pre>:HTFFILE</pre> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Attributes</th> <th style="text-align: left;">Values</th> <th style="text-align: left;">Uses</th> </tr> </thead> <tbody> <tr> <td>[ASYNC=</td> <td>Y N]</td> <td>Specifies whether asynchronous or synchronous record ADD processing is performed</td> </tr> <tr> <td>HOSTTRAN=</td> <td><i>host transaction file name</i></td> <td>Specifies the host transaction name that is used to communicate with the host system for a remote file</td> </tr> <tr> <td>[HTFERITM=</td> <td><i>host transaction facility error message item name</i>]</td> <td>Specifies the data item (error message item) used to store any text data accompanying a host transaction reply that carries a failure completion code</td> </tr> <tr> <td>[HTFWKITM=</td> <td><i>host transaction facility work item name</i>]</td> <td>Specifies the data item (transaction work item) used as the source of the text to be sent with the transaction ID to the host system</td> </tr> </tbody> </table>			Attributes	Values	Uses	[ASYNC=	Y N]	Specifies whether asynchronous or synchronous record ADD processing is performed	HOSTTRAN=	<i>host transaction file name</i>	Specifies the host transaction name that is used to communicate with the host system for a remote file	[HTFERITM=	<i>host transaction facility error message item name</i>]	Specifies the data item (error message item) used to store any text data accompanying a host transaction reply that carries a failure completion code	[HTFWKITM=	<i>host transaction facility work item name</i>]	Specifies the data item (transaction work item) used as the source of the text to be sent with the transaction ID to the host system
Attributes	Values	Uses																
[ASYNC=	Y N]	Specifies whether asynchronous or synchronous record ADD processing is performed																
HOSTTRAN=	<i>host transaction file name</i>	Specifies the host transaction name that is used to communicate with the host system for a remote file																
[HTFERITM=	<i>host transaction facility error message item name</i>]	Specifies the data item (error message item) used to store any text data accompanying a host transaction reply that carries a failure completion code																
[HTFWKITM=	<i>host transaction facility work item name</i>]	Specifies the data item (transaction work item) used as the source of the text to be sent with the transaction ID to the host system																
	[.]																	

ASYNC=	
Y	Specifies asynchronous record ADD processing is performed
N	Specifies synchronous record ADD processing is performed (default value)

HOSTTRAN=
<i>host transaction file name</i>
Specifies the host transaction name used by CSP/AD to communicate with the host system for a remote file

HTFERITM=
<i>host transaction facility error message item name</i>
Specifies the data item (error message item) used to store any text data accompanying a host transaction reply that carries a failure completion code

HTFERITM is required if ASYNC = N.

HTFERITM consists of 1 to 32 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The host transaction facility error message item name cannot contain blanks or have an EZE prefix.

HTFWKITM =

host transaction facility work item name

Specifies the data item (transaction work item) used as the source of the text to be sent with the transaction ID to the host system

HTFWKITM consists of 1 to 32 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The host transaction facility work item name cannot contain blanks or have an EZE prefix.

Application Definition Syntax Example

```
:appl      name      = xxxxxxxx
           date      = 'mm/dd/yy'  time    = 'hh:mm:ss'  type    = xxxxxxxxx
           workstor  = xxxxxxxxxxxxxxxxxxxx  mapgroup = xxxxxx
           helpgrp   = xxxxxx              helpkey  = xx      msgfile = xxxx
           psb       = xxxxxxxx           pfequate = x      implicit = x
           bypkey    = xx xx xx xx xx     firstmap = xxxxxxxx.

:mainprc   name      = xxxxxxxxxxxxxxxxxxxx.
MOVE A TO B;
:emainprc.
:tabrec    name      = xxxxxxxxxxxxxxxxxxxx
           type      = xxxxxx.
:callparm  name      = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
           type      = xxxxxx.

:prol.
Each prolog line can be up to 60 characters long . IMPORT
will split lines longer than 60 characters.
:eprol.
:genopts   loadlib   = xxxxxxxx          creatref = x          print    = x
           fold      = x                  validsql = x
           execmode  = xxxxxxxxxxxx
           mapgrp1   = xxxxxx              mapgrp2  = xxxxxx
           gengrp1  = x                    gengrp2  = x
           db2mod    = xxxxxxxx           db2gen   = x
           sqlmod    = xxxxxxxx           sqlgen   = x
           ansimod   = xxxxxxxx           ansigen  = x
           segtran   = 'xxxxxxx'          ddsgen   = x
           sqlid     = 'xxxxxxx'          sqlblock = x
           clist     = x                    validloc = x.
:gentable  name      = xxxxxxxx          tablegen = x          keep    = x.
:targsys   system    = xxxxxxxx.
:genfile   file name = xxxxxxxx          system   = xxxxxxxx
           sysname   = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
           filetype  = xxxxxxxx
           pcbno     = xxx.
:htffile   hosttran  = xxxxxxxx          async    = x
           htferitm  = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
           htfwkitm  = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

:eappl.
```

Note: This example illustrates the syntax of all application definition tags and attributes. Actual definitions do not require or use all tags and attributes shown here.



Chapter 3. Process Structures

This chapter describes the external source format tags and attributes that define a process structure in CSP/AD.

Process Member Definition

The PROCESS tag defines attributes for individual process members. The BEFORE, AFTER, SQL, DLICALL, SSA, and QUAL tags further define aspects of a process and can have attributes. The EPROCESS tag closes the definition, and if you use the PROCESS tag, the EPROCESS tag is required.

Syntax	Attributes	Values	Uses
:process	:PROCESS		
:before	Attributes		
:ebefore	[DATE =	'modification date']	Specifies the date the process member was last modified
:after	[DESC =	'description']	Describes the process
:eafter	[ERRRTN =	routine name]	Specifies an error-handling routine
:sql	[EXECBLD =	Y N]	Specifies whether the SQL statements are to be built as static SQL statements or prepared as dynamic statements
:esql	[MODEL =	DELETE UPDATE NONE]	Specifies generation of a model SQL statement
:dlicall	NAME =	name	Identifies the process member
:ssa	[OBJECT =	object name]	Specifies a record or map accessed by OPTION
:qual	[OPTION =	option]	Specifies the input/output function in a process
:eprocess	[REFINE =	Y N]	Specifies the method of displaying logic for the process when definition is performed on a programmable workstation facility
	[SINGROW =	Y N]	Specifies whether a single-row SELECT is to be performed for an SQL INQUIRY
	[TIME =	'modification time']	Specifies the time the process member was last modified
	[UPDPROC =	update process name]	Specifies the name of an UPDATE or SETUPD process
	[.]		
	:EPROCESS[.]		

DATE =

'modification date' Specifies the date the process member was last modified

DATE format is mm/dd/yy. If the external source format file was produced by CSP/AD export, this date shows when the process member was last modified in the MSL. The date is ignored on import.

DESC =

<i>'description'</i>	Describes the process
----------------------	-----------------------

DESC is a text string from 1 to 30 characters that describes a process. A process description is optional; it does not affect execution.

ERRRTN =

<i>routine name</i>	Specifies an error-handling routine
---------------------	-------------------------------------

ERRRTN calls an error routine when an error occurs during execution of a process option that accesses a record. For a record option, ERRRTN can be a main process, a statement group, or the special function words EZERTN, EZEFLO, or EZECLAS. If no error routine is specified, an application ends when an error occurs and a message appears describing the error condition.

Error routines cannot be specified for processes with map process objects or for EXECUTE processes. DISPLAY or CONVERSE errors cause the application to end.

EXECBLD =

Y	Prepares and executes SQL statements as dynamic or extended dynamic statements
N	Uses the execution mode specified at generation or execution

EXECBLD can only be used with the following process options:

- INQUIRY
- SETINQ
- SETUPD
- SQLEXEC
- UPDATE.

MODEL =

DELETE	Specifies that a default DELETE SQL statement is to be generated
UPDATE	Specifies that a default UPDATE SQL statement is to be generated
NONE	Specifies that no model statement is to be generated

You can specify MODEL for SQLEXEC processes only.

NAME =

<i>name</i>	Identifies a process member
-------------	-----------------------------

NAME consists of 1 to 18 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The name cannot contain blanks or have an EZE prefix.

The name can be a DBCS name up to 8 DBCS characters long with no embedded blanks.

OBJECT =

<i>object name</i>	Specifies a record or map accessed by OPTION
--------------------	--

EXECUTE processes do not have an object, and SQLEXEC processes may or may not have an object. Other processes must have an object.

OPTION =

<i>option</i>	Specifies the input/output function in a process
---------------	--

OPTION can be specified as any one of the following process options.

ADD places a new record in a file or database. The program should initialize all fields in the record prior to executing the ADD option.

CLOSE closes a file, disconnects a printer, or releases any unprocessed rows in a set of SQL row records selected by UPDATE, SETUPD, or SETINQ.

CONVERSE displays a map at a terminal and waits for input to be entered by the user at the terminal.

DELETE removes a record from a file or database.

DISPLAY sends a map to a printer or to a terminal output buffer.

EXECUTE is not associated with an input/output operation. It can be used for special processing such as control of flow between processes, initialization, processing not to be repeated in an I/O process, error handling, and termination processing. EXECUTE is the default OPTION value.

INQUIRY reads a single record from a file or database.

REPLACE puts a changed record back into a file or database.

SCAN reads the next record in a file or database.

SCANBACK reads the previous record in an indexed file.

SETINQ selects a set of rows from a relational database for later retrieval with the SCAN process option. The object must be an SQL row record.

SETUPD selects a set of rows from a relational database for faster processing with the SCAN process option. The records selected can be replaced or deleted. The object must be an SQL row record.

SQLEXEC executes an SQL statement that you define by using the SQL tag.

UPDATE reads a record from a file or database with the implied intention of replacing or deleting the record.

REFINE =	
Y	Indicates that the refinement takes place on a single screen
N	Indicates that the refinement takes place on multiple screens

REFINE specifies the method of displaying logic for the process when process members are defined using the CSP/AD Programmable Workstation Feature. The attribute is preserved in the MSL, but

it has no effect on entering the process definition using CSP/AD on the host or on the execution of the generated application.

SINGROW =	
Y	Specifies a single-row SELECT is to be performed for an SQL INQUIRY
N	Specifies a single-row SELECT is not to be performed

SINGROW can be used only when an application is running in static mode (on DB2 systems) or in extended dynamic mode (on SQL/DS systems). The single row SELECT is not available when running CSP/AD SQL applications in dynamic mode.

Note: If the external source file is created for Version 3 Release 3 or later, the default is YES. If the external source file is created for Version 3 Release 2 Modification 2, the default is NO.

TIME =	
'modification time'	Specifies the time that the process member was last modified

TIME format is hh:mm:ss. If the external source format file was produced by CSP/AD export, this time shows when the process member was last modified in the MSL. The time is ignored on import.

UPDPROC =	
update process name	Specifies the name of an UPDATE or SETUPD process that selected the rows to be replaced by this process

UPDPROC can be specified for a REPLACE process.

Logic Definition Before Process Option

The BEFORE tag begins the set of statements that perform logic before the process option is executed. This optional tag further defines the PROCESS and EPROCESS tag set. The EBEFORE tag closes the definition, and if you use the BEFORE tag, the EBEFORE tag is required.

The BEFORE tag can appear only once within a process definition. The statement syntax is a normal CSP/AD statement syntax. Validation is for statement syntax only, not for MSL member usage. Statements are limited to one per line; however, a single statement can span multiple lines.

```

:
:before. before processing statement1
        before processing statement2
        :
        before processing statementn
:ebefore.
:

```

For an EXECUTE process, all statements specified on the BEFORE tag are executed before any statements on the AFTER tag.

Syntax	Usage	
:process		
:	:BEFORE	
:before	Value	Usage
:		
:ebefore	[.before processing statements]	Specifies statements that perform logic before the process option is executed
:		
:eprocess	:EBEFORE[.]	

Statements can consist of multiple lines of text. If a statement line is longer than 71 characters, place a continuation character in column 72. Any character in column 72 is a continuation character. The continuation character causes concatenation

of the two lines on import. The two lines concatenate with no blanks between the last character on the first line and the first character on the second line. The concatenated line length can never exceed 73 bytes.

Logic Definition After Process Option

The AFTER tag begins the set of statements that perform logic after the process option is executed. This optional tag further defines the PROCESS and EPROCESS tag set. The EAFTER tag closes the definition, and if you use the AFTER tag, the EAFTER tag is required.

The AFTER tag can appear only once within a process definition. Statement syntax is normal CSP/AD statement syntax. Validation is for statement syntax only, not for MSL member usage. Statements are limited to one per line; however, a single statement can span multiple lines.

```

:
:after. after processing statement1
        after processing statement2
        :
        after processing statementn
:eafter.
:

```

For an EXECUTE process, all statements specified on the AFTER tag are executed after any statements on the BEFORE tag.

Syntax	Usage
:process	
:	:AFTER
:after	Value
:	Usage
:eafter	[<i>after processing statements</i>]
:	Specifies statements that perform logic after the process option is executed
:eprocess	:EAFTER[.]

Statements can consist of multiple lines of text. If a statement line is longer than 71 characters, place a continuation character in column 72. Any character in column 72 is considered a continuation character. The continuation character

causes concatenation of the two lines on import. The two lines concatenate with no blanks between the last character on the first line and the first character on the second line. The concatenated line length can never exceed 73 bytes.

SQL Selection Condition Specification

The SQL tag specifies the SQL selection conditions that CSP/AD and CSP/AE use to retrieve information from the SQL database. This optional tag further defines the PROCESS and EPROCESS tag set. The ESQL tag closes the specification, and if you use the SQL tag, the ESQL tag is required.

The SQL tag is valid for the INQUIRY, SETINQ, SETUPD, SQLEXEC, and UPDATE process options. The SQL tag is not valid for the EXECUTE, CONVERSE, DISPLAY, and SCANBACK process options because they cannot have an SQL row record as the process object.

You can repeat the SQL tag for each SQL clause that you specify, but the tag can appear only once for each type of clause.

Syntax	Attributes		
:process			
:			
:sql	:SQL		
:	Attributes	Values	Uses
:esql	CLAUSE =	clause type	Specifies the type of clause described by this tag
:	[HOSTVAR =	'?' '@']	Specifies the character used within the clause text to indicate the beginning of a host variable name
:eprocess	[.clause text]		Specifies the clause for the SQL statement
	:ESQL[.]		

CLAUSE =	
SELECT	Specifies the SELECT clause
INTO	Specifies the INTO clause
SET	Specifies the SET clause
WHERE	Specifies the WHERE clause
ORDERBY	Specifies the ORDER BY clause
SQLEXEC	Specifies the SQLEXEC clause
VALUES	Specifies the VALUES clause
FORUPDATEOF	Specifies the FORUPDATEOF clause
INSERTCOLNAME	Specifies the column names for the INSERT INTO clause

UPDATE	SELECT, INTO, WHERE, FORUPDATEOF
SETUPD	SELECT, INTO, WHERE, FORUPDATEOF
REPLACE	SET
SQLEXEC	SQLEXEC
ADD	VALUES, INSERTCOLNAME

HOSTVAR =	
'?'	Specifies that the question mark (?) is used to begin host variable names
'@'	Specifies that the at sign (@) is used to begin host variable names

Certain CLAUSE specifications are valid for certain process options. The following table identifies the applicable specifications:

Process option	CLAUSE specification
INQUIRY	SELECT, INTO, WHERE, ORDERBY
SETINQ	SELECT, INTO, WHERE, ORDERBY

Export always uses the question mark (?) as the host variable name indicator so that the at sign (@) can be properly interpreted as a valid character in a name. Using the at sign (@) as the host variable name indicator, when variable names also contain the at sign (@), is ambiguous and can cause unpredictable results on Import.

clause text

clause text Specifies the clause for the SQL statement

The lines in the external source format file correspond to the lines on panel EZEM3M - Object Selection: SQL Statement Definition.

The clause text for the SELECT, INTO, SET, VALUES, INSERTCOLNAME, and FORUPDATEOF clauses do not include their respective keywords.

The clause text for a WHERE clause is the clause that includes the WHERE keyword. The WHERE clause can contain the ORDER BY clause if the ORDER BY clause was not placed on a separate line during application definition of the SQL statement. A WHERE clause without any clause text indicates that there is no WHERE clause in the SQL statement.

The clause text for an ORDER BY clause is the clause that includes the ORDER BY keywords. An ORDER BY clause without any clause text indicates that there is no ORDER BY clause in the SQL statement.

The clause text for an SQLEXEC clause is the entire clause including all keywords.

Within the clause text, SQL column names always begin with the exclamation point (!). When produced from CSP/AD Export, host variable names always begin with the question mark (?).

Statements can consist of multiple lines of text. If a statement line is longer than 71 characters, place a continuation character in column 72. Any character in column 72 is considered a continuation character. The continuation character causes concatenation of the two lines on import. The two lines concatenate with no blanks between the last character on the first line and the first character on the second line. The concatenated line length can never exceed 73 bytes.

DL/I Call Definition

The DLICALL tag begins the definition of calls to access data stored in DL/I databases. This optional tag with its attributes further describes the PROCESS and EPROCESS tag set.

Syntax	Attributes		
:process : :dlcall : :eprocess	:DLICALL		
	Attributes	Values	Uses
	DBDNAME =	<i>database name</i> <i>number</i>	Specifies the database in the application program specification block that is to be accessed by this DL/I call
	PSB =	<i>PSB name</i>	Specifies a program specification block used by this process
	[SCANPAR =	Y N]	Specifies whether the range of the SCAN is limited to the currently established parent chain
	[SCANUPD =	Y N]	Specifies whether a segment retrieved by a SCAN process can be replaced or deleted
	[.]		

DBDNAME =	
<i>database name</i>	Accesses the first program communication block in the application program specification block with this database name
<i>number</i>	Accesses this number program communication block in the application program specification block

DBDNAME identifies the database in the application program specification block that is to be accessed by this DL/I call.

PSB =	
<i>PSB name</i>	Specifies a program specification block used by this process

SCANPAR =	
Y	Specifies the SCAN range is limited to the current parent chain in the database hierarchy
N	Specifies the next segment of that type in the database is retrieved regardless of parentage (default value)

SCANPAR can be specified only for a DL/I call for a SCAN process.

SCANUPD =	
Y	Specifies that the application can replace or delete the segments retrieved by the SCAN
N	Specifies that the application cannot replace or delete the segments retrieved by the SCAN (default value)

SCANUPD can be specified only for a DL/I call for a SCAN process.

Segment Search Argument Definition

The SSA tag identifies the segments in the database to be accessed on a DL/I call. This optional tag with its attributes further describes the PROCESS and EPROCESS tag set.

You can specify the SSA tag only if the DLICALL tag is specified. The SSA tag can be repeated.

The order of the SSA tags determines the order of the DL/I segment search arguments for the database call.

Syntax	Attributes		
:process			
:			
:dlcall			
:			
:ssa	:SSA		
:			
:eprocess			
	Attributes	Values	Uses
	[CMDCODES=	'codes']	Specifies special processing to be performed by the DL/I call
	SEGNAME=	segment name	Specifies the DL/I segment accessed by the DL/I call
	[.]		

CMDCODES -

'codes' Specifies special processing to be performed by the DL/I call

inserted. Specify this code to process any higher level segment on INQUIRY, UPDATE, or SCAN options. For an ADD option, specify this code only for the highest level segment you want inserted to add that segment and all lower level segments.

You can specify up to four command codes. Refer to the DL/I manuals for a more detailed description of the command codes. Valid command codes are:

- L For INQUIRY, UPDATE, and SCAN options, this code retrieves the last occurrence of this segment type under its parent that satisfies the qualification specified. If qualification statements are present, it retrieves the last segment that satisfies the search criteria. For the ADD option, this code is effective only for segments with a nonunique or no sequence field, and the segment is inserted at the last position within its parent.
- F For the SCAN option, this code starts scanning from the first occurrence of this segment type under its parent that satisfies the qualification specified. For the ADD option, this code ID is effective only for segments with a nonunique or no sequence field, and the segment is inserted at the first position within its parent.
- D This code allows the retrieval or insertion of multiple segments in a hierarchical path. This code is not required for the lowest level segment because it is always retrieved or

CSP/AE handles input/output buffering for segments retrieved or written using the D command code. If you retrieve multiple segments for update using the D code, a REPLACE option, with the lowest level segment as the object, replaces all the segments that were retrieved with the D code.

Specify the path call process option (P) in DL/I PSB generation if the D command code is used.

- N This code specifies to not replace this segment on a REPLACE option even though it was retrieved on the GET for an UPDATE option.
- Q This code causes CSP/AE to lock the retrieved segments until checkpoint or PSB termination.
- U This code specifies to not move the database position from this segment while searching its hierarchical dependents.
- V This code is the same as U except that the command code is automatically set at all higher levels in the invocation.

- | C This code specifies that the concatenated key is to be used; it is supported only for IMS.
- | P This code sets the parent position; it is supported only for IMS.
- | R This code causes the retrieval of the first occurrence in the subset.
- | M This code moves the subset pointer to the next occurrence.
- | S This code sets the subset pointer to the current position.
- | W This code conditionally sets the subset pointer to the current position.
- | Z This code sets the subset pointer to 0.
- | - The hyphen is an additional valid command code that serves as a placeholder for a command code.

| The M, R, S, W, and Z command codes must be followed by an integer from 1 to 8.

Certain command codes are applicable only to certain process options. The following table identifies the applicable command codes:

Option	Command Codes
INQUIRY	C, D, L, M, P, Q, R, S, U, V, W, Z
UPDATE	C, D, L, M, P, Q, R, S, U, V, W, Z
ADD	C, D, F, L, M, P, R, S, U, V, W, Z
REPLACE	M, N, S, W, Z
DELETE	Z
SCAN	C, D, F, L, M, P, Q, R, S, U, V, W, Z

Command codes are optional. The L and F, U and V, R and F, and R and Q command codes are not compatible. You cannot have more than one M, S, W, or Z command code for the same segment search argument.

SEGNAME =	
<i>segment name</i>	Specifies the DL/I segment accessed by the DL/I call

Specify SEGNAME in the chain that goes from the object segment back to the root segment in the database hierarchy. In addition, define the segment to CSP/AD as a DL/I segment record before testing or generating the application.

Qualification Statement Definition for the Segment Search Argument

The QUAL tag specifies the qualification statement for the segment search argument (SSA) consisting of a segment field, a relational operator, a comparison value, and a Boolean operator. This optional tag with its attributes further describes the PROCESS and EPROCESS tag set.

You can specify the QUAL tag only if the SSA tag is specified, and you can use it multiple times for each SSA tag.

The order of the QUAL tags determines the order of the qualification statements on the call.

Syntax	Attributes		
:process : :dlcall : :ssa : :qual : :eprocess	:QUAL		
	Attributes	Values	Uses
	[BOOLOP =	& AND OR]	Identifies the presence of an additional qualification statement and shows how the true or false values of the qualification statements are to be combined
	COMPVAL =	'comparison value'	Specifies the comparison value item that contains the value to be compared to the contents of the segment field
	[RELOP =	relational operator]	Specifies the comparison to be done between the segment field and the comparison value item
	[SEGFIELD =	field name]	Specifies the field used for segment selection
	[.]		

BOOLOP =

& or AND	AND operator
or OR	OR operator

BOOLOP shows how the true or false values of this qualification statement and the following qualification are to be combined.

If you specified command code C for the SSA, omit this attribute.

COMPVAL =

'comparison value'	Specifies the comparison to be done in the segment field and the comparison value item
--------------------	--

At execution time, CSP/AD uses the value in this item as the field value in building the SSA for the DL/I call. The comparison value is compared to the contents of the segment field. If the comparison is true, the search criteria of this qualification statement is satisfied.

The maximum length for the comparison-value item is 85 characters. If a comparison-value item is longer than 71 characters, place a continuation character in column 72. Any character in column 72 is considered a continuation character. The continuation character causes concatenation of the two lines on import. The two lines concatenate with no blanks between the last character on the first line and the first character on the second line. The length of the concatenated comparison-value item can never exceed 85 characters.

COMPVAL always applies when the command code is C.

RELOP =	
= or EQ	Equal
> or GT	Greater than
< or LT	Less than
>=, = >, or GE	Greater than or equal
<=, = <, or LE	Less than or equal
≠, = ≠, or NE	Not equal

RELOP tells DL/I how to compare the values in the segment field and the comparison value item.

RELOP is required unless you specified command code C for the SSA. If you specified command code C for the SSA, omit the RELOP attribute.

SEGFIELD =	
<i>field name</i>	Specifies the field used for segment selection

Specify the field name as defined in the DL/I database description.

SEGFIELD is required unless you specified command code C for the SSA. If you specified command code C for the SSA, omit the SEGFIELD attribute.

Process Definition Syntax Example

```
:process name = xxxxxxxxxxxxxxxxxxxx date = 'mm/dd/yy' time = 'hh:mm:ss'
         option = xxxxxxxx object = xxxxxxxxxxxxxxxxxxxx
         errrtn = xxxxxxxxxxxxxxxxxxxx execbld = x model = xxxxxx
         refine = x singrow = x updproc = xxxxxxxxxxxxxxxxxxxx
         desc = 'This description is a process'.

:before.
MOVE A
TO B;
:ebefore.
:after.
MOVE A
TO B;
:eafter.
:sql hostvar = '?' clause = xxxxxxxxxxxx.clause text
:esql.
:dlicall dbname = xxxxxxxx psb = xxxxxxxx
         scanupd = x scanpar = x.
:ssa segname = xxxxxxxx cmdcodes = 'xxxx'.
:qual segfield = xxxxxxxx relop = xx
         compval = 'Enter up to 85 characters'
         boolop = xxx.
:qual segfield = xxxxxxxx relop = xx
         compval = 'Enter up to 85 characters'.
:eprocess.
```

Note: This example illustrates the syntax of all process definition tags and attributes. Actual definitions do not require or use all tags and attributes shown here.



Chapter 4. Statement Group Structures

This chapter describes the external source format tags and attributes that define a statement group structure in CSP/AD.

Statement Group Definition

The GROUP tag with its attributes describes a statement group member. The STMTS and ESTMTS tags further define aspects of a statement group member and can have attributes. The EGROUP tag closes the definition, and if you use the GROUP tag, the EGROUP tag is required.

The GROUP tag specifies a set of Cross System Product processing statements that perform processing only (for example, statements that perform no input/output operations). When a statement group finishes executing, control returns to the processing statement following the statement that started the execution of the statement group.

Syntax	Attributes		
:group	:GROUP		
:	Attributes	Values	Uses
:stmts	[DATE =	'modification date']	Specifies the date the statement group was last modified
:	[DESC =	'statement group description']	Describes a statement group
:estmts	NAME =	name	Specifies a statement group member
:	[REFINE =	Y N]	Specifies the method for displaying logic for the process when definition is performed on a programmable workstation facility
:egroup	[TIME =	'modification time']	Specifies the time the statement group was last modified
	[.]		
	:EGROUP[.]		

DATE =

'modification date' Specifies the date the statement group was last modified

date shows when the statement group was last modified in the MSL. The date is ignored on import.

DATE format is mm/dd/yy. If the external source format file was produced by CSP/AD export, this

DESC =

'statement group description'
Describes a statement group

DESC is a 30-character text description of a statement group, not including the quotes.

NAME =
name Specifies a statement group member in the MSL

NAME consists of 1 to 18 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The name cannot contain blanks or have an EZE prefix.

The name can be a DBCS name up to 8 DBCS characters long with no embedded blanks.

REFINE =

Y	Specifies that the refinement takes place on a single screen
N	Specifies that the refinement of process logic takes place on multiple screens

REFINE specifies the method of displaying logic for the process when process members are defined on a programmable workstation facility. The attribute is preserved in the MSL, but has no effect on entering the statement group definition using CSP/AD on the host or on the execution of the generated application.

TIME =
'modification time' Specifies the time the statement group was last modified

TIME format is hh:mm:ss. If the external source format file was produced by CSP/AD export, this time shows when the statement group was last modified in the MSL. The time is ignored on import.

Chapter 5. Record Structures

This chapter describes the external source format tags and attributes that define records in CSP/AD.

Record Definition

The RECORD tag defines attributes for individual record members. The SQLTABLE, JOINCON, PROL, and RECDITEM tags further define aspects of a record and can have attributes. The ERECORD tag closes the definition, and if you use the RECORD tag, the ERECORD tag is required.

Syntax	Attributes	Values	Uses
:record ⋮ :sqltable ⋮ :joincon ⋮ :ejoincon ⋮ :prol ⋮ :eprol ⋮ :recditem ⋮ :erecord	:RECORD		
	Attributes		
	[ALTSPEC =	<i>record name</i>]	Specifies an existing record whose data item structure is to be used for this record
	[DATE =	<i>'modification date'</i>]	Specifies the date the record was last modified
	[FILELOC =	<i>file location</i>]	Specifies a host transaction identification name for use by DPPX/SP
	[FILENAME =	<i>file name</i>]	Associates a record specification with a physical file
	[KEY =	<i>item name</i>]	Specifies the data item that contains the record key or relative record number
	NAME =	<i>name</i>	Specifies a record in the MSL
	[NUMOCCUR =	<i>number of occurrences</i> <i>item name</i>]	Specifies the data item that contains the actual number of occurrences for the array that has a variable number of entries or elements
	[ORG =	INDEXED RELATIVE SERIAL DLISEG SQLROW WORKSTOR REDEFREC]	Specifies the organization of the record
	[REDEFREC =	<i>record name</i>]	Specifies a record to be redefined
	[SCOPE =	GLOBAL LOCAL]	Specifies the default scope for items in the record
	[TIME =	<i>'modification time'</i>]	Specifies the time the record was last modified
	[VARLENTH =	<i>variable length item name</i>]	Specifies a data item in a DL/I segment that contains the length of the rest of the segment or a data item in an indexed or serial file that contains the length value for the record
	[.]		
	:ERECD[.]		

ALTSPEC =

<i>record name</i>	Specifies an existing record whose data item structure is to be used for this record
--------------------	--

addition, an SQL row record should not be defined as an alternate specification for a record with another organization. Conversely, records not defined as SQL row records should not be specified as alternate specifications for SQL row records.

ALTSPEC should not be specified for a record that is already defined as an alternate specification. In

Note: If ALTSPEC is specified for any of the above records, the application will not generate.

DATE =	
<i>'modification date'</i>	Specifies the date the record was last modified

DATE format is mm/dd/yy. If the external source format file was produced by CSP/AD export, this date shows when the record was last modified in the MSL. The date is ignored on import.

FILELOC =	
<i>file location</i>	Specifies a host transaction identification name for use by DPPX/SP

FILELOC is used only by the DPPX/SP Host Transaction Facility (HTF) and specifies the transaction identification name. If FILELOC is not specified, the record exists on the local system.

You must specify a host transaction name of 1 to 8 characters if the record exists on a host system. The name must meet the following conventions:

- The first character must be alphabetic or national (A-Z, #, \$, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, #, \$, @).
- The name cannot contain blanks or have an EZE prefix.

FILENAME =	
<i>file name</i>	Associates a record specification with a physical file

FILENAME is specified for serial, relative, and indexed files.

FILENAME consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, #, \$, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, #, \$, @).
- The file name cannot contain blanks or have an EZE prefix.

A message file name cannot be used as a file name.

KEY =	
<i>item name</i>	Specifies the data item that contains the record key for an indexed file or DL/I segment record, the relative record number for a relative file, or search field for an SQL row record

For indexed files: KEY must be defined in the data item list for the record. The key item should have the same length and record offset as the key in the records in the physical file.

For relative files: KEY does not need to be defined within the record data structure. Define the key item with a numeric (NUM), packed (PACK), or binary (BIN) data type, no decimal positions, and a maximum length of nine digits.

When a relative record file is to be accessed at execution, the key item must contain a number that indicates the record position in a file relative to the beginning of the file.

For DL/I: KEY specifies the name of an item in a DL/I segment record that contains the segment key. The item must have the same name, length, and offset that the segment sequence field has in the DL/I database description. The key does not have a default value. Do not specify the KEY attribute if the DL/I segment has no sequence field.

The name consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, \$, #, @).
- The name cannot contain blanks or have an EZE prefix.

For SQL: KEY specifies the name of the item in an SQL row record that is to be used as the search field in default SQL SELECT statements. The KEY specification is optional. KEY is only valid when ALTSPEC is specified.

NAME =

name Specifies a record in the MSL

NAME consists of 1 to 18 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The name cannot contain blanks or have an EZE prefix.

The name can be a DBCS name up to 8 DBCS characters long with no embedded blanks.

The name for a DL/I segment name can consist of 1 to 8 characters. It cannot contain underscores or hyphens, and it cannot be a DBCS name.

NUMOCCUR =

number of occurrences item name
Specifies the data item that contains the actual number of occurrences for the array that has a variable number of entries or elements

NUMOCCUR is supported only with indexed and serial records. You can specify NUMOCCUR only for variable length records. The data item NUMOCCUR must have the following characteristics:

- A numeric (NUM), binary (BIN), or packed (PACK) data type
- A length of up to 9 digits with no decimal positions
- Located in the fixed length part of the record.

ORG =

INDEXED	Specifies indexed organization (default value)
DLISEG	Specifies DL/I segment organization
REDEFREC	Specifies redefined record organization
RELATIVE	Specifies relative organization
SERIAL	Specifies serial organization
SQLROW	Specifies SQL row organization
WORKSTOR	Specifies working storage organization

ORG describes how the file in which the record resides is organized. The organization determines which process options can be used to access the record in the application.

INDEXED: Indexed organization indicates that the records are in a file and are accessed by a key which is specified in the key element.

DLISEG: DL/I segment organization indicates that the record is a segment in a DL/I database. The record name must be the same as the name with which the segment is defined to DL/I.

REDEFREC: A redefined record is an alternate data item structure for an existing record. The alternate data structure lets the application access the information in a record using different data item names and definitions.

RELATIVE: Relative organization indicates that the file is an ordered set of fixed length records accessed by a relative record number, that is found in the key item specified for the record. For relative record access, the key item does not need to be part of the record structure. It can be in any map, record, or table used in the application.

SERIAL: Serial organization indicates that the records are stored in the file in sequential order. References to the records start at the beginning and go consecutively to the end of the file.

SQLROW: SQL row organization indicates that the record represents a row in a table in a relational database.

WORKSTOR: Working storage records define storage areas for temporary data items that are used in applications. The data items are not saved after execution unless they have been moved to a record and placed in a file.

Working storage can be defined as a unit of related items (data structure) in the same manner as a record. One or more single, unrelated data items can be defined for working storage instead of or in addition to the structure. Single data items are referred to as level-77 items. These items are defined with a LEVEL of 77 after all data items in the working storage structure have been defined.

Redefined records cannot be used as process objects, but they can be used in processing statements and as passed parameters.

The following table shows which tags and attributes are valid for the record organization types listed previously. Bullets indicate the tags and attributes that are valid. An **R** indicates that the tag or attribute is required.

<i>Tags and Attributes</i>	I N D E X E D	R E L A T I V E	S E R I A L	W O R K S T O R	D L I S E G	R E D E F R E C	S Q L R O W
:record	R	R	R	R	R	R	R
ALTSPEC =
DATE =
FILELOC =	.	.	.				
FILENAME =	R	R	R				
KEY =	R	R			.		.
NAME =	R	R	R	R	R	R	R
NUMOCCUR =	.		.				
ORG =
REDEFREC =						R	
SCOPE =
TIME =
VARLENTH =	.		.		.		
:sqltable							.
CREATOR =							.
LABEL =							.
TABLEID =							R*
TBLENAM =							R*
:joincon							.
HOSTVAR =							.
:ejoincon							.
:prol
:eprol
:recditem
BYTES =
COLNAME =							.
DATACODE =							.
DECIMALS =
DESC =
EVENSQ =
KEY =							.
LEVEL =	
NAME =	R	R	R	R	R	R	R
OCCURS =	
READONLY =							.
SCOPE =
TYPE =
:erecord	R	R	R	R	R	R	R

* Use either TABLEID or TBLENAME. Only one can be specified.

REDEFREC =	
<i>record name</i>	Specifies a record to be redefined

REDEFREC identifies the name of the record that is being redefined when record organization is specified as redefined record. You can specify REDEFREC only for records with ORG = REDEFREC.

SCOPE =	
GLOBAL	Indicates that global data item definitions are to be used for this item in the table
LOCAL	Indicates that local data item definitions are to be used for this item in the table

SCOPE defines the default scope for items within the record.

TIME =	
<i>'modification time'</i>	Specifies the time the record was last modified

TIME format is hh:mm:ss. If the external source format file was produced by CSP/AD export, this time shows when the record was last modified in the MSL. The time is ignored on import.

VARLENTH =	
<i>variable length item name</i>	Specifies a data item in a DL/I segment that contains the length of the rest of the segment or a data item in an indexed or serial file that contains the length value for the record

VARLENTH is the name of a data item in a DL/I segment that contains the length value for the record. Specify the name if the segment has a variable length. You cannot specify VARLENTH if the segment has fixed length. The data item must have the same length and offset as the length field in the segment in the DL/I database description.

For DL/I, VARLENTH consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, \$, #, @).
- The name cannot contain blanks or have an EZE prefix.

For indexed or serial records, VARLENTH is the name of the data item that contains the length value for the record. The VARLENTH item does not have to be within the record. If the record has fixed length, do not specify VARLENTH.

VARLENTH must have the following characteristics:

- A numeric (NUM), binary (BIN), or packed (PACK) data type
- A length of up to 9 digits with no decimal positions.

SQL Table Name Definition

The SQLTABLE tag specifies the name of a relational table that an SQL row record represents. The SQLTABLE tag is required for an SQL row record unless there is an alternate specification record specified (altspec). This optional tag further describes the RECORD and ERECORD tag set.

Repeat the SQLTABLE tag for each table represented by the record.

Syntax	Attributes		
:record	:SQLTABLE		
:	Attributes	Values	Uses
:sqltable	[CREATOR =	'creator ID']	Specifies the user identifier or authorization ID of the creator of the table
:	[LABEL =	'table label']	Specifies a shortened version of the table name
:joincon	[TABLEID =	'table ID']	Identifies the table represented by the SQL row record
:	[TBLENAME =	'table name']	Identifies the table represented by the SQL row record
:ejoincon	[.]		
:			
:erecord			

CREATOR =

'creator ID' Specifies the user identifier or authorization ID of the creator of the table

TABLEID =

'table ID' Identifies the table represented by the SQL row record

CREATOR plus the table name uniquely identifies a table in a relational database. The maximum length of a creator ID is 8 bytes. CREATOR can only be specified if you use TBLENAME.

The format of the table ID is not checked, and it can be any name that is coded in a FROM clause in an SQL SELECT statement. The maximum length of the table ID is 60 bytes. The table ID name is inserted in SQL statements and passed to the database manager exactly as specified.

LABEL =

'table label' Specifies a shortened version of the table name

Note: Use either TABLEID or TBLENAME. Only one can be specified.

Use LABEL as a qualifier to uniquely identify column names in SQL row definitions and SQL statements when the SQL row record represents two or more tables joined together. The maximum length of a table label is 4 bytes.

TBLENAME =

'table name' Identifies the table represented by the SQL row record

IMPORT automatically generates a table label if the label is not specified.

TBLENAME is left justified and padded with blanks, but is not changed to uppercase or checked for validity.



| The table name will be concatenated to the
| CREATOR and inserted in SQL statements and
| passed to the database manager exactly as

specified. The maximum length of a table name is
20 bytes.

| **Note:** Use either TBLENAM or TABLEID. Only
| one can be specified.



Default Selection Criteria Definition

The JOINCON tag defines the default selection criteria used when SELECT statements are built for processes that have an SQL row record as the process object. An SQL join permits data retrieval from two or more tables based on matching columns, and a join condition specifies a relationship between the tables to be joined.

You can specify JOINCON only if you have specified values for TABLEID or TBLENAME on the SQLTABLE tag or ALTSPEC on the RECORD tag. The JOINCON tag is optional and further describes the RECORD and ERECORD tag set. The EJOINCON tag closes the default selection criteria, and if you use the JOINCON tag, the EJOINCON tag is required.

Syntax	Usage	Uses
record :	:JOINCON	
:sqltable :	Attributes	Values
:joincon :	[HOSTVAR =	'?' '@']
:ejoincon :	[. [<i>clause text</i>]]	Specifies the character used within the clause text to indicate the beginning of a host variable name
:erecord :	:EJOINCON [.]	Contains the WHERE clause of the join condition.

HOSTVAR =

'?'	Specifies that the question mark (?) is used to begin host variable names
'@'	Specifies that the at sign (@) is used to begin host variable names

Export always uses the question mark (?) as the host variable name indicator so that the at sign (@) can be properly interpreted as a valid character in a name. Using the at sign (@) as the host variable name indicator, when variable names also contain the at sign (@), is ambiguous and can cause unpredictable results on Import.

CLAUSE TEXT is the WHERE clause of the join condition. A line in the external source format file corresponds to a line on screen EZEM16 - SQL Row Default Select Conditions Definition.

Within the clause text, SQL column names always begin with the exclamation point (!). When produced from CSP/AD Export, host variable names always begin with the question mark (?).

Statements can consist of multiple lines of text. If a line of text is longer than 71 characters, place a continuation character in column 72. Any character in column 72 is a continuation character. The continuation character causes concatenation of the two lines on import. The two lines concatenate with no blanks between the last character on the first line and the first character on the second line. The concatenated line length can never exceed 73 bytes.

Record Prolog Definition

The PROL tag is a text description of a defined record. This optional tag further defines the RECORD and ERECORD tag set. The EPROL tag closes the description, and if you use the PROL tag, the EPROL tag is required.

Syntax	Usage
:record	
:	
:prol	:PROL
:	Value
:eprol	Usage
:	[. [<i>prolog lines</i>]]
:erecord	Describes the record.
	:EPROL[.]

Only the MSL size and machine size limit the number of lines that you can specify. Each line can contain up to 60 characters. You can use both uppercase and lowercase in the prolog. Text is

saved as entered. When text is longer than 60 characters, it is split into two lines during import.

Record Item Definition

The RECDITEM tag is a list of data items to be defined for this record. This optional tag with its attributes further defines the RECORD and ERECORD tag set.

- | Repeat the RECDITEM tag for each data item in the record except when it is an alternate specification record. Specify the TYPE, BYTES, DECIMALS, EVENSQ, and DESC attributes for local data items only.
- | Define the global data items in the list with the ITEM tag. The order of the RECDITEM tags determines the order of the data items within the record.

Syntax	Attributes																																												
<pre> :record : :recditem : :erecord </pre>	<p>:RECDITEM</p> <table border="0" style="width: 100%;"> <thead> <tr> <th style="text-align: left;">Attributes</th> <th style="text-align: left;">Values</th> <th style="text-align: left;">Uses</th> </tr> </thead> <tbody> <tr> <td>[BYTES =</td> <td>3 <i>field length in bytes</i>]</td> <td>Specifies the field length for filler items</td> </tr> <tr> <td>[COLNAME =</td> <td>'<i>column name</i>'</td> <td>Specifies the column name that identifies an item in an SQL row record to the relational database manager</td> </tr> <tr> <td>[DATACODE =</td> <td>SQL <i>data code</i>]</td> <td>Specifies the SQL data type of the data item in an SQL row record to the relational database manager</td> </tr> <tr> <td>[DECIMALS =</td> <td>0 <i>decimal places</i>]</td> <td>Specifies the number of decimal places</td> </tr> <tr> <td>[DESC =</td> <td>'<i>field description</i>'</td> <td>Describes a data item</td> </tr> <tr> <td>[EVENSQ =</td> <td>Y N]</td> <td>Specifies whether packed fields of even length or odd length are passed to a relational database</td> </tr> <tr> <td>[KEY =</td> <td>Y N]</td> <td>Specifies whether the data item is a key column for the SQL table</td> </tr> <tr> <td>[LEVEL =</td> <td>10 <i>number</i>]</td> <td>Indicates relative placement of a data item to adjacent data items</td> </tr> <tr> <td>NAME =</td> <td>* <i>item name</i></td> <td>Specifies a data item member</td> </tr> <tr> <td>[OCCURS =</td> <td>1 <i>number of occurrences</i>]</td> <td>Specifies the number of repetitions of a data item in a data structure</td> </tr> <tr> <td>[READONLY =</td> <td>Y N]</td> <td>Indicates whether an item in an SQL row record can be written to the relational database</td> </tr> <tr> <td>[SCOPE =</td> <td>GLOBAL LOCAL]</td> <td>Defines the scope for the data item</td> </tr> <tr> <td>[TYPE =</td> <td>BIN CHA DBCS HEX MIX NUM NUMC PACK PACF]</td> <td>Specifies the data item type for filler items</td> </tr> </tbody> </table> <p>[.]</p>			Attributes	Values	Uses	[BYTES =	3 <i>field length in bytes</i>]	Specifies the field length for filler items	[COLNAME =	' <i>column name</i> '	Specifies the column name that identifies an item in an SQL row record to the relational database manager	[DATACODE =	SQL <i>data code</i>]	Specifies the SQL data type of the data item in an SQL row record to the relational database manager	[DECIMALS =	0 <i>decimal places</i>]	Specifies the number of decimal places	[DESC =	' <i>field description</i> '	Describes a data item	[EVENSQ =	Y N]	Specifies whether packed fields of even length or odd length are passed to a relational database	[KEY =	Y N]	Specifies whether the data item is a key column for the SQL table	[LEVEL =	10 <i>number</i>]	Indicates relative placement of a data item to adjacent data items	NAME =	* <i>item name</i>	Specifies a data item member	[OCCURS =	1 <i>number of occurrences</i>]	Specifies the number of repetitions of a data item in a data structure	[READONLY =	Y N]	Indicates whether an item in an SQL row record can be written to the relational database	[SCOPE =	GLOBAL LOCAL]	Defines the scope for the data item	[TYPE =	BIN CHA DBCS HEX MIX NUM NUMC PACK PACF]	Specifies the data item type for filler items
Attributes	Values	Uses																																											
[BYTES =	3 <i>field length in bytes</i>]	Specifies the field length for filler items																																											
[COLNAME =	' <i>column name</i> '	Specifies the column name that identifies an item in an SQL row record to the relational database manager																																											
[DATACODE =	SQL <i>data code</i>]	Specifies the SQL data type of the data item in an SQL row record to the relational database manager																																											
[DECIMALS =	0 <i>decimal places</i>]	Specifies the number of decimal places																																											
[DESC =	' <i>field description</i> '	Describes a data item																																											
[EVENSQ =	Y N]	Specifies whether packed fields of even length or odd length are passed to a relational database																																											
[KEY =	Y N]	Specifies whether the data item is a key column for the SQL table																																											
[LEVEL =	10 <i>number</i>]	Indicates relative placement of a data item to adjacent data items																																											
NAME =	* <i>item name</i>	Specifies a data item member																																											
[OCCURS =	1 <i>number of occurrences</i>]	Specifies the number of repetitions of a data item in a data structure																																											
[READONLY =	Y N]	Indicates whether an item in an SQL row record can be written to the relational database																																											
[SCOPE =	GLOBAL LOCAL]	Defines the scope for the data item																																											
[TYPE =	BIN CHA DBCS HEX MIX NUM NUMC PACK PACF]	Specifies the data item type for filler items																																											

BYTES =	
3	Specifies 3 bytes (default value)
<i>field length in bytes</i>	Specifies the field length in bytes

BYTES is the number of bytes required to store a data item value internally. The following table lists the maximum length of data items:

Item Type	Byte Values
CHA, HEX, Mixed	1-32,767
DBCS	1-32,766
NUM, NUMC	1-18
PACK, PACF	1-10
PACK in SQL row record	1-8
BIN	2, 4, 8 only
BIN in SQL row record	2, 4 only

Note: The BYTES attribute can be specified only for local data items on the RECDITEM tag.

COLNAME =	
<i>'column name'</i>	Specifies the column name that identifies an item in an SQL row record to the relational database manager

COLNAME can be up to 36 characters long and can be either of the following:

- The actual name of a column in the relational table or view definition

If the SQL row record is defined as a join of multiple tables or views, the column name should be qualified by the table label defined for the table or view to which it belongs.
- An SQL expression made up of column names and SQL operators, constants, and built-in functions.

As an expression is entered into the SQL column name field, a virtual column is defined which can be used as a read-only data item in the SQL row record definition. The expression is calculated at the time the SQL row is read from the database.

The specified name is not checked for validity. All single-byte characters not within double quotes are folded to uppercase. The name is inserted into generated SQL statements as entered. The name is validated by the relational database manager during SQL statement preparation for an application.

DATACODE =	
<i>SQL data code</i>	Identifies the SQL data type of the data item to the relational database manager

CSP/AD determines the SQL data codes for all data items except HEX data items. You must enter the data code for HEX items. The DATACODE attribute is produced on export for HEX, CHA, and DBCS data items only.

DECIMALS =	
0	Specifies no decimal places (default value)
<i>decimal places</i>	Specifies the number of decimal places

DECIMALS specifies the number of decimal places in numeric data items. This number is included in a data item's length; however, the decimal point is not stored with the data. The default is 0 for no decimal places.

Note: The DECIMALS attribute can be specified only for local data items on the RECDITEM tag.

DESC =	
<i>'field description'</i>	Describes a data item

DESC consists of 1 to 30 characters that can be entered in uppercase and lowercase. Mixed data is also permitted.

Note: The DESC attribute can be specified only for local data items on the RECDITEM tag.

EVENSQ =

- Y** Specifies whether packed fields of even length are passed to a relational database
- N** Specifies whether packed fields of odd length are passed to a relational database (default value)

EVENSQ indicates whether CSP/AD and CSP/AE pass either even- or odd-length packed fields to the SQL/DS or DB2 relational databases. Specify EVENSQ as Y only if you are using an SQL table that has even-length columns defined.

Note: The EVENSQ attribute can be specified only for local data items on the RECDITEM tag.

KEY =

- Y** Specifies that a data item is a key column for the SQL table
- N** Specifies that a data item is not a key column for the SQL table

KEY specifies that the data item is a key column for the SQL table. One use of KEY is when the SQL table is defined with a multiple-column key. This enables CSP/AD to provide default SQL statements that use the indexes in the SQL table correctly.

Note: KEY is used only with SQL row records.

LEVEL =

- 10** Default level number
- number* Specifies the relative placement of a data item to adjacent data items in a data structure

LEVEL is a number (3-49, or 77 for single data items in working storage) that is unique to a record, table, or working storage definition. Level numbers can differ for the same data item that is used in several record, table, or working storage definitions.

Data items with the lowest level number in a structure occupy the highest position in the structure. Data items with higher level numbers represent substructures of the previous item in the structure list with a lower level number.

The total length of the data items in a substructure must equal the length of the owning data item.

Working storage can contain single data items in addition to or in place of a data structure. Single data items have a level specified as 77 and must follow the structure if one exists.

NAME =

- *** Specifies a filler data item
- item name* Specifies the data item member

For a DL/I item, NAME identifies each data item. It consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, \$, #, @).
- The name cannot contain blanks or have an EZE prefix.

For a non-DL/I item, NAME identifies each data item. It consists of 1 to 32 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The name cannot contain blanks or have an EZE prefix.

For a non-DL/I item, the name can be a DBCS name up to 15 DBCS characters long with no embedded blanks.

A data item name in a record structure can be specified with an asterisk. This type of data item, called a filler, cannot be referred to by the application; it acts as a space holder.

Note: SQL row records cannot contain a filler data item. A filler is always a local data item.

OCCURS =

1 Default data item OCCURS number
number of occurrences
 Specifies the number of repetitions of a data
 item in a data structure

You can specify OCCURS as a number from 1 to 32,767. Data items cannot have nested occurrences. Once you define a data item greater than 1 with OCCURS, no data item within its substructure can have an OCCURS value greater than 1.

The OCCURS characteristic for a data item applies only for the record in which the item is defined. If you use a data item in more than one record or working storage, the OCCURS can be different in each.

READONLY =

Y Indicates an item in an SQL row record cannot be
 written to the relational database
N Indicates an item in an SQL row record can be
 written to the relational database

READONLY determines what columns are included in generated SQL statements that write to the database. Specify READONLY=Y for columns that cannot be updated and for columns that the application never needs to change. In addition, the following items must be read only:

- Items with SQL column names that are expressions (not valid SQL column names)
- Items that come from an SQL row defined as a join.

SCOPE =

GLOBAL Indicates that global data item definitions are to
 be used for this item in the record
LOCAL Indicates that local data item definitions are to
 be used for this item in the record

SCOPE defines the scope for the data item. The SCOPE value overrides the value specified on the SCOPE attribute of the RECORD tag. If a SCOPE value is not specified, the value specified on the SCOPE attribute of the RECORD tag is the default.

TYPE =

BIN Binary number
CHA Character data (default value)
DBCS Double-byte character data
HEX Hexadecimal data
MIX DBCS data intermingled with single-byte
 character data
NUM Numeric characters with positive sign in
 F format
NUMC Numeric characters with positive sign in
 C format
PACF Packed decimal characters with positive sign in
 F format
PACK Packed decimal characters with positive sign in
 C format

TYPE specifies the internal format or type of data. The data type determines how the item is processed when referred to in processing statements.

Note: The TYPE attribute can be specified for local data items only on the RECDITEM item tag.

Record Definition Syntax Example

```
:record      name      = xxxxxxxxxxxxxxxxxxxx  date = 'mm/dd/yy'
             time = 'hh:mm:'                    org      = xxxxxxxx
             altspec   = xxxxxxxxxxxxxxxxxxxx  filename = xxxxxxxx
             key       = xxxxxxxxxxxxxxxxxxxx
             redefrec  = xxxxxxxxxxxxxxxxxxxx
             varlenth  = xxxxxxxxxxxxxxxxxxxx
             numoccur  = xxxxxxxxxxxxxxxxxxxx
             fileloc   = xxxxxxxx
             scope     = xxxxxxx.
:sqltable    creator   = 'xxxxxxx'             tableid = 'Enter up to 60 characters'
             tblname   = 'xxxxxxx'
             label     = 'xxxx'.
:joincon     hostvar   = '?'.
             EMPNO IN (SELECT EMPNO FROM DSN8130.TEMPL WHERE
             WORKDEPT = 'E11');
:ejoincon.
:prol.
Each prolog line can be up to 60 characters long. IMPORT
will split lines longer than 60 characters.
:eprl.
:recditem    name      = xxxxxxxxxxxxxxxxxxxx  level = xx
             occurs   = xxxxx      type    = xxxx      bytes = xxxxx
             colname  = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
             readonly = x          datacode = xxx      scope = xxxxxx
             decimals = xx         evensql  = x
             desc     = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
             key      = x.
:erecord.
```

Note: This example illustrates the syntax of all record definition tags and attributes. Actual definitions do not require or use all tags and attributes shown here.

Chapter 6. Table Structures

This chapter describes the external source format tags and attributes that define tables in CSP/AD.

Table Definition

The TBLE tag and its attributes define table members. The TBLE tag specifies data that can be used for the following:

- Editing data that a user enters on a map.
- Storing information that an application refers to during execution.

The GENOPTS, PROL, DEFITEM, CONTITEM, and ROW tags further define aspects of a table member and can have attributes. The ETBLE tag closes the definition, and if you use the TBLE tag, the ETBLE tag is required.

Syntax	Attributes		
:tble	:TBLE		
:	Attributes	Values	Uses
:genopts	[DATE =	' <i>modification date</i> '	Specifies the date the table was last modified
:	[FOLD =	Y N]	Specifies whether character data or single-byte data in mixed fields in the table is changed to uppercase or left as entered in the table contents
:prol			
:			
:eprol			
:			
:defitem	NAME =	<i>name</i>	Identifies a table member in the MSL
:			
:contitem	[SCOPE =	GLOBAL LOCAL]	Specifies the default scope for items in the table
:			
:row	[TABTYPE =	UNSPECIFIED MATCHVALID MATCHINVALID RANGEMATCH]	Defines how the table is to be used
:			
:etble	[TIME =	' <i>modification time</i> '	Specifies the time the table was last modified
	[.]		
	:ETBLE[.]		

DATE =

'*modification date*' Specifies the date the table member was last modified

DATE format is mm/dd/yy. If the external source format file was produced by CSP/AD export, this date shows when the table member was last modified in the MSL. The date is ignored on import.

FOLD =

Y Specifies that the contents of a table are folded to uppercase when updated by the user, and that test facility and generation fold the contents as they are used (default value)

N Specifies that table contents will be used as they were last entered

IMPORT does not fold the table contents. Table definition uses this flag to determine if folding was requested.

NAME =

name Identifies a table member in the MSL

NAME identifies a table member in the MSL. It consists of 1 to 7 characters and must meet the following conventions:

- The first character must be alphabetic (A-Z).
- The remaining characters must be alphanumeric (A-Z, 0-9).
- The name cannot contain blanks or have an EZE prefix.
- Table names cannot end in 0.

SCOPE =

GLOBAL Indicates that global data item definitions are to be used for this item in the table
LOCAL Indicates that local data item definitions are to be used for this item in the table

SCOPE defines the default scope for the data items within the table.

TABTYPE =

UNSPECIFIED	The table is a reference table only (default value). It cannot be used for editing data (as an EDITRTN on the MAPEDITS tag). This type of table can be used only with CSP/AD processing statements.
MATCHVALID	The table can be used for editing data. Data entered by a user must match a value in the first column of the table.
MATCHINVALID	The table can be used for editing data. Data entered by a user must not match any of the values in the first column of the table.
RANGEMATCH	The table can be used for editing data. Data entered by a user must be within a range of values that are contained in the table data.

TABTYPE defines how the table is to be used.

TIME =

'modification time' Specifies the time the table member was last modified

TIME format is hh:mm:ss. If the external source format file was produced by CSP/AD export, this time shows when the table member was last modified in the MSL. The time is ignored on import.

Table Generation Option Specification

The GENOPTS tag specifies the saved generation options associated with the table. This optional tag further describes the TBLE and ETBLE tag set.

Syntax	Attributes		
.:tbl : :genopts : :etble	:GENOPTS		
	Attributes	Values	Uses
	[LOADLIB =	<i>load library name</i>]	Specifies the library to which the generated modules are written
	[TYPEUSE =	SHARED SINGLE]	Specifies whether the table is to be shared by multiple occurrences of an application
	[.]		

LOADLIB =

load library name Specifies the library to which the generated modules are written

TYPEUSE =

SHARED Specifies that the table is to be shared by multiple occurrences of an application or by multiple applications (default value)

SINGLE Specifies that each occurrence of an application gets its own copy of the table

The load library is the application load file (ALF), a VSAM data set, used by CSP/AE during generation. The default is FZERSAM.

SINGLE allows writing to a table for temporary storage.

Table Prolog Description

The PROL tag begins a text description of a defined table. This optional tag further describes the TBLE and ETBLE tag set. The EPROL tag closes the description, and if you use the PROL tag, the EPROL tag is required.

Syntax	Value	Usage
:tbl :		
:prol :	[.{ <i>prolog lines</i> }]	Describes the table
:eprol :	:EPROL{.}	
:etble		

Only the MSL size and machine size limit the number of lines that you can specify. Each line can contain up to 60 characters. You can use both uppercase and lowercase in the prolog text.

Prologs are saved as entered. When text is longer than 60 characters, it is split into multiple lines during import.

BYTES =	
3	Specifies 3 bytes (default value)
<i>field length in bytes</i>	Specifies the field length in bytes

BYTES is the number of bytes required to store a data item value internally. The following table lists the maximum length of data items:

Item Type	Byte Values
CHA, DBCS, Mixed	1-254
HEX	1-127
NUM, NUMC	1-18
PACK, PACF	1-10
BIN	2, 4, 8 only

Note: BYTES only applies when the item is a local data item or table contents are supplied but the CONTITEM tags are not specified.

DECIMALS =	
0	Specifies no decimal places (default value)
<i>decimal places</i>	Specifies the number of decimal places

DECIMALS is the number of positions to the right of an implied decimal point in numeric items. The length specified for the data item must include the places set aside for decimal positions. The maximum number of decimal positions is 18 or the number of digits defined for the data item, whichever is smallest. The decimal point is not stored with the data.

Note: DECIMALS only applies when the item is a local data item or when table contents are supplied but the CONTITEM tag is not specified.

DESC =	
<i>'field description'</i>	Describes a data item

DESC consists of 1 to 30 characters that can be entered in uppercase and lowercase. Mixed data is also permitted.

Note: DESC only applies when the item is a local data item.

EVENSQ =	
Y	Specifies whether packed fields of even length are passed to a relational database
N	Specifies whether packed fields of odd length are passed to a relational database (default value)

EVENSQ indicates whether CSP/AD and CSP/AE pass either even- or odd-length packed fields to the SQL/DS or DB2 relational databases. Specify EVENSQ as Y only if you are using an SQL table that has even-length columns defined.

Note: EVENSQ only applies when the item is a local data item.

LEVEL =	
10	Default level number
<i>number</i>	Specifies the relative placement of a data item to adjacent data items in a data structure

LEVEL is a number (3-49) that is unique to a record, table, or working storage definition. Level numbers can differ for the same data item that is used in several record, table, or working storage definitions.

Data items with the lowest level number in a structure occupy the highest position in the structure. Data items with higher level numbers represent substructures of the previous item in the structure list with a lower level number.

The total length of the data items in a substructure must equal the length of the owning data item.

NAME -

*	Specifies a filler data item
<i>item name</i>	Specifies the data item member

NAME consists of 1 to 32 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The name cannot contain blanks or have an EZE prefix.

The name can be a DBCS name up to 15 DBCS characters long with no embedded blanks.

A data item name in a table structure can be specified with an asterisk. This type of data item, called a filler, cannot be referred to by the application; it acts as a space holder.

Note: A filler is always a local data item.

SCOPE -

GLOBAL	Indicates that global data item definitions are to be used for this item in the table
LOCAL	Indicates that local data item definitions are to be used for this item in the table

SCOPE defines the scope for the data item. The SCOPE value overrides the value specified on the SCOPE attribute of the TBLE tag. If a SCOPE value is not specified, the value specified on the SCOPE attribute of the TBLE tag is the default.

TYPE -

BIN	Binary number
CHA	Character data (default value)
DBCS	Double-byte character data
HEX	Hexadecimal data
MIX	DBCS data intermingled with single-byte character data
NUM	Numeric characters with positive sign in F format
NUMC	Numeric characters with positive sign in C format
PACF	Packed decimal characters with positive sign in F format
PACK	Packed decimal characters with positive sign in C format

TYPE specifies the internal format or type of data. The data type determines how the item is processed when referred to in processing statements.

Note: TYPE only applies when the item is a local data item or when table contents are supplied but the CONTITEM tag is not specified.

Data Content Attribute Identification

The CONTITEM tag specifies the contents structure and identifies the attributes of the actual data contents. These attributes may conflict with those in DEFITEM if the table definition has been changed and the contents have not yet been updated. If the contents structure derived from the DEFITEM tags matches the table contents, the CONTITEM tag is not necessary. The following table lists the data items with the lowest level number of the table structure:

Table Structure	Contents Structure
10 NAME 20 LAST 20 FIRST 20 MIDDLE	NAME
10 ADDRESS 20 LINE1 20 LINE2 20 LINE3	ADDRESS

If CONTITEM tags are not specified but ROW tags are specified, CSP/AD derives the contents structure from the DEFITEM tag information (name, level, type, bytes, and decimals). The CONTITEM tag is optional and further defines the TBLE and ETBLE tag set.

Syntax	Attributes															
:tbl : :contitem : :etble	<p>:CONTITEM</p> <table border="1"> <thead> <tr> <th>Attributes</th> <th>Values</th> <th>Uses</th> </tr> </thead> <tbody> <tr> <td>[BYTES =</td> <td>3 <i>field length in bytes</i>]</td> <td>Specifies the field length</td> </tr> <tr> <td>[DECIMALS =</td> <td>0 <i>number of decimals</i>]</td> <td>Specifies the number of decimal places</td> </tr> <tr> <td>NAME =</td> <td>* <i>item name</i></td> <td>Specifies a data item member</td> </tr> <tr> <td>[TYPE =</td> <td>BIN CHA DBCS HEX MIX NUM NUMC PACK PACF]</td> <td>Specifies the data item type</td> </tr> </tbody> </table> <p>[.]</p>	Attributes	Values	Uses	[BYTES =	3 <i>field length in bytes</i>]	Specifies the field length	[DECIMALS =	0 <i>number of decimals</i>]	Specifies the number of decimal places	NAME =	* <i>item name</i>	Specifies a data item member	[TYPE =	BIN CHA DBCS HEX MIX NUM NUMC PACK PACF]	Specifies the data item type
Attributes	Values	Uses														
[BYTES =	3 <i>field length in bytes</i>]	Specifies the field length														
[DECIMALS =	0 <i>number of decimals</i>]	Specifies the number of decimal places														
NAME =	* <i>item name</i>	Specifies a data item member														
[TYPE =	BIN CHA DBCS HEX MIX NUM NUMC PACK PACF]	Specifies the data item type														

BYTES =

3	Specifies 3 bytes (default value)
<i>field length in bytes</i>	Specifies the field length in bytes

BYTES is the number of bytes required to store a data item value internally. The following table lists the maximum length of data items:

Item Type	Byte Values
CHA, DBCS, Mixed	1-254
HEX	1-127
NUM, NUMC	1-18
PACK, PACF	1-10
BIN	2, 4, 8 only

DECIMALS =

0	Specifies no decimal places (default value)
<i>decimal places</i>	Specifies the number of decimal places

DECIMALS is the number of positions to the right of an implied decimal point in numeric items. The length specified for the data item must include the places set aside for decimal positions. The maximum number of decimal positions is 18 or the number of digits defined for the data item, whichever is smallest. The decimal point is not stored with the data.

NAME =

*	Specifies a filler data item
<i>item name</i>	Specifies the data item member

NAME consists of 1 to 32 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The name cannot contain blanks or have an EZE prefix.

The name can be a DBCS name up to 15 DBCS characters long with no embedded blanks.

A data item name in a table structure can be specified with an asterisk. This type of data item, called a filler, cannot be referred to by the application; it acts as a space holder.

Note: A filler is always a local data item.

TYPE =

BIN	Binary number
CHA	Character data (default value)
DBCS	Double-byte character data
HEX	Hexadecimal data
MIX	DBCS data intermingled with single-byte character data
NUM	Numeric characters with positive sign in F format
NUMC	Numeric characters with positive sign in C format
PACF	Packed decimal characters with positive sign in F format
PACK	Packed decimal characters with positive sign in C format

TYPE specifies the internal format or type of data. The data type determines how the item is processed when referred to in processing statements.

Row Specification

The ROW tag contains the contents of a single table row in character format. This tag is optional, but you can repeat it once for each row of data in the table. You cannot specify the ROW tag unless you specify the DEFITEM or CONTITEM tags. If you do not specify the CONTITEM tag, CSP/AD derives the contents structure from the DEFITEM tags. The order of the ROW tags determines the order of the rows in the table.

Syntax	Usage
:tbl : :contitem : :row : :etble	:ROW Value [. [<i>row content</i>]] Usage Contains the contents of one row in the table

Columns of data must be separated by one or more blanks. If the data needs to contain a blank, or starts with a single quotation mark, enclose the column in single quotation marks. When the data needs to contain quotation marks, use double sets of quotation marks. If the row is longer than 71 characters, place a continuation character in column 72. Any character in column 72 is

considered a continuation character. The continuation character causes concatenation of the two lines on import. The two lines concatenate with the character in column 71 being immediately followed by the character in column 1 of the next line. This continues until an external source format tag is detected in column 1.

Table Definition Syntax Example

```
:tbl name = xxxxxx .date = 'mm/dd/yy' time = 'hh:mm:ss'  
      tabtype = xxxxxxxxxxxx fold = x scope = xxxxxx.  
:genopts typeuse = xxxxxx loadlib = xxxxxxxx.
```

```
:prol.
```

Each prolog line can be up to 60 characters long so there is no need to worry about them splitting.

```
:eprol.
```

```
:defitem name = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
         level = xx type = xxxx bytes = xxxxx  
         decimals = xx desc = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'  
         evensql = x scope = xxxxxx.
```

```
:contitem name = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
         type = xxxx bytes = xxxxx decimals = xx.
```

```
:row.'These are the row contents. This can span more than one line and x  
is split on character boundaries because table data is just a stream of x  
characters. The next row starts with another :row in column 1.'
```

```
:row.Thisrowhasnosurroundingquotesbecausetheyarenotnecessaryiftherearenx  
oblanksorquotesintherowcontents
```

```
:row.'This is another row''s contents. It also spans more than one linx  
e and is split on character boundaries. The table contents are terminax  
ted with a :etble in column 1 that is coming up.
```

```
:etble.
```

Note: This example illustrates the syntax of all table definition tags and attributes. Actual definitions do not require or use all tags and attributes shown here.



Chapter 7. Data Item Structures

This chapter describes the external source format tags and attributes that define data items in CSP/AD.

Data Item Definition

The ITEM tag defines attributes for individual data item members. The MAPEDITS and MESSAGES tags further define aspects of an item and can have attributes. The EITEM tag closes the definition, and if you use the ITEM tag, the EITEM tag is required.

Syntax	Attributes		
:item	:ITEM		
:	Attributes	Values	Uses
:mapedits	[BYTES =	3 <i>field length in bytes</i>]	Specifies the length of the field
:	[DATE =	' <i>modification date</i> ']	Specifies the date the data item was last modified
:messages	[DECIMALS =	0 <i>decimal places</i>]	Specifies the number of decimal places
:	[DESC =	' <i>field description</i> ']	Describes the field
:eitem	[EVENSQ =	Y N]	Specifies whether packed fields of even length or odd length are passed to a relational database
	NAME =	<i>item name</i>	Specifies a data item member
	[TIME =	' <i>modification time</i> ']	Specifies the time the data item was last modified
	[TYPE =	BIN CHA DBCS HEX MIX NUM NUMC PACF PACK]	Specifies the data item type
	[.]		
	:EITEM[.]		

BYTES =	
3	Specifies 3 bytes (default value)
<i>field length in bytes</i>	Specifies the field length in bytes

DATE =	
' <i>modification date</i> '	Specifies the date the data item member was last modified

BYTES is the number of bytes required to store a data item value internally. The following table lists the maximum length of data items:

Item Type	Byte Values
CHA, HEX, Mixed	1-32,767
DBCS	1-32,766
NUM, NUMC	1-18
PACK, PACF	1-10
BIN	2, 4, 8 only

DATE format is mm/dd/yy. If the external source format file was produced by CSP/AD export, this date shows when the member was last modified in the MSL. The date is ignored on import.

DECIMALS =

0	Specifies no decimal places (default value)
<i>decimal places</i>	Specifies the number of decimal places

DECIMALS is the number of positions to the right of an implied decimal point in numeric items. The length specified for the data item must include the places set aside for decimal positions. The maximum number of decimal positions is 18 or the number of digits defined for the data item, whichever is smaller. The decimal point is not stored with the data.

DESC =

<i>'field description'</i>	Describes what the data item represents
----------------------------	---

DESC is a text description of what the data item represents. The text can be up to 30 characters, not including the quotes.

EVENSQ =

Y	Specifies whether packed fields of even length are passed to a relational database
N	Specifies whether packed fields of odd length are passed to a relational database (default value)

EVENSQ indicates whether CSP/AD passes either even- or odd-length packed fields to the relational database. Specify EVENSQ as Y only if you are using an SQL table that has even-length columns defined.

NAME =

<i>name</i>	Specifies the data item member
-------------	--------------------------------

NAME consists of 1 to 32 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The name cannot contain blanks or have an EZE prefix.

The name can be a DBCS name up to 15 DBCS characters long with no embedded blanks.

TIME =

<i>'modification time'</i>	Specifies the time the data item member was last modified
----------------------------	---

TIME format is hh:mm:ss. If the external source format file was produced by CSP/AD export, this time shows when the member was last modified in the MSL. The time is ignored on import.

TYPE =

BIN	Binary number
CHA	Character data (default value)
DBCS	Double-byte character data
HEX	Hexadecimal data
MIX	DBCS data intermingled with single-byte character data
NUM	Numeric characters with positive sign in F format
NUMC	Numeric characters with positive sign in C format
PACF	Packed decimal characters with positive sign in F format
PACK	Packed decimal characters with positive sign in C format

TYPE specifies the internal format or type of data. The data type determines how the item is processed when referred to in processing statements.

Map Edit Characteristic Definition

The MAPEDITS tag defines the default map edit characteristics for a data item. This optional tag with its attributes further describes the ITEM and EITEM tag set.

Syntax	Attributes																																												
<pre> : item . : mapedits . : eitem </pre>	<p>:MAPEDITS</p> <table border="1"> <thead> <tr> <th style="text-align: left;">Attributes</th> <th style="text-align: left;">Values</th> <th style="text-align: left;">Uses</th> </tr> </thead> <tbody> <tr> <td>[CURRSYMB =</td> <td>Y N]</td> <td>Specifies whether the currency symbol is supported</td> </tr> <tr> <td>[DATEFORM =</td> <td><i>number</i>]</td> <td>Provides date format editing</td> </tr> <tr> <td>[EDITRTN =</td> <td><i>edit routine</i>]</td> <td>Specifies a routine or edit table for special data editing</td> </tr> <tr> <td>[FILLCHAR =</td> <td>'N' <i>fill character</i>]</td> <td>Specifies the character used to fill unused data item positions</td> </tr> <tr> <td>[FLDFOLD =</td> <td>Y N]</td> <td>Specifies whether data entered in this field is folded</td> </tr> <tr> <td>[HEXEDIT =</td> <td>Y N]</td> <td>Specifies whether only hexadecimal digits can be entered in the input field</td> </tr> <tr> <td>[INPUTREQ =</td> <td>Y N]</td> <td>Specifies whether valid data must be entered</td> </tr> <tr> <td>[JUSTIFY =</td> <td>LEF RIG N]</td> <td>Specifies the position of data when it is shorter than the length of the field</td> </tr> <tr> <td>[MININPUT =</td> <td>N <i>positions</i>]</td> <td>Specifies the minimum number of required characters</td> </tr> <tr> <td>[NUMSEP =</td> <td>Y N]</td> <td>Specifies whether data can contain numeric separators</td> </tr> <tr> <td>[RANGE =</td> <td><i>low value</i> <i>high value</i>]</td> <td>Specifies the range of valid numeric values</td> </tr> <tr> <td>[SIGN =</td> <td>LEA TRA N]</td> <td>Specifies a sign in a field as leading, trailing, or none</td> </tr> <tr> <td>[ZEROEDIT =</td> <td>Y N]</td> <td>Specifies the format for numeric fields that have zero values</td> </tr> </tbody> </table> <p>[.]</p>			Attributes	Values	Uses	[CURRSYMB =	Y N]	Specifies whether the currency symbol is supported	[DATEFORM =	<i>number</i>]	Provides date format editing	[EDITRTN =	<i>edit routine</i>]	Specifies a routine or edit table for special data editing	[FILLCHAR =	'N' <i>fill character</i>]	Specifies the character used to fill unused data item positions	[FLDFOLD =	Y N]	Specifies whether data entered in this field is folded	[HEXEDIT =	Y N]	Specifies whether only hexadecimal digits can be entered in the input field	[INPUTREQ =	Y N]	Specifies whether valid data must be entered	[JUSTIFY =	LEF RIG N]	Specifies the position of data when it is shorter than the length of the field	[MININPUT =	N <i>positions</i>]	Specifies the minimum number of required characters	[NUMSEP =	Y N]	Specifies whether data can contain numeric separators	[RANGE =	<i>low value</i> <i>high value</i>]	Specifies the range of valid numeric values	[SIGN =	LEA TRA N]	Specifies a sign in a field as leading, trailing, or none	[ZEROEDIT =	Y N]	Specifies the format for numeric fields that have zero values
Attributes	Values	Uses																																											
[CURRSYMB =	Y N]	Specifies whether the currency symbol is supported																																											
[DATEFORM =	<i>number</i>]	Provides date format editing																																											
[EDITRTN =	<i>edit routine</i>]	Specifies a routine or edit table for special data editing																																											
[FILLCHAR =	'N' <i>fill character</i>]	Specifies the character used to fill unused data item positions																																											
[FLDFOLD =	Y N]	Specifies whether data entered in this field is folded																																											
[HEXEDIT =	Y N]	Specifies whether only hexadecimal digits can be entered in the input field																																											
[INPUTREQ =	Y N]	Specifies whether valid data must be entered																																											
[JUSTIFY =	LEF RIG N]	Specifies the position of data when it is shorter than the length of the field																																											
[MININPUT =	N <i>positions</i>]	Specifies the minimum number of required characters																																											
[NUMSEP =	Y N]	Specifies whether data can contain numeric separators																																											
[RANGE =	<i>low value</i> <i>high value</i>]	Specifies the range of valid numeric values																																											
[SIGN =	LEA TRA N]	Specifies a sign in a field as leading, trailing, or none																																											
[ZEROEDIT =	Y N]	Specifies the format for numeric fields that have zero values																																											

CURRSYMB =

Y	Specifies that the currency symbol is supported
N	Specifies that the currency symbol is not supported (default value)

CURRSYMB indicates whether the currency symbol is supported in the field. Y is valid only for numeric fields. When defining field length, remember that the currency symbol takes up one position.

DATEFORM =*number* Provides date format editing

DATEFORM indicates which date format is used when data is entered or displayed as follows.

Number	Format	Example
1	MM/DD/YY	06/30/89
2	MM-DD-YY	06-30-89
3	MM:DD:YY	06:30:89
4	YY/MM/DD	89/06/30
5	YY-MM-DD	89-06-30
6	YY:MM:DD	89:06:30
7	DD/MM/YY	30/06/89
8	DD-MM-YY	30-06-89
9	DD:MM:YY	30:06:89
10	YY-DDD	89-181
11	YY:DDD	89:181

Field edits should be defined as follows when you specify a date edit option:

- TYPE = NUM
- DECIMALS = 0
- CURRSYMB = N
- NUMSEP = N
- SIGN = N.

EDITRTN =*edit routine* Specifies a routine or edit table for special editing of data in a variable field

EDITRTN indicates the name of a routine or edit table used for special editing of data that a user entered in a variable field. You can specify one of the following:

- The name of one of the following types of editing tables:
 - Match valid table
 - Match invalid table
 - Range match valid table.

- The name of one of the following special function word subroutines:
 - Modulus 10 check digit routine (EZEC10)
 - Modulus 11 check digit routine (EZEC11).

- The name of a statement group used as an edit routine.

FILLCHAR =

'N' Specifies a null fill character (default value)

'fill character' Specifies the character used to fill unused data item positions

FILLCHAR indicates the character used to fill unused field positions on output to the terminal or printer. The default fill character for CHA, MIX, and DBCS fields is a blank. The default for HEX fields is a 0. You must enclose all values for FILLCHAR in quotes.

FLDFOLD =

Y Specifies that data entered in this field is folded

N Specifies that data entered in this field is not folded (default value)

FLDFOLD specifies whether lowercase alphabetic characters that the user enters are to be folded (converted) to uppercase.

HEXEDIT =

Y Specifies that only hexadecimal digits can be entered

N Specifies that the map field is not checked for hexadecimal characters (default value)

HEXEDIT specifies whether the input field is checked for hexadecimal digits. The data type of the variable field must be CHA or HEX for Y to be specified.

INPUTREQ =

Y	Specifies that valid data must be entered in a map field
N	Specifies that input is not required in the field (default value)

INPUTREQ indicates whether valid data must be entered in a map field. When a field contains data other than blanks for character type or 0s for numeric types, the field is considered to have input.

Even if blanks for character fields and 0s for numeric fields are valid values, INPUTREQ will return an error message unless the user types data into the field.

JUSTIFY =

LEF	Specifies left justification (default for character data)
RIG	Specifies right justification (default for numeric data)
N	Specifies no justification

JUSTIFY specifies the position of data in a variable field when the data is shorter than the length of the field. If JUSTIFY is not specified, character data is left justified and numeric data is right justified. Right justification is required for numeric data with decimal positions specified. JUSTIFY = N is not valid for numeric fields.

MININPUT =

N	Specifies no minimum number of characters required (default value)
<i>positions</i>	Specifies the minimum number of characters that must be entered in a valid variable field

MININPUT specifies the minimum number of characters that must be entered in a variable field if any data is entered.

NUMSEP =

Y	Specifies that data can contain numeric separators
N	Specifies that data cannot contain numeric separators (default value)

NUMSEP specifies whether data in a field can contain the previously defined numeric separator. You can specify Y for numeric fields only. When defining field length, remember that each numeric separator takes up one position.

RANGE =

<i>low value</i>	Specifies the smallest numeric value for a field
<i>high value</i>	Specifies the largest numeric value for a field

RANGE specifies the range of valid numeric values for a field. Low value is the smallest numeric value that can be entered in a specified field. High value is the largest numeric value that can be entered in a specified field. The high and low values must have the same length, number of decimal positions, and sign as defined for the field. Both low and high values must be specified if this attribute is used. The values are separated by a space and cannot be longer than the number of spaces in the field being defined. RANGE can be specified only for numeric fields.

SIGN =

LEA	Specifies a leading sign (default value for numeric fields)
TRA	Specifies a trailing sign
N	Specifies no sign (default value for character fields)

SIGN specifies whether a sign is displayed in a field and whether it is a leading or a trailing sign. You can specify signs for numeric fields only. When defining field length, remember that the sign takes up one position.

ZEROEDIT =	
Y	Specifies the display format for numeric fields containing zero values
N	Specifies that no editing is done on 0 numeric fields (default value)

ZEROEDIT specifies the display format for numeric fields that have 0 values. The following table gives a list of what a numeric field will contain when ZEROEDIT is specified with either Y or N. The sample field has been defined as right justified with a length of 11. A "b" represents a blank fill character.

DEC (2)	CURRSYMB / NUMSEP	FILL	ZEROEDIT = N	ZEROEDIT = Y
N	N	N	nulls	0
N	N	b	blanks	bbbbbbbbb0
N	N	0	00000000000	00000000000
N	N	.	*****	*****0
Y	N	N	nulls	0.00
Y	N	b	blanks	bbbbbb0.00
Y	N	0	00000000000	00000000.00
Y	N	.	*****	*****0.00
Y	Y	N	nulls	\$0.00
Y	Y	b	blanks	bbbbbb\$0.00
Y	Y	0	00000000000	\$000,000.00
Y	Y	.	*****	*****\$0.00

Data Item Message Definition

The MESSAGES tag specifies the data item messages from a user-defined message file. This optional tag with its attributes further describes the ITEM and EITEM tag set.

Message numbers must be in the range of 1 to 9999.

Syntax	Attributes																				
<pre> :item : :messages : :eitem </pre>	<p>:MESSAGES</p> <table border="1"> <thead> <tr> <th style="text-align: left;">Attributes</th> <th style="text-align: left;">Values</th> <th style="text-align: left;">Uses</th> </tr> </thead> <tbody> <tr> <td>[EDITMSG =</td> <td><i>message number</i></td> <td>Specifies the message number of the edit routine error message</td> </tr> <tr> <td>[INVALMSG =</td> <td><i>message number</i></td> <td>Specifies the message number of the data type error message</td> </tr> <tr> <td>[MININMSG =</td> <td><i>message number</i></td> <td>Specifies the message number of the minimum input error message</td> </tr> <tr> <td>[RANGEMSG =</td> <td><i>message number</i></td> <td>Specifies the message number of the value error message</td> </tr> <tr> <td>[REQMSG =</td> <td><i>message number</i></td> <td>Specifies the message number of the input required error message</td> </tr> </tbody> </table> <p>[.]</p>			Attributes	Values	Uses	[EDITMSG =	<i>message number</i>	Specifies the message number of the edit routine error message	[INVALMSG =	<i>message number</i>	Specifies the message number of the data type error message	[MININMSG =	<i>message number</i>	Specifies the message number of the minimum input error message	[RANGEMSG =	<i>message number</i>	Specifies the message number of the value error message	[REQMSG =	<i>message number</i>	Specifies the message number of the input required error message
Attributes	Values	Uses																			
[EDITMSG =	<i>message number</i>	Specifies the message number of the edit routine error message																			
[INVALMSG =	<i>message number</i>	Specifies the message number of the data type error message																			
[MININMSG =	<i>message number</i>	Specifies the message number of the minimum input error message																			
[RANGEMSG =	<i>message number</i>	Specifies the message number of the value error message																			
[REQMSG =	<i>message number</i>	Specifies the message number of the input required error message																			

EDITMSG =

message number Specifies the number of the message displayed when the data fails a modulus check or table edit check

If you do not specify this attribute, the default error message for a modulus check is "Modulus check error—reenter," and the default error message for a table edit check is "Table edit validity error—reenter."

INVALMSG =

message number Specifies the message number of the message displayed when the data entered is incompatible with the variable field data type

If you do not specify this attribute, the default error message for invalid data type is "Data type error in input—reenter."

MININMSG =

message number Specifies the number of the message displayed when the minimum input edit check fails

If you do not specify this attribute, the default error message for this error is "Input minimum length error—reenter."

RANGEMSG =

message number Specifies the number of the message displayed when the minimum or maximum value check fails

If you do not specify this attribute, the default error message is "Input not within defined range—reenter."

REQMSG=

<i>message number</i>	Specifies the number of the message displayed when the input required edit check fails
-----------------------	--

If you do not specify this attribute, the default error message for this error is "No input received for required field—reenter."



Data Item Definition Syntax Example

```
:item      name = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
           date = 'mm/dd/yy'   time   = 'hh:mm:ss' type   = xxxx
           bytes = xxxxx       decimals = xx       evensql = x
           desc  = 'A 30 character description'.
:mapedits  fldfold = x
           range = xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx
           mininput = xx
           fillchar = 'x'
           editrtn = xxxxxxxxxxxxxxxxxxxxxxxx
           dateform = xx   currsymb = x   sign   = xxx numsep = x
           inputreq = x   justify = xxx hexedit = x   zeroedit = x.
:messages  invalmsg = xxxx
           mininmsg = xxxx
           reqmsg = xxxx
           editmsg = xxxx
           rangemsg = xxxx.
:eitem.
```

Note: This example illustrates the syntax of all data item definition tags and attributes. Actual definitions do not require or use all tags and attributes shown here.



Chapter 8. Program Specification Block Structures

This chapter describes the external source format tags and attributes that define a program specification block (PSB) structure in CSP/AD.

PSB Definition

The PSB tag and its attributes define PSB members. The PCB and SENSEG tags further define aspects of a PSB member and can have attributes. The EPSB tag closes the definition, and if you use the PSB tag, the EPSB tag is required.

The PSB tag specifies a set of DL/I database structures that an application can access. The PSB MSL member contains a subset of the information in a DL/I PSB.

Syntax	Attributes	Values	Uses
:psb	:PSB		
:			
:pcb	Attributes		
:			
:senseg	[DATE =	'modification date']	Specifies the date the PSB member was last modified
:			
:epsb	NAME =	name	Specifies a PSB member
	[TIME =	'modification time']	Specifies the time the PSB member was last modified
	[.]		
	:EPSB[.]		

DATE =	
'modification date'	Specifies the date the PSB member was last modified

DATE format is mm/dd/yy. If the external source format file was produced by CSP/AD export, this date indicates when the PSB member was last modified in the MSL. The date is ignored on import.

NAME =	
name	Specifies the PSB member

NAME specifies a PSB member. It consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic (A-Z).
- The remaining characters must be alphanumeric (A-Z, 0-9).
- The name cannot contain blanks or have an EZE prefix.

TIME =	
'modification time'	Specifies the time the PSB member was last modified

TIME format is hh:mm:ss. If the external source format file was produced by CSP/AD export, the time indicates when the PSB member was last modified in the MSL. The time is ignored on import.

Program Communication Block Specification

The PCB tag specifies a program communication block entry in the DL/I program specification block. This tag further describes the PSB and EPSB tag set.

The PCB tag can be repeated once for each program communications block in the DL/I program specification block.

Syntax	Attributes		
:psb :	:PCB		
:pcb :	Attributes	Values	Uses
:senseg :	[DBNAME =	<i>database name</i>]	Specifies a database used with the PSB
:epsb :	[TYPE =	TP DB GSAM]	Specifies the type of PCB
	[.]		

DBNAME =	
<i>database name</i>	Specifies a database used with the PSB

TYPE =	
TP	Specifies a teleprocessing PCB
DB	Specifies a database PCB (default value)
GSAM	Specifies a GSAM PCB

DBNAME signifies the start of a new database PCB in the PSB. DBNAME is required for DB and GSAM PCBs and cannot be specified for TP PCBs.

DBNAME consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, \$, #, @).
- The database name cannot contain blanks or have an EZE prefix.

TYPE specifies the type of PCBs.

Specify PCBs in the following order:

1. TP
2. DB
3. GSAM.

Note: If you specify TP, you must specify at least two TP PCBs.

Segment Sensitivity Specification

The SENSEG tag specifies segment sensitivity for a PCB entry in the DL/I PSB. This tag is required for TYPE DB only and its attributes further describe the PSB and EPSB tag set. SENSEG is not allowed for teleprocessing (TP) and GSAM PCBs.

The SENSEG tag can be repeated for each sensitive segment in a DB PCB. The order of the SENSEG tags determines the order of the sensitive segments in the PCB.

Syntax	Attributes		
:psb :	:SENSEG		
:pcb :	Attributes	Values	Uses
:senseg :	{IKEY =	<i>secondary index field name</i>	Specifies the secondary index key field
:epsb :	{PARENT =	0 <i>parent name</i>	Specifies the segment's parent segment
	SEGMENT =	<i>segment name</i>	Specifies a segment in the database
	[.]		

IKEY =
secondary index field name
 Specifies the secondary index key field

PARENT =
 0 Specifies 0 as the parent (only valid for the root segment)
parent name Specifies the segment's parent segment

IKEY must be the name of an item defined in the segment or in working storage. IKEY must also be the same name defined in the NAME operand in the XDFLD statement that defines the secondary index field in the DL/I database description (DBD). You can specify IKEY for the root segment only.

IKEY consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, #, \$, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, #, \$, @).
- The secondary index field name cannot contain blanks or have an EZE prefix.

PARENT must be the name defined in the PARENT operand in the SENSEG statement in the DL/I PSB.

PARENT consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, \$, #, @).
- The parent name cannot contain blanks or have an EZE prefix.

SEGMENT =

segment name Specifies a segment in the database

SENSEG must be the same as defined in the NAME operand in the SENSEG statement in the DL/I PSB.

SEGMENT consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, \$, #, @).
- The segment name cannot contain blanks or have an EZE prefix.



PSB Definition Syntax Example

```
:psb      name      = xxxxxxxx   date   = 'mm/dd/yy'  time = 'hh:mm:ss'.
:pcb      dbname    = xxxxxxxx   type   = xxxx.
:senseg   segment   = xxxxxxxx   parent = 0          ikey = xxxxxxxx.
:pcb      dbname    = xxxxxxxx   type   = xxxx.
:senseg   segment   = xxxxxxxx   parent = 0          ikey = xxxxxxxx.
:senseg   segment   = xxxxxxxx   parent = xxxxxxxx.
:pcb      dbname    = ELAWORK     type   = xxxx.
:pcb      dbname    = ELAMSG     type   = xxxx.
:pcb      dbname    = xxxxxxxx   type   = xxxx.
:epsb.
```

Note: This example illustrates the syntax of all PSB definition tags and attributes. Actual definitions do not require or use all tags and attributes shown here.



Chapter 9. Map Structures

This chapter describes the external source format tags and attributes that define map structures in CSP/AD.

Map Definition

The MAP tag defines attributes for individual map members. The PRESENT, CFIELD, CATTR, VFIELD, MAPEDITS, MESSAGES, and VATTR tags further define aspects of a map and can have attributes. The EMAP tag closes the definition.

Naming fields on a map member does not define the associated data-item members or create the data-item members for fields that are defined in external source format when the map is edited with Map Definition.

The CFIELD and VFIELD tags describe constant and variable fields on the map. You must describe the fields in the order that their field markers appear on the map (top to bottom, left to right). Therefore, the order of CFIELD and VFIELD tags depends on the constant and variable fields defined on the map.

Syntax	Attributes	Values	Uses
:map	:MAP		
:	Attributes		
:present	[BYPKEY =	<i>number list</i>]	Specifies up to five PF keys that allow the user to bypass map edits and map edit groups
:			
:cfield	[DATE =	<i>'modification date'</i>]	Specifies the date the map was last modified
:			
:cattr	DEVICES =	<i>device names</i>	Specifies the device or devices for which the map is defined
:			
:ecfield	GRPNAME =	<i>name</i>	Identifies a group of maps used with an application
:			
:vfield	[HELPKEY =	<i>number</i>]	Specifies the PF key that displays help
:			
:mapedits	[HELPMAP =	<i>map name</i>]	Specifies the name of a user-defined help map
:			
:messages	MAPNAME =	<i>name</i>	Identifies the map within a map group
:			
:vattr	[MAPSIZE =	<i>lines columns</i>]	Specifies the number of lines and columns for the map
:			
:evfield	[SOSIPOS =	Y [N]	Specifies whether SO and SI delimiters take a position (this value is for DBCS only)
:			
:emap	[STARTPOS =	<i>line column</i> [1 1] NEXT SAME]	Specifies the starting position of the map
:			
	[TIME =	<i>'modification time'</i>]	Specifies the time the map was last modified
	[.]		
	:EMAP[.]		

BYPKEY =

number list Specifies up to five PF keys that allow the user to bypass map edits and map edit groups

Specify BYPKEY keys as integers from 1 to 24. Separate multiple keys with blanks. No default bypass edit PF keys are designated.

Note: Specifying the bypass edit PF keys for a map overrides the application specification bypass PF keys for that map.

DATE =

'modification date' Specifies the date the map member was last modified

DATE format is mm/dd/yy. If the external source format file was produced by CSP/AD export, this date shows when the member was last modified in the MSL. The date is ignored on import.

DEVICES =

device names Specifies the device or devices for which the map is defined

You must specify at least one device. If you specify multiple devices, you must separate them with a blank. The devices you specify must be compatible with each other.

Devices are considered compatible when they are of the same type (display, printer, DBCS display, or DBCS printer), and the map being defined fits within the row and column limitations of the smallest physical device.

The following table lists the valid devices and their sizes.

Supported IBM Devices	Rows	Columns
3643-2	6	40
3277-1	12	40
3643-4	16	64
3278-1 3278-1B 8775-1C ANY-1D	12	80
3278-2 3278-2B 8775-2C ANY-2D	24	80
5550D (DBCS display device)	24	80
3278-3 3278-3B 8775-3C ANY-3D	32	80
3278-4 3278-4B 8775-4C ANY-4D	43	80
3278-5 3278-5B ANY-5D	27	132
ANY-D (3290 Configured as 62x160)	255	160
3767 PRINT-B PRINTER (physical size 66x132)	255	132
5550P (DBCS printer with size 66x158)	255	158

GRPNAME =

name Identifies a group of maps used with an application

GRPNAME identifies a group of maps used with one application. It consists of 1 to 6 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, #, \$, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, #, \$, @).
- The name cannot contain blanks or have an EZE prefix.

HELPKEY =

number Specifies the number of the PF key that displays help

Specify HELPKEY as an integer ranging from 1 to 24. If you do not specify this attribute, the help key defined for the application is used. You can only specify a help key if a help map name is defined.

HELPMAP =

<i>map name</i>	Specifies the name of a user-defined help map
-----------------	---

HELPMAP must be in the help map group or the main map group specified for the application. The help map cannot have variable fields, and it cannot be a floating map. If the help map is a partial map, it replaces the full screen. The help map must be for a display device and not a printer.

MAPNAME =

<i>name</i>	Identifies the map within a map group
-------------	---------------------------------------

MAPNAME consists of 1 to 8 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, \$, #, @).
- The name cannot contain blanks or have an EZE prefix.

MAPSIZE =

<i>lines</i>	Specifies the number of lines
<i>columns</i>	Specifies the number of columns

MAPSIZE defaults to the screen size or print line length for the device specified for the map that has the smallest screen size or line length. Printer maps default to a size of 66 lines but can be made larger. You can specify the size as less than the default value to define a partial map.

SOSIPOS =

Y	Specifies that SO/SI positions will be represented by blanks in the printed output (default value)
N	Specifies that SO/SI positions will not be represented by blanks in the printed output

SOSIPOS specifies whether shift-out (SO) and shift-in (SI) delimiters take a position when printing mixed data from an application. When directing printer output containing DBCS or mixed data to a system printer, specify SOSIPOS = N if you do not want SO and SI to take a position.

STARTPOS =

<i>line</i>	Specifies the line coordinate where the map starts
<i>column</i>	Specifies the column coordinate where the map starts
1	Specifies line one (default value)
1	Specifies column one (default value)
NEXT	Specifies the next available line within the floating area specified on the AREA tag (in map group specification)
SAME	Specifies the next available column within the floating area specified on the AREA tag (in map group specification)

STARTPOS specifies the map's starting position on a screen using a line and column coordinate. Partial maps (maps smaller than the screen size) can share the same screen display if their position allows all of them to be vertically contained within the screen area.

TIME =

<i>'modification time'</i>	Specifies the time the map member was last modified
----------------------------	---

TIME format is hh:mm:ss. If the external source format file was produced by CSP/AD export, this time shows when the member was last modified in the MSL. The time is ignored on import.

Presentation Information Definition

The PRESENT tag defines the presentation information used for the map. This optional tag with its attributes further describes the MAP and EMAP tag set.

Note: Field markers indicate the beginning of a field on a map. Field markers cannot be a numeric character or a semicolon.

Syntax	Attributes																																			
<pre>:map : :present : :emap</pre>	<p>:PRESENT</p> <table border="1"> <thead> <tr> <th style="text-align: left;">Attributes</th> <th style="text-align: left;">Values</th> <th style="text-align: left;">Uses</th> </tr> </thead> <tbody> <tr> <td>[CONSTANT =</td> <td>'#' <i>'character'</i></td> <td>Specifies the field marker that signifies the beginning of a constant field</td> </tr> <tr> <td>[DBCSCONS =</td> <td>'+' <i>'character'</i></td> <td>Specifies the field marker that signifies the beginning of a DBCS constant field</td> </tr> <tr> <td>[DBCVAR =</td> <td>'@' <i>'character'</i></td> <td>Specifies the field marker that signifies the beginning of a DBCS variable field</td> </tr> <tr> <td>[DEFFOLD =</td> <td>Y N</td> <td>Specifies whether character data is folded during online map definition processing</td> </tr> <tr> <td>[MIXCONS =</td> <td>'%' <i>'character'</i></td> <td>Specifies the field marker that signifies the beginning of a MIX constant field</td> </tr> <tr> <td>[MIXVAR =</td> <td>'!' <i>'character'</i></td> <td>Specifies the field marker that signifies the beginning of a MIX variable field</td> </tr> <tr> <td>[SPACER =</td> <td>'J' <i>'character'</i></td> <td>Specifies a field marker that provides equal spacing between fields</td> </tr> <tr> <td>[TABPOS =</td> <td><i>number list</i></td> <td>Specifies tab positions used during map definition</td> </tr> <tr> <td>[VARFOLD =</td> <td>Y N</td> <td>Specifies whether character data or single-byte data in mixed fields is folded at execution time</td> </tr> <tr> <td>[VARIABLE =</td> <td>'-' <i>'character'</i></td> <td>Specifies a field marker that signifies the beginning of a variable field</td> </tr> </tbody> </table> <p>[.]</p>			Attributes	Values	Uses	[CONSTANT =	'#' <i>'character'</i>	Specifies the field marker that signifies the beginning of a constant field	[DBCSCONS =	'+' <i>'character'</i>	Specifies the field marker that signifies the beginning of a DBCS constant field	[DBCVAR =	'@' <i>'character'</i>	Specifies the field marker that signifies the beginning of a DBCS variable field	[DEFFOLD =	Y N	Specifies whether character data is folded during online map definition processing	[MIXCONS =	'%' <i>'character'</i>	Specifies the field marker that signifies the beginning of a MIX constant field	[MIXVAR =	'!' <i>'character'</i>	Specifies the field marker that signifies the beginning of a MIX variable field	[SPACER =	'J' <i>'character'</i>	Specifies a field marker that provides equal spacing between fields	[TABPOS =	<i>number list</i>	Specifies tab positions used during map definition	[VARFOLD =	Y N	Specifies whether character data or single-byte data in mixed fields is folded at execution time	[VARIABLE =	'-' <i>'character'</i>	Specifies a field marker that signifies the beginning of a variable field
Attributes	Values	Uses																																		
[CONSTANT =	'#' <i>'character'</i>	Specifies the field marker that signifies the beginning of a constant field																																		
[DBCSCONS =	'+' <i>'character'</i>	Specifies the field marker that signifies the beginning of a DBCS constant field																																		
[DBCVAR =	'@' <i>'character'</i>	Specifies the field marker that signifies the beginning of a DBCS variable field																																		
[DEFFOLD =	Y N	Specifies whether character data is folded during online map definition processing																																		
[MIXCONS =	'%' <i>'character'</i>	Specifies the field marker that signifies the beginning of a MIX constant field																																		
[MIXVAR =	'!' <i>'character'</i>	Specifies the field marker that signifies the beginning of a MIX variable field																																		
[SPACER =	'J' <i>'character'</i>	Specifies a field marker that provides equal spacing between fields																																		
[TABPOS =	<i>number list</i>	Specifies tab positions used during map definition																																		
[VARFOLD =	Y N	Specifies whether character data or single-byte data in mixed fields is folded at execution time																																		
[VARIABLE =	'-' <i>'character'</i>	Specifies a field marker that signifies the beginning of a variable field																																		

CONSTANT =

'#'	Default constant field marker
<i>'character'</i>	Specifies a constant field marker

CONSTANT is the field marker that signifies the beginning of a constant field.

DBCSCONS is the field marker that signifies the beginning of a DBCS constant field.

DBCVAR =

'@'	Default DBCS variable field marker
<i>'character'</i>	Specifies a DBCS variable field marker

DBCVAR is the field marker that signifies the beginning of a DBCS variable field.

DBCSCONS =

'+'	Default DBCS constant field marker
<i>'character'</i>	Specifies a DBCS constant field marker

DEFFOLD =

Y	Specifies that character data is folded during map presentation processing
N	Specifies that character data is not folded during map presentation processing (default value)

DEFFOLD indicates whether characters entered in the Map Presentation Display of CSP/AD Map Definition will be folded. Folding affects online definition processing only. See FLDFOLD on page 9-13 and VARFOLD for information on how to affect execution processing.

MIXCONS =

'%'	Default MIX constant field marker
'character'	Specifies a MIX constant field marker

MIXCONS is the field marker that signifies the start of a MIX constant field.

MIXVAR =

' '	Default MIX variable field marker
'character'	Specifies a MIX variable field marker

MIXVAR is the field marker that signifies the beginning of a MIX variable field.

SPACER =

' '	Default spacer marker
'character'	Specifies a spacer marker

SPACER is the field marker used by map definition to automatically provide equal spacing between fields on a line of a map.

TABPOS =

<i>number list</i>	Specifies tab positions used during map definition
--------------------	--

TABPOS allows you to specify ten tab positions; import sorts them into ascending order. You must separate tab positions with blanks. The default is one tab of 1.

VARFOLD =

Y	Specifies that all character data and all single-byte data in mixed fields on the map will be folded to uppercase (default value)
N	Specifies that character data and single-byte data in mixed fields remains as entered (uppercase and lowercase) or will be folded on a field-to-field basis as specified on the MAPEDITS tag for each VFIELD.

VARFOLD specifies whether all single-byte data entered on a map during execution is folded to uppercase. Folding does not affect numeric or DBCS data.

VARIABLE =

' '	Default variable field marker
'character'	Specifies a variable field marker

VARIABLE is the field marker that signifies the beginning of a variable field.

Constant Field Definition

The CFIELD tag specifies information about a constant field on the map. The CFIELD and CATTR tags are a set of tags that describe a single constant field and must be repeated for each constant field on the map. The CATTR tag is always optional. The ECFIELD tag closes the definition and is required for each CFIELD.

Syntax	Attributes		
:map	:CFIELD		
:	Attributes	Values	Uses
:cfield	BYTES =	<i>field length in bytes</i>	Specifies the number of positions that the constant field occupies
:	COLUMN =	<i>column number</i>	Specifies the column of the constant field marker
:cattr	ROW =	<i>row number</i>	Specifies the row of the constant field marker
:	[TYPE =	CHA DBCS MIX]	Specifies the data type
:ecfield	[. <i>field text</i>]		Specifies the value of the constant field
:emap	:ECFIELD[.]		

BYTES =

field length in bytes Specifies the number of positions that the constant field occupies

field text

field text Specifies the value of the constant field

BYTES does not include the field marker.

COLUMN =

column number Specifies the column of the constant field marker

ROW =

row number Specifies the row of the constant field marker

TYPE =

CHA Specifies any character data
DBCS Specifies double-byte character set, ideographic character data that requires two positions for each character
MIX Specifies DBCS and single-byte data in the same field

Field text begins in the next available byte immediately following the period. The period is required if field text is supplied. If a line of text in the external source format file is longer than 71 characters, place a continuation character in column 72. Any character in column 72 is a continuation character. The continuation character causes concatenation of the two lines on import. If a continuation character is in column 72 and an external source format tag is in column 1 of the next line, the tag is interpreted as data instead of as a tag. If no continuation character is in column 72 and an external source format tag is in column 1 of the next line, the tag is interpreted as an external source format tag. If quotes enclose the field text, the quotes are considered to be part of the text.

Constant Field Attribute Definition

The CATTR tag defines attributes for the constant described in the CFIELD tag. This optional tag with its attributes further describes the MAP and EMAP tag set.

Syntax	Attributes	Values	Uses
:map :	:CATTR		
:cfield :	Attributes		
:cattr :	[COLOR =	MONO BLUE RED PINK GREEN TURQUOISE YELLOW WHITE]	Specifies the color of the field
:ecfield :	[CURSOR =	Y N]	Specifies if the cursor will be positioned on this field
:emap	[DATA =	ALPHA NUMERIC]	Specifies the field data type
	[DETECT =	Y N]	Specifies whether the field is light pen detectable
	[ENTER =	Y N]	Specifies if data is required to be entered into a field
	[FILL =	Y N]	Specifies whether an error occurs when a user does not enter enough characters to fill a field
	[HILITE =	NOHILITE BLINK RVIDEO USCORE]	Specifies the highlighting characteristics of a constant field
	[INTENSE =	NORMAL DARK BRIGHT]	Specifies the light intensity of a constant field
	[MDT =	Y N]	Controls the default setting of the modified data tag indicator
	[OUTLINE =	{NOOUTLINE BOX [ORIGHT OLEFT][OOVER][OUNDER}]}	Specifies if lines are to be drawn at the edges of fields on DBCS devices
	[PROTECT =	UNPROTECT PROTECT ASKIP]	Specifies whether data can be entered into the field
	[.]		

COLOR =

MONO	Specifies monochrome (default value)
BLUE	Specifies blue
RED	Specifies red
PINK	Specifies pink
GREEN	Specifies green
TURQUOISE	Specifies turquoise
YELLOW	Specifies yellow
WHITE	Specifies white

CURSOR =

Y	Specifies that the cursor will be positioned on this field when the map is displayed
N	Specifies that the cursor will not be positioned in this field when the map is displayed (default value)

The cursor attribute can be overridden by the application using the SET map item CURSOR processing statement. The default is the first unprotected variable field on the display.

COLOR specifies the color of the field when displayed on a color device.

DATA =	
ALPHA	Specifies that the field will accept character data (default value)
NUMERIC	Specifies that the field will accept only numeric data

DATA specifies the data type for the field. Devices that support numeric shift will automatically shift to numeric mode when data is entered in this field if NUMERIC is specified.

DETECT =	
Y	Specifies that a light pen will cause a display device interrupt
N	Specifies that a light pen will not cause a display device interrupt (default value)

DETECT specifies whether a light pen can be used to cause a display device interrupt in this field. If DETECT = Y and a light pen is pointed at the field (or it can be cursor selected on some devices), the application is notified.

ENTER =	
Y	Specifies data is required in this field before the Enter or an action key is pressed
N	Specifies data is not required in this field (default value)

ENTER specifies whether the device enforces required input entry.

FILL =	
Y	Specifies an execution error if a user does not enter enough characters to fill the field
N	Specifies no error if less data is entered into a field than is required to fill the field (default value)

FILL specifies whether the device enforces filling an entire input field. This does not imply that input is required for a field. It means that if a user enters any characters in a field that has this attribute, the entire field must be filled.

HILITE =	
NOHILITE	Specifies that the field has no special highlighting (default value)
BLINK	Specifies that the field flashes on and off when a map displays
RVIDEO	Specifies that the field displays in reverse video—a dark or light background with contrasting light or dark text
USCORE	Specifies that the field is underlined

INTENSE =	
NORMAL	Specifies that text is displayed at normal light intensity (default value)
DARK	Specifies that text is not visible on a terminal display (passwords, for example)
BRIGHT	Specifies that text is displayed at higher or brighter light intensity than normal

INTENSE specifies the brightness of the value in the field when displayed on a screen.

MDT =	
Y	Specifies that the MDT is set on for a field
N	Specifies that the MDT is set off for a field (default value)

MDT (modified data tag) controls the default setting of the modified data tag indicator for the field when the field is displayed.

If a field has MDT set on, default data can be presented to the user during execution. When the map is displayed during execution, the default data for the field is displayed; the user can accept that data and press the Enter key or enter new data over the default data.

OUTLINE =

NOUTLINE	Draws no outline (default value)
BOX	Draws all four outline lines
ORIGHT	Draws a vertical outline on the right side of the field
OLEFT	Draws a vertical outline on the left side of the field
OOVER	Draws a line between the previous row and the current row of the field
OUNDER	Draws a line between the current row and the next row of the field

OUTLINE allows lines to be drawn at the edges of fields on the DBCS devices. When using this attribute, you must specify either NOUTLINE, BOX, or any combination of the remaining values. This is the only attribute that applies to both displays and printers.

PROTECT =

UNPROTECT	Specifies that data can be entered in a field
PROTECT	Specifies that data cannot be entered in a field
ASKIP	(Autoskip) Specifies that the field is automatically skipped

PROTECT specifies whether data can be entered in a field. A field with ASKIP specified is also protected.

Variable Field Definition

The VFIELD tag specifies information about a variable field on the map. The VFIELD, MAPEDITS, MESSAGES, and VATTR tags are a set of tags that describe a single variable field and must be repeated for each variable field on the map. The MAPEDITS, MESSAGES, and VATTR tags are always optional. The EVFIELD tag closes the definition and is required for each VFIELD.

Syntax	Attributes		
:map			
⋮			
:vfield	:VFIELD		
⋮	Attributes	Values	Uses
:mapedits	BYTES =	<i>field length in bytes</i>	Specifies the number of positions that the variable field occupies
⋮			
:messages	COLUMN =	<i>column number</i>	Specifies the column of the variable field marker
⋮	[DECIMALS =	<i>0[decimal places]</i>	Specifies the number of positions to the right of a decimal point in numeric items
:vattr			
⋮	[DESC =	<i>'field description'</i>	Describes what the field represents
:evfield	[EDITORDR =	<i>number</i>	Specifies the map edit sequence
⋮	[INDEX =	<i>index value</i>	Specifies the index value when the field is an array
:emap	[NAME =	<i>field name</i>	Identifies a map variable field
	ROW =	<i>row number</i>	Specifies the row of the variable field marker
	[TYPE =	CHA DBCS MIX NUM	Specifies the data type
		<i>[,initial value]</i>	Specifies the initial value of the field
	:EVFIELD[.]		

BYTES =
field length in bytes Specifies the number of positions that the variable field occupies

DECIMALS is the number of positions to the right of an implied decimal point in numeric items. Decimal places can be specified only for numeric data. The maximum number of decimal positions is 18 or the number of digits defined for the field, whichever is smaller.

BYTES does not include the field marker.

COLUMN =
column number Specifies the column of the variable field marker

DESC =
'field description' Describes what the variable field represents

DECIMALS =
0 Specifies 0 decimal places (default value)
decimal places Specifies a number of decimal places

DESC is a text description of what the variable field represents. The text can be up to 30 characters.

EDITORDR =

number Specifies the map edit sequence

EDITORDR allows you to modify the editing sequence in which variable fields on the map are checked on input to an application. If you specify EDITORDR for any variable named field on the map, you must specify it for all named fields, unless the field is in an array. EDITORDR can only be specified for the first element of an array. If not specified for any variable field, the default edit order is the order in which the variable fields appear on the map.

INDEX =

index value Specifies the index value when the field on the map is an array

INDEX must be a number between one and 9999. When the index value is 1, any of the attributes of the VFIELD tag can be specified. If the field is part of an array, then INDEX must be specified.

If INDEX is greater than 1, the following attributes can be specified:

- ROW
- COLUMN
- BYTES
- TYPE
- NAME
- INITIAL VALUE.

If INDEX is greater than 1, the following attributes cannot be specified:

- DECIMALS
- DESC
- EDITORDR.

If INDEX is greater than 1, the following tags cannot be specified:

- MAPEDITS
- MESSAGES.

If you do not explicitly reference the first element (INDEX = 1) in an array, then message EZE00782I is treated as an action message, and the map is not imported.

NAME =

field name Identifies a map variable field

NAME consists of 1 to 32 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national characters, hyphens, or underscores (A-Z, 0-9, \$, #, @, -, _).
- The name cannot contain blanks or have an EZE prefix, except EZEMSG.

The name can be a DBCS name up to 15 DBCS characters long with no embedded blanks.

ROW =

row number Specifies the row of the variable field marker

TYPE =

CHA	Specifies any character data
DBCS	Specifies double-byte character set, ideographic character data that requires two positions for each character
MIX	Specifies DBCS and single-byte data in the same field
NUM	Specifies numeric data

TYPE specifies the type of data that is acceptable for the variable field. Data input is checked to ensure that the type of data entered is valid for the field.

Initial value

initial value Specifies the initial value for the variable field

Map Edit Characteristic Definition

The MAPEDITS tag defines a list of map editing characteristics that can be specified for variable fields described with the VFIELD tag. This optional tag and its attributes further describes the MAPEDITS tag.

Syntax	Attributes		
:map	:MAPEDITS		
:vfield	Attributes	Values	Uses
:mapedits	[CURRSYMB =	Y N]	Specifies whether the currency symbol is supported
:messages	[DATEFORM =	<i>number</i>]	Provides date format editing
:vattr	[EDITRTN =	<i>edit routine</i>]	Specifies a routine or edit table for special data editing
:evfield	[FILLCHAR =	'N' <i>fill character</i>]	Specifies the character used to fill unused field positions
:emap	[FLDFOLD =	MAP Y N]	Specifies whether data entered into this field is folded
	[HEXEDIT =	Y N]	Specifies whether only hexadecimal digits can be entered in the input field
	[INPUTREQ =	Y N]	Specifies whether valid data must be entered
	[JUSTIFY =	LEF RIG N]	Specifies the position of data when it is shorter than the length of the field
	[MININPUT =	N <i>positions</i>]	Specifies the minimum number of required characters
	[NUMSEP =	Y N]	Specifies whether data can contain numeric separators
	[RANGE =	<i>low value high value</i>]	Specifies the range of valid numeric values
	[SIGN =	LEA TRA N]	Specifies a sign in a field as leading or trailing
	[ZEROEDIT =	Y N]	Specifies the format for numeric fields that have zero values
	[.]		

CURRSYMB =	
Y	Specifies that the currency symbol is supported
N	Specifies that the currency symbol is not supported (default value)

DATEFORM =	
<i>number</i>	Provides date format editing

CURRSYMB indicates whether the currency symbol is supported in the field. Y is valid only for numeric fields. When defining field length, remember that the currency symbol takes up one position.

DATEFORM indicates which date format is used when data is entered or displayed as follows.

Number	Format	Example
1	MM/DD/YY	06/30/89
2	MM-DD-YY	06-30-89
3	MM:DD:YY	06:30:89
4	YY/MM/DD	89/06/30
5	YY-MM-DD	89-06-30
6	YY:MM:DD	89:06:30
7	DD/MM/YY	30/06/89
8	DD-MM-YY	30-06-89
9	DD:MM:YY	30:06:89
10	YY-DDD	89-181
11	YY:DDD	89:181

Field edits should be defined as follows when you specify a date edit option:

- TYPE = NUM
- DECIMALS = 0
- CURRSYMB = N
- NUMSEP = N
- SIGN = N.

EDITRTN =	
<i>edit routine</i>	Specifies a routine or edit table for special editing of data in a variable field

EDITRTN indicates the name of a routine or edit table used for special editing of data that a user enters in a variable field. You can specify one of the following:

- The name of one of the following types of editing tables:
 - Match valid table
 - Match invalid table
 - Range match valid table.
- The name of a special function word subroutine:
 - Modulus 10 check digit routine (EZEC10)
 - Modulus 11 check digit routine (EZEC11).
- The name of a statement group used as an edit routine.

FILLCHAR =	
'N'	Specifies a null fill character
'fill character'	Specifies the character used to fill unused field positions

FILLCHAR indicates the character used to fill unused field positions on output to the terminal or printer. The default fill character for CHA, MIX, and DBCS fields is a blank. The default for HEX fields is 0. You must enclose all values for FILLCHAR in quotation marks.

FLDFOLD =	
MAP	Specifies folding on all character variable fields
Y	Specifies that data is folded
N	Specifies that data is not folded

FLDFOLD specifies whether lowercase alphabetic characters entered by the user are to be folded (converted) to uppercase. MAP means that all character variable fields and all single-byte data entered into mixed fields are folded when a user enters data into the fields. This field must be MAP if you specify VARFOLD = Y. If you specify VARFOLD = N, this value can be either Y or N. For more information, see VARFOLD = on page 9-5.

HEXEDIT =	
Y	Specifies that only hexadecimal digits can be entered
N	Specifies that the map field is not checked for hexadecimal characters (default value)

HEXEDIT specifies whether the input field is checked for hexadecimal digits. The data type of the variable field must be CHA for Y to be specified.

INPUTREQ =	
Y	Specifies that valid data must be entered in a map field
N	Specifies that input is not required in the field (default value)

INPUTREQ indicates whether valid data must be entered in a map field. When a field contains data other than blanks for character type or 0s for

numeric types, the field is considered to have input. Blanks in a character field and 0 in a numeric field do not satisfy the input required edit check. Even if blanks and 0s are valid values, INPUTREQ returns an error message unless the user types data into the field.

JUSTIFY =	
LEF	Specifies left justification (default for character data)
RIG	Specifies right justification (default for numeric data)
N	Specifies no justification

JUSTIFY specifies the position of data in a variable field when the data is shorter than the length of the field. If JUSTIFY is not specified, character data is left justified and numeric data is right justified. Right justification is required for numeric data with decimal positions specified. JUSTIFY=N is not valid for numeric fields.

MININPUT =	
N	Specifies no minimum number of characters required (default value)
<i>positions</i>	Specifies the minimum number of characters that must be entered in a valid variable field

MININPUT specifies the minimum number of characters that must be entered in a variable field if any data is entered.

NUMSEP =	
Y	Specifies that data can contain numeric separators
N	Specifies that data cannot contain numeric separators (default value)

NUMSEP specifies whether data in a field can contain the previously defined numeric separator. You can specify Y for numeric fields only. When defining field length, remember that each numeric separator takes up one position.

RANGE =	
<i>low value</i>	Specifies the smallest numeric value for a field
<i>high value</i>	Specifies the largest numeric value for a field

RANGE specifies the range of valid numeric values for a field. Low value is the smallest numeric value that can be entered in a specified field. High value is the largest numeric value that can be entered in a specified field. The high and low values must have the same length, number of decimal positions, and sign as defined for the field. Both low and high values must be specified if this attribute is used. The values are separated by a space and cannot be longer than the number of spaces in the field being defined. RANGE can be specified only for numeric fields.

SIGN =	
LEA	Specifies a leading sign (default value for numeric fields)
TRA	Specifies a trailing sign
N	Specifies no sign (default value for character fields)

SIGN specifies whether a sign is displayed in a field and whether it is a leading or a trailing sign. You can specify signs for numeric fields only. When defining field length, remember that the sign takes up one position.

ZEROEDIT =	
Y	Specifies the display format for numeric fields containing zero values
N	Specifies that no editing is done on zero numeric fields (default value)

ZEROEDIT specifies the display format for numeric fields that have 0 values.

The following table gives a list of what a numeric field will contain when ZEROEDIT is specified with either Y or N. The sample field has been defined as right justified with a length of 11. A "b" represents a blank fill character.

DEC (2)	CURRSYMB / NUMSEP	FILL	ZEROEDIT = N	ZEROEDIT = Y
N	N	N	nulls	0
N	N	b	blanks	bbbbbbbbb0
N	N	0	0000000000	0000000000
N	N	.	*****	*****0
Y	N	N	nulls	0.00
Y	N	b	blanks	bbbbbb0.00
Y	N	0	0000000000	00000000.00
Y	N	.	*****	*****0.00
Y	Y	N	nulls	\$0.00
Y	Y	b	blanks	bbbbbb\$0.00
Y	Y	0	0000000000	\$000,000.00
Y	Y	.	*****	*****\$0.00

Field Edit Message Definition

The MESSAGES tag specifies the user-defined messages for the field defined on the previous VFIELD tag that will be generated when the corresponding field edits are called as specified on the MAPEDITS tag. This optional tag and its attributes further describes the VFIELD tag.

The message number specified must correspond with the number in the user-defined message file.

Syntax	Attributes		
:map	:MESSAGES		
:	Attributes	Values	Uses
:vfield	[EDITMSG =	<i>message number</i>]	Specifies the number of the edit routine error message
:	[INVALMSG =	<i>message number</i>]	Specifies the number of the data type error message
:mapedits	[MININMSG =	<i>message number</i>]	Specifies the number of the minimum input error message
:	[RANGMSG =	<i>message number</i>]	Specifies the number of the value error message
:messages	[REQMSG =	<i>message number</i>]	Specifies the number of the input required error message
:	[.]		
:vatr			
:			
:vfield			
:			
:emap			

EDITMSG =

message number Specifies the number of the message displayed when the data fails a modulus check or table edit check

If you do not specify this attribute, the default error message for a modulus check is "Modulus check error—reenter," and the default error message for a table edit check is "Table edit validity error—reenter."

INVALMSG =

message number Specifies the message number of the message displayed when the data entered is incompatible with the variable field data type

If you do not specify this attribute, the default error message for invalid data type is "Data type error in input—reenter."

MININMSG =

message number Specifies the number of the message displayed when the minimum input edit check fails

If you do not specify this attribute, the default error message for this error is "Input minimum length error—reenter."

RANGMSG =

message number Specifies the number of the message displayed when the minimum or maximum value check fails

If you do not specify this attribute, the default error message is "Input not within defined range—reenter."

REQMSG =

message number Specifies the number of the message displayed when the input required edit check fails

If you do not specify this attribute, the default error message for this error is "No input received for required field—reenter."

Variable Field Attribute Definition

The VATTR tag defines attributes for individual variable fields described in the VFIELD tag. This tag is optional.

Syntax	Attributes																																				
:map : :vfield : :mapedits : :messages : :vattr : :evfield : :emap	<p>:VATTR</p> <table border="1"> <thead> <tr> <th>Attributes</th> <th>Values</th> <th>Uses</th> </tr> </thead> <tbody> <tr> <td>[COLOR =</td> <td>MONO BLUE RED PINK GREEN TURQUOISE WHITE YELLOW]</td> <td>Specifies the color of the field</td> </tr> <tr> <td>[CURSOR =</td> <td>Y N]</td> <td>Identifies if the cursor will be positioned on this field</td> </tr> <tr> <td>[DATA =</td> <td>ALPHA NUMERIC]</td> <td>Specifies the field data type attribute</td> </tr> <tr> <td>[DETECT =</td> <td>Y N]</td> <td>Specifies whether the field is light pen detectable</td> </tr> <tr> <td>[ENTER =</td> <td>Y N]</td> <td>Specifies whether data is required to be entered</td> </tr> <tr> <td>[FILL =</td> <td>Y N]</td> <td>Specifies whether an error occurs when a user does not enter enough characters to fill a field</td> </tr> <tr> <td>[HILITE =</td> <td>NOHILITE BLINK RVIDEO USCORE]</td> <td>Specifies highlighting characteristics of a variable field</td> </tr> <tr> <td>[INTENSE =</td> <td>NORMAL DARK BRIGHT]</td> <td>Specifies the light intensity</td> </tr> <tr> <td>[MDT =</td> <td>Y N]</td> <td>Controls the default setting of the modified data tag indicator</td> </tr> <tr> <td>[OUTLINE =</td> <td>{NOOUTLINE BOX [ORIGHT]{OLEFT}[OOVER][OUNDER]}}</td> <td>Specifies if lines are to be drawn at the edges of fields on DBCS devices</td> </tr> <tr> <td>[PROTECT =</td> <td>UNPROTECT PROTECT ASKIP]</td> <td>Specifies whether data can be entered into the field</td> </tr> </tbody> </table> <p>[.]</p>	Attributes	Values	Uses	[COLOR =	MONO BLUE RED PINK GREEN TURQUOISE WHITE YELLOW]	Specifies the color of the field	[CURSOR =	Y N]	Identifies if the cursor will be positioned on this field	[DATA =	ALPHA NUMERIC]	Specifies the field data type attribute	[DETECT =	Y N]	Specifies whether the field is light pen detectable	[ENTER =	Y N]	Specifies whether data is required to be entered	[FILL =	Y N]	Specifies whether an error occurs when a user does not enter enough characters to fill a field	[HILITE =	NOHILITE BLINK RVIDEO USCORE]	Specifies highlighting characteristics of a variable field	[INTENSE =	NORMAL DARK BRIGHT]	Specifies the light intensity	[MDT =	Y N]	Controls the default setting of the modified data tag indicator	[OUTLINE =	{ NOOUTLINE BOX [ORIGHT]{OLEFT}[OOVER][OUNDER]}}	Specifies if lines are to be drawn at the edges of fields on DBCS devices	[PROTECT =	UNPROTECT PROTECT ASKIP]	Specifies whether data can be entered into the field
Attributes	Values	Uses																																			
[COLOR =	MONO BLUE RED PINK GREEN TURQUOISE WHITE YELLOW]	Specifies the color of the field																																			
[CURSOR =	Y N]	Identifies if the cursor will be positioned on this field																																			
[DATA =	ALPHA NUMERIC]	Specifies the field data type attribute																																			
[DETECT =	Y N]	Specifies whether the field is light pen detectable																																			
[ENTER =	Y N]	Specifies whether data is required to be entered																																			
[FILL =	Y N]	Specifies whether an error occurs when a user does not enter enough characters to fill a field																																			
[HILITE =	NOHILITE BLINK RVIDEO USCORE]	Specifies highlighting characteristics of a variable field																																			
[INTENSE =	NORMAL DARK BRIGHT]	Specifies the light intensity																																			
[MDT =	Y N]	Controls the default setting of the modified data tag indicator																																			
[OUTLINE =	{ NOOUTLINE BOX [ORIGHT]{OLEFT}[OOVER][OUNDER]}}	Specifies if lines are to be drawn at the edges of fields on DBCS devices																																			
[PROTECT =	UNPROTECT PROTECT ASKIP]	Specifies whether data can be entered into the field																																			

COLOR =

MONO	Specifies monochrome (default value)
BLUE	Specifies blue
RED	Specifies red
PINK	Specifies pink
GREEN	Specifies green
TURQUOISE	Specifies turquoise
YELLOW	Specifies yellow
WHITE	Specifies white

COLOR specifies the color of the field when displayed on a color device.

CURSOR =

Y	Specifies the field where the cursor is positioned when the map is displayed
N	Specifies that the cursor will not be positioned in this field when the map is displayed (default value)

CURSOR specifies if the cursor will be positioned on this field when the map is displayed. The cursor attribute can be overridden by the application using the SET map item CURSOR processing statement. The default is the first unprotected variable field on the display.

DATA =

ALPHA	Specifies that the field will accept character data (default value)
NUMERIC	Specifies that the field will accept only numeric data

DATA specifies the data type for the field. Devices that support numeric shift will automatically shift to numeric mode when data is entered in this field if NUMERIC is specified.

DETECT =

Y	Specifies that a light pen will cause a display device interrupt
N	Specifies that a light pen will not cause a display device interrupt (default value)

DETECT specifies whether a light pen can be used to cause a display device interrupt in this field. If DETECT=Y and a light pen is pointed at the field (or it can be cursor selected on some devices), the application is notified.

ENTER =

Y	Specifies data is required in this field before the Enter or an action key is pressed
N	Specifies data is not required in this field (default value)

ENTER specifies whether the device enforces required input entry.

FILL =

Y	Specifies an execution error if a user does not enter enough characters to fill the field
N	Specifies no error if less data is entered into a field than is required to fill the field (default value)

FILL specifies whether the device enforces filling an entire input field. This specification does not imply that input is required for a field. It means that if a user enters any characters in a field that has this attribute, the entire field must be filled.

HILITE =

NOHILITE	Specifies that the field has no special highlighting (default value)
BLINK	Specifies that the field flashes on and off when a map displays
RVIDEO	Specifies that the field displays in reverse video—a dark or light background with contrasting light or dark text
USCORE	Specifies that the field is underlined

INTENSE =

NORMAL	Specifies that text is displayed at normal light intensity (default value)
DARK	Specifies that text is not visible on a terminal display (passwords, for example)
BRIGHT	Specifies that text is displayed at higher or brighter light intensity than normal

INTENSE specifies the brightness of the value in the field when displayed on a screen.

MDT =

Y	Specifies that the MDT is set on for a field
N	Specifies that the MDT is set off for a field (default value)

MDT (modified data tag) controls the default setting of the modified data indicator for the field when the field is displayed.

If a field has MDT set on, default data can be presented to the user during execution. When the map is displayed during execution, the definition data for the field is displayed; the user can accept that data and press Enter or enter new data over the default data.

OUTLINE =

NOOUTLINE	Draws no outline (default value)
BOX	Draws all four outline lines
ORIGHT	Draws a vertical outline on the right side of the field
OLEFT	Draws a vertical outline on the left side of the field
OOVER	Draws a line between the previous row and the current row of the field
OUNDER	Draws a line between the current row and the next row of the field

OUTLINE allows lines to be drawn at the edges of fields on the DBCS devices. When using this attribute, you must specify either NOOUTLINE, BOX, or any combination of the remaining values. This is the only attribute that applies to both displays and printers.

PROTECT =

UNPROTECT	Specifies that data can be entered in a field (default value)
PROTECT	Specifies that data cannot be entered in a field
ASKIP	(Autoskip) Specifies that the field is automatically skipped

PROTECT specifies whether data can be entered in a field. A field with ASKIP specified is also protected.

Chapter 10. Map Group Structures

The MAPG tag defines attributes for individual map group members. The AREA tag further defines aspects of a map group and can have attributes. A map group member defines the floating areas used by the maps in the map group. The EMAPG tag is required to close the definition.

Note: The MAPG tag is required only for maps with floating areas.

Syntax	Attributes															
:mapg	:MAPG <table border="1"> <thead> <tr> <th>Attributes</th> <th>Values</th> <th>Uses</th> </tr> </thead> <tbody> <tr> <td>{DATE=</td> <td>'<i>modification date</i>'</td> <td>Specifies the date the map group member was last modified</td> </tr> <tr> <td>GRPNAME=</td> <td><i>name</i></td> <td>Identifies the map group</td> </tr> <tr> <td>{TIME=</td> <td>'<i>modification time</i>'</td> <td>Specifies the time the map group member was last modified</td> </tr> <tr> <td>[.]</td> <td></td> <td></td> </tr> </tbody> </table>	Attributes	Values	Uses	{DATE=	' <i>modification date</i> '	Specifies the date the map group member was last modified	GRPNAME=	<i>name</i>	Identifies the map group	{TIME=	' <i>modification time</i> '	Specifies the time the map group member was last modified	[.]		
Attributes		Values	Uses													
{DATE=		' <i>modification date</i> '	Specifies the date the map group member was last modified													
GRPNAME=		<i>name</i>	Identifies the map group													
{TIME=		' <i>modification time</i> '	Specifies the time the map group member was last modified													
[.]																
:																
:area																
:																
:emapg																
	:EMAPG[.]															

DATE =

'*modification date*' Specifies the date the map group member was last modified

TIME =

'*modification time*' Specifies the time the map group member was last modified

DATE format is mm/dd/yy. If the external source format file was produced by CSP/AD export, this date shows when the map group member was last modified in the MSL. The date is ignored on import.

TIME format is hh:mm:ss. If the external source format file was produced by CSP/AD export, this time shows when the map group member was last modified in the MSL. The time is ignored on import.

GRPNAME =

name Identifies a group of maps used with an application

GRPNAME identifies a group of maps used with one application. It consists of 1 to 6 characters and must meet the following conventions:

- The first character must be alphabetic or national (A-Z, \$, #, @).
- The remaining characters must be alphanumeric or national (A-Z, 0-9, \$, #, @).
- The name cannot contain blanks or have an EZE prefix.

Floating Area Definition

The AREA tag defines a floating area. This required tag further describes the MAPG and EMAPG tag set. If multiple areas are specified, they must have unique devices specified.

Syntax	Attributes	Values	Uses
:mapg : :area : :emapg	:AREA Attributes DEVICE = [SIZE = [STARTPOS = [.]	<i>device name</i> <i>depth width</i> 1 1 <i>line column</i>	 Specifies the device associated with the floating area Specifies the size of the floating area Specifies the starting position of the floating area

DEVICE =	
<i>device name</i>	Specifies the device associated with the floating area

SIZE =	
<i>depth</i>	Specifies the depth of the floating area in number of lines
<i>width</i>	Specifies the width of the floating area in number of columns

You can specify each device only once in the map group.

The following table gives the valid devices and their sizes.

Supported IBM Devices	Rows	Columns
3643-2	6	40
3277-1	12	40
3643-4	16	64
3278-1 3278-1B 8775-1C ANY-1D	12	80
3278-2 3278-2B 8775-2C ANY-2D	24	80
5550D (DBCS display device)	24	80
3278-3 3278-3B 8775-3C ANY-3D	32	80
3278-4 3278-4B 8775-4C ANY-4D	43	80
3278-5 3278-5B ANY-5D	27	132
ANY-D (3290 Configured as 62x160)	255	160
3767 PRINT-B PRINTER (physical size 86x132)	255	132
5550P (DBCS printer with size 66x158)	255	158

SIZE is the size of the floating area. Specify size in terms of depth (number of lines) and width (number of columns).

STARTPOS =	
1	Specifies line one (default value)
1	Specifies column one (default value)
<i>line</i>	Specifies the line coordinate where the floating area starts
<i>column</i>	Specifies the column coordinate where the floating area starts

STARTPOS is the starting position for the floating area. Specify the starting position in terms of the beginning line and column. Separate the line and column values by a space.

Map Group Structure Example

```
:mapg      grpname = xxxxxx
           date    = 'mm/dd/yy' time    = 'hh:mm:ss'.
:area      size    = xxx xxx  startpos = xxx xxx
           device  = xxxxxxxx.
:emapg.
```

Note: This example illustrates the syntax of all map definition tags and attributes. Actual definitions do not require or use all tags and attributes shown here.



Appendix A. External Source Format Functions with CSP/AD and CSP/AE Commands

This appendix describes the CSP/AD and CSP/AE commands that control external source format functions.

Import and Export of Member Specification Library Members

External source format files are application load file (ALF) members or serial files. In a serial file, the data is written as 80-byte records with sequence numbers in columns 73-80. Column 72 is reserved for a continuation character that is used by some member types. On import, the external source format provides defaults for tags and attributes that are optional and not specified. Export produces all tags and attributes with two exceptions. The export function does not produce the syntax in the following cases:

- If the map and map edit attributes are default values
- If the SQL clauses are unmodified default values.

After using external source format to define your applications and members, use the following methods to import and export them using either a member in an ALF or a serial file.

As a member in an ALF:

- To move the definitions to the ALF, use the ALFUTIL GET command. GET does not check for errors.
- After moving the definitions to the ALF, use the CSP/AD IMPORT command to move them to the MSL. IMPORT does check for errors.
- To move MSL members back into the ALF, use the CSP/AD EXPORT command. EXPORT creates all of the external source format syntax, translating the definitions back into a readable format.
- To move ALF members back to serial files, use the ALFUTIL PUT command. PUT does not create external source format syntax; this command only moves the member definitions between files.

As a serial file:

- The IMPORT and EXPORT commands can translate definitions directly, without writing to the ALF. In other words, you can IMPORT from a serial file to an MSL and EXPORT from an MSL to a serial file.

Note: Do not export Interactive Map Definition (IMD) maps because some of the IMD function will be lost. For example, IMD allows a map to be displayed differently on other devices. CSP, however, requires the same presentation for all devices. In this case, the multiple presentations are lost. The lost functions are the same functions that are lost if an IMD map is edited using CSP/AD map definition. If you want to export an IMD map in external source format, you must first edit the map using CSP/AD map definition.

Files Used For Importing and Exporting

The following files are used during IMPORT and EXPORT:

- External source format file
- Internal format file
- Message print file
- Batch log file
- Import work file
- CSP/AD batch command file.

External Source Format File

The external source format file is either a member in a variable-length ALF or a serial file with 80-byte fixed-length records. When the external source format file is PUT from the ALF into a file by ALFUTIL, it is placed in a fixed-length serial file with 80-byte records. Within each record, the first 72 columns contain MSL definitions in the external source format. Columns 73-80 contain a sequence number that is produced during export and ignored on import. The format of definitions within the external source format file varies depending on the MSL member type.

CSP/AE host environments facilitate the movement of external source format files between the ALF and a serial file using the ALFUTIL GET and PUT functions.

ALFUTIL references the serial file name using the name ALFB (for CICS) or ALFA (for all other environments), which are defined as fixed length files with logical record lengths of 80 bytes.

You do not have to specify the file format when you request ALFUTIL to PUT a member to a serial file. ALFUTIL determines what format the member is in prior to copying it to ALFB (for CICS) and ALFA (for all other environments). When you request ALFUTIL to GET a copy of a serial file ALFA, you must specify the file format. With the FORMAT keyword on the GET command, you can specify either external source format or another type of format. The default is non-external source format. In online mode, specify the input file format on the ALF02 panel.

Header Record: Table A-1 on page A-3 shows the format of the first 50 bytes of the first record of an external source format file or an internal format file. The internal format file produced by CSP/AD Version 3 Release 2 Modification 2 and earlier versions do not have the header. For any external source format files that do not have the header, the system returns message EZE00479A, "Invalid data in input file."

CSP/AD produces a new header on the first export of a job and each time the system resource name associated with the external source format file changes. Each time CSP/AD produces a new header, it resets the sequence number to 100. A single file might contain multiple headers if the system that CSP/AD is operating on supports appending to the end of a file.

Table A-1. File Format

Field Description	Data Type	Length (Bytes)
Identifier ':EZEI'-Internal format files ':EZEE'-External source format files	Character	6
CSP/AD version number 322 330	Character	3
Reserved	Character	1
Reserved for enabled products	Character	12
Reserved	Character	1
File creation date - mm/dd/yy	Character	8
Reserved	Character	1
File creation time - hh:mm:ss	Character	8
Reserved	Character	10

Internal Format File

The internal format file is either a member in a variable length ALF or a serial file with 256-byte fixed length records. If it is an ALF member or a serial file (created using the ALFUTIL PUT function) from a previous release of CSP/AD, it does not contain a header record.

The internal format is still useful for the following:

- Upward compatibility
- Compatibility with IMD and Screen Definition Facility-II (SDF-II)
- Quick movement of members between MSLs
- Efficient archiving of MSLs.

Message Print File

Messages detected during validation go to the current print file for the CSP/AD session. The format of the print file remains unchanged.

Batch Log File

This file contains the final status of the CSP/AD batch command. It is fixed-length, 121 bytes with the first character of each record being a carriage control character.

Import Work File

If BYPASS = N is specified on import or if a selective import is requested, a work file is required to import an external source format file. A work file is a fixed-length serial file with a record length of 256 bytes. The import function automatically defines the work file, DCAIMPW, in the DCAWORK area. If you use the BYPASS = N option, you might need to increase the index space in DCAWORK because of the size of the work file.

CSP/AD Batch Command Files

The CSP/AD batch command file can contain the IMPORT and EXPORT batch commands.

For information on the import and export commands contained in the file, refer to the *CSP/AD Developing Applications* manual and the Cross System Product system administration manual for your operating environment.

Error Processing

The IMPORT command validates the external source format members individually to prevent invalid members from getting in the MSL. However, members in the same IMPORT file are not validated against each other. When multiple members in the external source format file have the same name, only one of them remains in the MSL. Which member remains depends on the options you specify. If you specify "replace duplicate members" as NO, the first valid member in the external source format file is imported provided no members by that name already exist in the MSL before IMPORT begins. If you specify "replace duplicate members" as YES, the last valid member in the file imported by that name is the one that remains in the MSL.

When invalid members are in the import file, you receive an error message. Error messages are written to the current print destination.

IMPORT does not compare definitions to be imported with members that exist in the MSL. This can cause errors. For example, if the member you import is already in the MSL as an EXECUTE or file input/output process, and you specify the REPLACE option to replace the imported member with a CONVERSE process, then any batch application in the MSL that uses the process is now in error.

Conditions detected by the test facility or by the preprocessor are not checked for on import and do not prevent the IMPORT. For example, IMPORT does not detect the following:

- Statements that refer to members of the wrong member types
- Process objects that do not match the process option
- Map edit routines that cause screen input/output.

Conditions detected by application, data, or map definition are verified on import and do prevent the import. For a complete list of the detected error conditions, refer to the *CSP/AD and CSP/AE Messages, Codes, and Problem Determination* manual. Some examples are as follows:

- Invalid syntax in process statements
- Invalid level numbers in record structures
- Hex edit only valid for character fields on a map
- Maximum number of occurs exceeded.

Prolog lines that are too long are an example of a condition that is corrected on IMPORT and does not prevent IMPORT. They are split into multiple lines if they are too long.

CSP/AE Batch Commands

Use the batch ALFUTIL GET and PUT functions to migrate definitions that have been exported to an ALF from one development system to another. The batch ALFUTIL PUT function copies an external source format export file within an ALF to a serial file, and the ALFUTIL GET function copies an external source format serial file to an ALF.

For information on PUT and GET, refer to the Cross System Product system administration manual for your operating environment.



Appendix B. DBCS Support

If you specify the DBCS startup parameter, tags and attributes are produced in uppercase. Otherwise, EXPORT produces tags and attributes in lowercase. Tags and attributes are accepted in uppercase and lowercase during import.

Whenever DBCS data is produced in the external source format file, CSP/AD inserts the necessary shift-out (SO) and shift-in (SI) characters into the data stream so that the file can be displayed on a DBCS device. On import, DBCS and mixed data are validated and must be legal DBCS or mixed strings. For DBCS constant and variable fields and for table contents (but not for literals in statements), the SO and SI characters are stripped from the data before it is placed in the MSL. For mixed constant and variable fields, for table contents, and for comments (but not for literals in statements), contiguous SO and SI characters that appear at the end of one line and at the beginning of the next are stripped from the data before it is placed in the MSL. This allows extra SO and SI characters to be added to the data in the external source format file when data must be split across multiple lines.

When splitting data to fill column 71 of the external source format file causes a DBCS character to be split in half, place an SO character in column 71, a continuation character in column 72, and the rest of the data on the next line (with an SI character in column 1). See **1** in Figure B-1.

When column 71 is filled with the second half of a DBCS character, put an SO character in column 72 (it has the double duty of being an SO character and a continuation character) and put the rest of the data on the next line (with an SI character in column 1). See **2** in Figure B-1.

When you want a contiguous SO and SI character set in the data and the data must be split between the SO and the SI characters, put an extra pair of SI and SO characters in the external source format file. See **3** in Figure B-1.

External Source Format file	Data as stored on the MSL
1 6 7 8	
1234567890 ... 012345678901234567890	
1 za<DiDjDkDl>x00000300	za<DiDjDkDlDmDnDo>ab
<DmDnDo>ab 00000400	
2 a<DiDjDkDlDm>00000100	a<DiDjDkDlDmDnDo>abc
<DnDo>abc 00000200	
3 za<DiDjDkDl>x00000500	za<DiDjDkDl><DmDnDo>
<><DmDnDo> 00000600	

Figure B-1. External Source Format Data Storage



Appendix C. Report Formats

The information in this appendix provides no programming interfaces for customers. For detailed information about programming interface information, see the statement about programming interfaces in the Preface of this manual.

In batch mode, member messages are sent to the current print destination. For a description of these messages, refer to the *CSP/AD and CSP/AE Messages, Codes, and Problem Determination* manual.

All messages for a member are grouped together below the beginning member tag to keep individual definitions separate. Figure C-1 on page C-2 shows a sample format of messages sent to the current print destination.

The following information appears in the printed message file:

- The tag that delimits the member definition
- The attributes identifying the name of the member being defined
- Three groups of messages for import:
 - Messages dealing with a specific tag or attribute. The following information precedes these messages: the line number, the input line, and the sequence number of the line from the input file that triggered the message.

When the input file is an ALF member, the line number is the logical record number. Three logical 80-byte records are grouped together to form one physical record in the ALF member.

- Messages that deal with a specific tag that is a repeatable tag in the definition. The line number of the tag prints preceding these messages.
- Messages that are not related to a single attribute but to the relationships between attributes.

If the name of the member is missing or not recognized, question marks appear in place of the name. If a member definition is completed and the tag following it does not correctly begin a new member recognized by CSP/AD, question marks replace the tag beginning the member as well as the name of the member.

External Source Format Import Messages

:MAP GRPNAME=MAPG MAPNAME=MAP1

Line related messages

00004 helpkey=2 bypkey=1 3 5 7 9 11 ... 00000400
EZE00827A HELPKEY specified without HELPMAP

Tag related messages

00025
EZE00770A Field text or initial value is too large for field length

Member related messages

EZE00766A Conflicting field markers
EZE00767A Fields at 003,008 and 003,011 overlap

:TBLE NAME=TABLE1

Line related messages

00346 name=field1 type=PACK bytes=4 decimals=11 ... 00035700
EZE00870A DECIMALS value is invalid or exceeds specified length

Figure C-1. Sample Printed Messages

Glossary

This glossary defines terms and abbreviations used in the CSP/AD, CSP/AE, and CSP/370 Runtime Services libraries. If you cannot find the term that you are looking for in this glossary, refer to the index or to the *Dictionary of Computing*, SC20-1699.

A

access module. An SQL term only, an access module is the machine code for the SQL commands in an application and is stored in the SQL/DS database. CSP/AD builds the access module for CSP/AD applications directly using SQL/DS extended dynamic statements.

ADD option. A CSP/AD process option that places a new record in a file or database.

AID byte. An attention identifier (AID) that shows that an operator action initiated a transmission. It tells whether a function key (PF key), a PA key, or the Enter key was pressed.

AIM. See *application image module*.

ALF. See *application load file*.

ALFUTIL. See *ALF utility*.

ALF utility (ALFUTIL). A utility that maintains and transports the contents of an application load file (ALF).

allocation. The assignment of facilities of a computer system for the accomplishment of specific jobs.

alphabetic. Related to any one of the letters A through Z (uppercase and lowercase). Some CSP/AD elements allow definitions that include national characters (#, \$, @) as alphabetic characters, as well as the hyphen and the underscore.

alphanumeric. Related to a character set that contains letters (A-Z), digits (0-9), and usually the national characters (#, \$, @), as well as the hyphen and the underscore.

alternate PCB. In IMS/VS, an alternate program communication block (PCB) is also known as a teleprocessing (TP) PCB. It represents the message queue for a logical terminal or an alternate transaction. The logical terminal can be different from the terminal from which the input message came. An alternate PCB sends messages to its destination at the next commit point. See also *express PCB*; compare with *I/O PCB*.

alternate specification. A parameter that uses the name of a defined record structure for a new file

specification with a unique file name. An alternate specification can be used to simplify the definition of an alternate index, alternate organization, or alternate location.

application. A set of definitions and processes that, when generated using CSP/AD and run using CSP/AE or CSP/370 Runtime Services, performs a systematic sequence of operations to produce a specific result.

application control statement. In CSP/AD, a statement that affects the logical execution of an application.

application definition. One of the facilities used to define a Cross System Product that helps create an application logic and structure.

application image module (AIM). The first module to run in a Cross Systems Product application. It contains the application specifications and has the same name as the application.

application load file (ALF). A data set that contains CSP/AD output data sets (generation, export, trace, and others). The generation output data sets include application load modules that CSP/AE can run interpretatively.

application logic. Part of the application structure defined by processes and statement groups.

application name. A unique identification of an application that must be 1 to 7 characters. The first character must be alphabetic (A-Z) and the remaining characters must be alphanumeric (A-Z, 0-9). No national or special characters, embedded blanks, or EZE prefixes are allowed.

application plan. A DB2 term only, an application plan is the control structure produced during the DB2 bind process and used to process SQL statements encountered while the application runs.

application process list. A list or sequence of main processes. The list contains the specifications for each main process.

application processing table (APT). A table that contains an entry for each process, statement group, and the special function words EZECLOS, EZERTN, and EZEFL0.

application segment. A self-contained portion of an application for which all internal storage resources can be released after running the application. CSP/AD main transactions can be run in segmented mode on DPPX, CICS/VS, IMS/VS, and VSE systems. Storage resources are released each time a map is conversed

and the application is waiting for a response from the user.

application specification. The part of the application definition that defines the general information about the application you are creating, such as whether it is a transaction or batch application, and which maps it will use.

application structure. A top-down structure of processing blocks that defines the actual logic of the application. The logic is defined by processes and statement groups. The following make up the application structure: lists of processing blocks, nonprocedural options for each processing block, procedural processing statements coded within processing blocks or as application flow logic, and database calls that carry out the process option when the object is a segment in a DL/I database or a row in a relational database.

APT. See *application processing table*.

arithmetic statement. A statement that contains arithmetic operations and operands. The statement results in a single numeric value when it runs.

associated members. Members that share a commonality with their hierarchical counterparts. For a record or table, associated members are all global data items. For a map or map group, the associated members are the members with the same group name. For PSB, the associated members are the segment records in the PSB, the global items in those records, and any items named as secondary index fields in the PSB. For an application, the associated members are the processes, statement groups, maps, PSBs, tables, records, and global data items referenced directly or indirectly within the application. Processes, data items, and statement groups have no associated members.

authorization ID. A name identifying a DB2 or SQL/DS user or group of users.

B

batch message processing (BMP). A form of IMS/VS processing in which a program is started as a batch job, but runs in a region controlled by IMS/DC, allowing the capabilities of an online region. It can access IMS/VS message queues for input and output, as well as DL/I, fast path, DB2 databases, and operating system files. See also *batch-oriented BMP* and *transaction-oriented BMP*.

batch mode. A mode in which CSP/AD accepts a request to read a command file, and then runs the commands in the file without requiring user interaction.

Batch Terminal Simulator (BTS). An IBM product that allows IMS/VS DB/DC programs to run in a TSO or batch environment. BTS provides a comprehensive means of checking application logic, IMS/VS application interfaces, teleprocessing activity, 3270 format control blocks, and database activity.

batch-oriented BMP. A batch message processing (BMP) program that does not access the message queue for input. Databases and operating system files are available for input and output processing, and message queues are available for output.

batch transaction. A type of application that runs on a system without user interaction.

BIN. See *binary data*.

binary data (BIN). A type of data consisting of numeric values stored in bit patterns of zeroes and ones. Binary data allows a large number to be placed in a smaller space of storage. See also *data type*.

bind. A DB2 term only, a bind is the process by which the output from the DB2 precompiler is converted to a usable control structure called an application plan. During this process, access paths to the data are selected and some authorization checking is performed.

blink. A map field characteristic that causes the contents of a field to be flashed on and off when data is displayed in the field.

BMP. See *batch message processing*.

bright. A map field characteristic that causes the contents of a field to be displayed with brighter than normal intensity on a screen.

BTS. See *Batch Terminal Simulator*.

C

CALL statement. An external branching statement that automatically returns control to the original application after the called application, subroutine, or high-level language program has run.

called application. An application that is started by a link from another application. It might have addressability to storage in the calling application. A CSP/AD CALL statement provides the link interface to a called application.

called batch. A type of batch application that performs a function and then returns control to the calling application.

called parameter list. The list of maps, records, and data items passed to a called application. The list is defined during application definition.

called transaction. A type of called application that is started by a link in another transaction application. It can contain statements that cause terminal interaction.

CHA. See *character data*.

character data (CHA). The default data type. CHA data consists of the input from the displayable character set entered from any terminal keyboard. See also *data type*.

child segment. A segment in a DL/I database that is not at the highest level in the database hierarchy. It is called the child of the segment to which it is related at the next higher level in the hierarchy. See also *hierarchy*, *parent segment*, and *segment*.

CICS. Customer Information Control System.

CICS/VS conversational mode. A mode of operation of a computer system in which a sequence of alternating entries and responses between a user and the system takes place like a dialog between two persons. During this dialog, the application holds the locks and the storage resources until the next user response. Contrast with *segmented mode* and *pseudoconversational*.

CL. See *Control Language*.

CLOSE option. A process option that (1) closes the file associated with a serial file, relative file, indexed file, or a print map, and (2) releases a set of rows selected from a relational database for scanning or updating.

codes. CSP/AD supplies formatting code symbols used to design the form of a map with CSP/AD. The codes and default symbols are constant (#), variable (¬), spacer (/), mixed constant (%), mixed variable (|), DBCS constant (+), and DBCS variable (@).

color. A map field characteristic that allows a color to be specified for a field. The available colors on a color display terminal are mono (standard screen color), blue, red, pink, green, turquoise, yellow, and white.

column. The vertical component of a table. A column has a name and a particular data type. All values in a particular column are alike in that they are the same type.

command codes. DL/I control codes that request special processing for segments. Command codes are passed as part of the segment search arguments (SSAs) in a DL/I called parameter list.

commit. The process of making changes permanent to a recoverable file or database. When data has been

committed, a new logical unit of work (LUW) is established. CSP/AD applications commit changes by calling the EZECOMIT service.

common programming interface (CPI). A definition that provides languages, commands, and calls that programmers can use to develop applications that take advantage of the consistency offered by Systems Application Architecture (SAA).

comparison value item. The name of a data item specified in a qualification statement found in a segment search argument (SSA) of a DL/I call definition. At run time, the current value of the item is used as the comparison value in the DL/I SSA built by the execution facility. DL/I uses the value in determining which segments it returns in response to the DL/I call.

compatible. The capability of an application to run the same on different operating systems.

conditional statement. A statement that permits running a choice of several possible operations. A conditional statement is usually dependent on the result of some prior operation or command. Conditional processing statements include IF, AND, OR, WHILE, ELSE, FIND, and TEST.

constant code (#). The default code symbol that begins or ends a constant field or ends a variable field on a map. The default code symbol can be changed.

constant field. A fixed-length field that cannot be used for inputting data or displaying data; that is, it cannot be changed by the user or accessed by the application.

control function. An action that affects the recording, processing, transmission, or interpretation of data; for example, starting or stopping a process, carriage return, font change, rewind, and end of transmission.

Control Language (CL). The set of all commands with which a user requests AS/400 functions.

conventional files. Non-database files.

conversational MPP. A type of message processing program (MPP) characterized by the ability to save data during end user think time, even though locks on the database are lost. IMS/VS conversational is similar to CICS/VS pseudoconversational. IMS/VS has no capability that is similar to CICS/VS conversational.

CONVERSE option. A process option that displays a map on a screen and accepts a user response of data or a terminal key.

COPYLIST. A command that copies all members of the current list from the read-only member specification library (MSL) to the read/write MSL.

copy utility. (1) A utility that reads data from a source, leaving the source data unchanged, and writes the same data elsewhere in a physical form that might differ from the source. (2) A utility that causes a member in the member specification library (MSL) to be copied into a new member of an MSL.

CPI. See *common programming interface*.

creator ID. The authorization ID of a DB2 or SQL/DS user that created a table in a relational database.

Cross System Products. See *Cross System Product/Application Development*, *Cross System Product/Application Execution*, and *CSP/370 Runtime Services*.

Cross System Product/Application Development (CSP/AD). An online tool allowing users to define, test, and generate applications.

Cross System Product/Application Execution (CSP/AE). A product that provides the means whereby user applications, defined and generated using CSP/AD, can run on one or several computer environments.

CSP/370 Runtime Services. The Cross System Product facility that supports COBOL generated programs and the execution of CSP/AD applications in the IMS environment.

CSP/AD processing statement. See *processing statements*.

CSP/AD special function word. See *special function word*.

cursor. (1) When used with a terminal, a cursor is the movable indicator on a panel, usually indicating where the next character is to be typed, replaced, or deleted. (2) When used with a relational database, a cursor is a named control structure used by an application to point to a row of interest. The position of the row is within some selected set of rows from the set. The cursor is managed for you in CSP/AD applications.

cursor field. The field that identifies the name of the data item on a map where the cursor will be positioned when the map is displayed.

cursor stability. One of the two possible settings for isolation level. A logical unit of work (LUW) is protected from accessing uncommitted updates of another LUW. It is not protected from updating uncommitted reads of another LUW (except for the data that the cursor is currently accessing). A user might not get the same values if selecting the same data repeatedly. (Cursor here refers to the pointer marking the location of data in the database and not the terminal cursor.)

D

dark. A map field characteristic that causes the contents of a field to be invisible on a map.

database. (1) DL/I database: A set of related data structured as a hierarchy of segment types and accessed using Data Language/I. (2) SQL relational database: A set of structured data perceived by its users as a collection of tables and accessed using the Structured Query Language. (3) DTMS database: A collection of data managed by the Database and Transaction Management System under DPPX. DTMS databases are treated as files by CSP/AD.

Database and Transaction Management System (DTMS). An associated DPPX licensed program that manages access to databases and processes transactions.

database definition. A formal definition of a database to DL/I. DL/I uses the information in the database description and CSP/AD uses the application view of the database defined in the program specification block (PSB).

database description (DBD). The collection of macro parameter statements that describes a DL/I, fast path, or GSAM database. The segment content and hierarchy is specified in the DBD, as well as the physical characteristics, such as organization and access method.

database identifier. The parameter in the parameter list of the DL/I call that points to the program communication block (PCB) identifying the database that DL/I is to access on the call.

database manager. A program that manages databases. IBM database managers are IMS/VS and DL/I DOS/VS for DL/I databases, DB2 and SQL/DS for relational databases, and DTMS for DTMS databases on the DPPX operating system.

database PCB. A program communication block (PCB) representing a database that an application can access. In addition, the database PCB specifies the data, its segment or field level, and the processes that are valid.

database request module (DBRM). A DB2 component created by the DB2 precompiler that contains information about SQL statements. DBRMs are input during the bind process.

data definition. (1) In CSP/AD, a facility where a user describes the features of, specifies the relationship of, or establishes the context of data. (2) Information that describes the contents and characteristics of a field, record, or file.

data description specification (DDS). A description of the user's database or device file that is entered into the system using a fixed-form syntax and stored in a DDS source member object. This source member can then be used as input to system utilities that create the actual system objects related to the described file.

Data Entry Data Base (DEDDB). A type of IMS fast path database that is designed to contain large volumes of data with a high rate of availability. Long chains of segment occurrences are managed using subset pointers. One segment type is stored near the root segment and the occurrences are kept in chronological order.

data item. A unit of information defined by length, data type, and other characteristics.

data item name. A unique identification within a data item list definition that can be 1 to 32 characters. The first character must be alphabetic or national (A-Z, \$, #, @), the remaining characters must be alphanumeric, national (A-Z, 0-9, \$, #, @), hyphen, or underscore. A data item name can be a DBCS name. No special characters, embedded blanks, or EZE prefixes are allowed. See also *DBCS name*.

Data Language/I (DL/I). A data management control system that lets programmers create, access, and maintain shared databases. The DL/I DOS/VS and IMS/ESA data manager (DM) products provide DL/I services on VSE systems. The IMS/VS database (DB) product provides DL/I services on MVS systems.

data load module. A module that contains the data structure and data item location tables and are then copied into read/write storage while an application runs.

data manipulation statement. A statement that is used to assign a value to a data item, to look for and retrieve a value from a data table, or to set a condition or a field on a map.

data set. The major unit of data storage and retrieval. A data set consists of a collection of data in one of several prescribed arrangements, described by control information to which the system has access.

data structure. The organization of a collection of data items. In CSP/AD data can be structured as a record, table, or working storage.

data type. Specifies the internal format or type of data. Data type determines how an item is processed when referenced in processing statements.

DB2. See *IBM Database 2*.

DB PCB. See *database PCB*.

DBCS. See *double-byte character set*.

DBCS constant code (+). The default code symbol that specifies the starting position of a double-byte character constant field made up entirely of characters of a double-byte character set. The default code symbol can be changed.

DBCS literal. Literal data enclosed in quotes, single or double, and shift-out (SO) and shift-in (SI) characters. The first quote is prefaced with a 'G'.

DBCS name. A name consisting of double-byte characters only. It must begin with a SO (shift-out) delimiter and end with a SI (shift-in) delimiter; and it must contain at least one non-42nd ward DBCS character. It cannot contain DBCS blanks.

DBCS variable code (@). The default code symbol that specifies the starting position of a double-byte character variable field made up entirely of characters from a double-byte character set. The default code symbol can be changed.

DBD. See *database description*.

DBD name. In DL/I, the name of a control block called the database description, which describes the structure of a database. One DBD is defined for each database.

DBRM. See *database request module*.

DCT. See *destination control table*.

DDS. See *data description specification*.

DEDDB. See *Data Entry Data Base*.

default definition. A definition that is implicit or preset.

deferred program switch. A type of program switch in which the first program responds to the terminal informing IMS to start another transaction that is associated with a second program on the next input from the terminal. For a conversational message processing program (MPP), the program switch is achieved by modifying the scratchpad area (SPA) to specify the new transaction name before sending it back to IMS (via the I/O PCB). For a nonconversational MPP, the program switch is achieved by including the next transaction name on the map in such a way that it becomes the first 8 bytes of the input message.

definition facility. The CSP/AD facility that prompts you to define and describe a record, table, data item, map, application, process, statement, or program specification block (PSB). These definitions are stored in the member specification library (MSL).

DELETE option. A process option that removes a record from a file or database. An UPDATE or SCAN

for the update option must be run before a DELETE option.

delete utility. A utility that causes a member in the member specification library (MSL) to be removed.

destination control table (DCT). In CICS/VS, a table describing each of the transient data definitions used in the system or in connected CICS/VS systems.

detect. A map field characteristic that allows selection by a light pen on a display.

device name. A unique identification of a CSP/AD supported device that can be used to display maps. The device name can be 6 to 7 characters.

directory print utility. A utility that produces a list of members in the member specification library (MSL).

DISPLAY option. A process option that either sends a map to a screen for viewing only (no input from user) or to a printer.

Distributed Processing Programming Executive

(DPPX). A comprehensive collection of licensed programs that make up the operating system for 8100 Information System hardware.

DL/I. See *Data Language/I*.

DL/I call. An invocation of DL/I by an application. DL/I is passed a program communications block (PCB) address, the function to be performed, the address of an I/O area, and the segment search argument (SSA) that describe the segments that are to be returned.

DL/I database. A collection of data organized in hierarchical segments to establish data relationships and dependencies and reduce data redundancy.

DL/I segment. The primary unit of data in a DL/I database. The segment is a single contiguous block of data like a CSP/AD record. The segment is divided into data fields that are like the data items in a record. Segments are defined in the member specification library (MSL) as records with DL/I segment organization.

double-byte character set (DBCS). A character set of single characters that require 2 bytes of data for each character in a record, table, or map. This data requires a terminal or printer that supports DBCS. This character set is used to represent Chinese, Japanese, and Korean languages.

DPPX. See *Distributed Processing Programming Executive*.

DTB. See *dynamic transaction back out*.

DTMS. See *Database and Transaction Management System*.

duplicate MSL member. A member in a member specification library (MSL) that exists in more than one of the MSLs currently accessed. One may be a copy of the other, or they might be totally different definitions.

DXFR statement. A transfer statement that starts running another CSP/AD application, non-CSP/AD program, or transaction. DXFR might run differently and might have different restrictions depending on the execution environment. The current application ends when the DXFR statement is linked.

dynamic mode. A mode of running SQL applications in which each SQL statement in the application is prepared (precompiled) just before it runs.

dynamic transaction back out (DTB). CICS error recovery.

E

edit routine. A routine specified for special editing of data that is typed by a user in a map field.

edit table. A table that is named in place of an edit routine in the definition of a map variable field edit.

element, nonprocedural. See *nonprocedural element*.

element, procedural. See *procedural element*.

error code. A code that contains information about an exception condition. A procedure might continue despite the error code or might end.

error routine. A routine called when an error occurs while running a process option that accesses a record. If no error routine is specified, an application ends when an error occurs and displays a message describing the error condition.

EXECUTE option. A process option that causes processing statements to be performed. This option can be used for (1) non-input/output processing such as setting conditions to control the flow between processes or (2) one time functions such as initialization or termination. It has no process object.

execution SQL process table. A table that contains the process-related information needed to issue SQL requests while an application runs.

execution SQL record table. A table that contains the record-related information needed to issue SQL requests while an application runs.

export utility. A CSP/AD utility that copies member specification library (MSL) members to an application load file (ALF) or a serial file. See also *import utility*.

express PCB. An alternate program communication block (PCB) that send a completed message immediately to its destination. When a failure occurs, a message sent to an express PCB cannot be backed out. See also *alternate PCB*.

extended dynamic mode. A mode of running SQL applications on SQL/DS systems. In this mode SQL statements are prepared once by an application generator or support program and extended dynamically without needing to be prepared again. The statements are passed to the database manager one at a time. They are then prepared and stored in an SQL/DS access module associated with the SQL application.

external branching statement. A statement that transfers control from a running program either to another CSP/AD application or to a non-CSP/AD program.

external source format. A feature of the Cross System Product family that makes it possible to import definitions developed outside CSP/AD. The format consists of a readable syntax of mark-up tags and attributes. The external source format is also used to export MSL members for hard copy editing and analysis. See also *Internal format*.

EZ-PREP. A feature of the Cross System Product set that, on a personal computer, imports and generates an application. It also imports messages and data files from a host to a personal computer.

EZ-RUN. A feature of the Cross System Product set that, on a personal computer, runs a CSP/AD application.

F

facility. An operational capability or the means for providing such a capability. CSP/AD facilities provide the tools or capabilities to define data, maps, and processing statements, to test and generate, and to select, maintain, or define members in member specification libraries (MSLs).

fast path. A type of IMS/VS processing that expedites handling of certain transactions and supports special databases designed for large volumes of data with a high availability rate, such as DEDBs and MSDBs.

FCT. See *file control table*.

field attribute. A character code that can be specified to determine the appearance of a constant or a variable

field on a map. The default code symbols can be changed.

field name. A data item name as it appears in a map definition. In maps, a data item name can be prefixed by EZE if the name is EZEMSG. A field name can be from 1 to 32 characters where the first character must be alphabetic or national (A-Z, #, \$, @) and the remaining characters are alphanumeric, national (A-Z, 0-9, #, \$, @), hyphen, or underscore. No special characters, embedded blanks, or EZE prefixes are allowed. A field name can be a DBCS name. See also *DBCS name*.

field outlining. A map field characteristic that permits horizontal and vertical bars to be drawn around a field on a map on devices displaying double-byte character data.

file. A collection of information or data that is organized by some method and stored on a device such as a disk. See also *indexed, relative, and serial*.

file connect name. The name of a file in a physical data set that must be associated or connected to a file ID that is specified for an application.

file control. In COBOL, the name and header of an environment division paragraph in which the data files for a given source program are named and assigned to specific input/output devices.

file control table (FCT). In CICS/VS, a table containing the characteristics of files accessed by file control.

file ID. A field that contains the file name with an associated record definition during data definition.

file I/O. An attribute that displays the results for every I/O process.

file name. The name used to associate a record specification with a physical file. It can be 1 to 8 characters where the first character must be alphabetic or national (A-Z, \$, #, @) and the remaining characters must be alphanumeric or national (A-Z, 0-9, \$, #, @). No special characters, embedded blanks, or EZE prefixes are allowed.

file specification. A CSP/AD panel that is used to identify the file associated with a record definition. See also *alternate specification, organization, record ID item, and record name*.

fill character. A character that is used in place of blank or empty positions in the variable field of a displayed map. If N is specified, nulls are used as fill characters.

filler. A data item given an asterisk for a name in a record or table definition. A filler is used to define

space in a record or table row that is not to be named or referenced.

FIND statement. A conditional statement that performs a statement group or process depending on whether the data item is found in the search column of a table.

first found member. The member in the member specification library (MSL) with a specific name that is located first when searching the concatenated MSLs in order from 1 to 6.

fixed maps. Maps with a specific starting line and column number.

floating map. A map that has no fixed starting position and is displayed without any consideration to any partial fixed maps or full fixed maps that might be displayed. A floating map must fit the defined floating area for an application's map group. It occupies the next available space in a defined floating area not already occupied by a floating map.

flow stage. The connecting logic between processes in an application that contains instructions that control the sequence of running the main processes in an application. If no flow stage statements are defined, the next main process in the list is performed by default.

flow statement. A statement that identifies the next main process to be started.

fold. To change lowercase single-byte character set (SBCS) alphabetic characters to uppercase SBCS alphabetic characters.

fold input. An attribute that specifies whether all data a user types on a map is to be folded to uppercase or only selected fields.

function. A part of a CSP/AD facility. See also *facility*.

function code. A code that tells DL/I whether to get, insert, replace, or delete segments from the database. Part of the parameter list of the DL/I call.

function word. See *special function word*.

G

generalized sequential access method (GSAM). Allows IMS/VS batch applications and batch message processing (BMP) programs to access a sequential OS/VS data set as a database. The database is a root-only database, and the entire root segment represents a record. GSAM files can have checkpoints and can be restarted like other DL/I databases.

generate. (1) To produce a form of an application, map group, or table suitable to run under CSP/AE and CSP/370 Runtime Services. (2) To build the EZ-RUN files set up during the importing procedure by loading data objects, scanning processes, and statement groups, and then by resolving symbolic references.

generated application. (1) A CSP/AD application that has been processed in the host environment so that CSP/AE can run the application interpretively in the CICS/DOS/VS, VSE Batch, CICS/OS/VS, MVS Batch, VM/SP, DPPX/SP, MVS/TSO, or OS/400 environments. (2) A CSP/AD application that has been processed with the CSP/370 Runtime Services to run in the IMS/VS, IMS/ESA, and MVS Batch environments.

generation facility. A facility that makes a defined application ready to run. Data items, maps, and CSP/AD statements are put into a form usable by CSP/AE for the environment where the application will be used.

GSAM. See *generalized sequential access method*.

GSAM PCB. This program communication block (PCB) represents a generalized sequential access method (GSAM) database that a program can access; this PCB also contains the processing option (get or load) available for the program.

H

hard error. In Cross System Product, a fatal error that causes the application to end. However, by specifying EZEFEK the developer can choose to have the application continue. Contrast with *soft error*.

HEX. See *hexadecimal data*.

hexadecimal data (HEX). The data item type is called HEX, which stands for hexadecimal base 16 numbering and can represent any possible bit pattern in a byte string.

hierarchy. The relationship between different types of segments in a DL/I database. The relationship is a top down tree structure. The segment at the top of the structure is called the root segment. Each segment can have one or more child segments related to it at the next level in the structure. A segment with a child is called the parent of the child. Each segment in the hierarchy, except for the root, has only one parent. The root segment has no parent.

host variable. In an application, a variable referenced by embedded SQL statements.

IBM Database 2 (DB2). A program that manages relational databases on MVS systems.

IF statement. A statement that marks the start of a set of processing statements that control conditional execution of logic dependent on the results of one or more comparisons.

Immediate program switch. A type of program switch in which the first program passes control directly to another transaction, which is associated with a second program, without first responding to the originating terminal. For a conversational message processing program (MPP), the program does this by inserting the scratchpad area (SPA) to an alternate PCB that has its destination set to the new transaction name. For a nonconversational MPP, the program inserts a message to an alternate PCB that has its destination set to the new transaction name.

Import utility. (1) A function that allows a member specification library (MSL) to receive members that are copied from a source data set, which contains exported MSL members. (2) A utility that processes a CSP/AD application that was exported from the host. It converts it into EZ-PREP applications by reading sequential CSP/AD application files, converting data to ASCII, and building personal computer files necessary for generation.

IMS/DB. See *Information Management System/Database*.

IMS/DC. See *Information Management System/Data Communications*.

IMS/VS. See *Information Management System/Virtual System*.

Index. The key to the location of a record is contained in a portion of the file called the index. The index contains record IDs whose structural location is identified by the record ID item at file specification. See also *indexed*, *record ID*, *relative*, and *serial*.

Indexed. The organization specified in record definition file specification for records that are accessed by a key (record ID).

Indexed file. (1) A file in which the position of a record is recorded in a portion of the file called an index. The index contains an index key and disk address for each record in the file. (2) In a database, a file whose access path is built on key values. Each record in the file is identified by a key field. See also *relative file*, *serial file*, and *record ID*.

Information Management System/Database (IMS/DB). The database feature of IMS/VS. DL/I is the database

manager and the means used to create and access hierarchical databases.

Information Management System/Data Communications (IMS/DC). The data communications feature of IMS/VS. It brings the databases into an online environment to permit users interactive access. It manages transactions, terminals, and the flow of data among these resources.

Information Management System/Virtual System (IMS/VS). A database and data communications system capable of handling a large, complex network of databases and terminals. It consists of a database feature (IMS/DB) and a data communications feature (IMS/DC).

Initialization. The setting of data items, records, or map items to starting values at the beginning of, or at specified points in, the operation of an application.

Input editing. Editing that takes the text a user types on a map, and then removes the symbols, right-justifies numeric data, and fills numeric fields with zeros as the data is stored in its internal storage.

Input field. An unprotected field on a map where data can be typed, changed, or deleted.

Input/output (I/O). A function on a device that handles the transfer of data between applications and terminals, printers, data sets, or host communication support systems.

INQUIRY option. A process option that obtains data contained in a record.

Insert. An attribute that copies an input file (containing material that was extracted from another system) into an application load file (ALF).

Interactive test facility. A facility that runs an application directly from the definitions that are stored in a member specification library (MSL).

Internal branching statement. A statement that transfers control of a running program to either an internal process or a statement group.

Internal source format. A format used during importing and exporting to transfer definitions between MSLs and ALFs. The internal format file is either an ALF member or a serial file with 256-byte fixed-length records. See also *external source format*.

INTO clause. (1) A clause that specifies that the selected columns are to be read into the data items in the SQL record definition. (2) A clause that identifies the data items that receive the data when a row is retrieved with the FETCH command associated with the cursor.

I/O. See *input/output*.

I/O area address. Address of the buffer that contains the segment after it is read from the database or before it is written to the database. The I/O area address is part of the parameter list of the DL/I call.

I/O PCB. In IMS/VS, an input/output (I/O) program communication block (PCB) that represents an IMS/VS logical terminal. DL/I calls for an I/O PCB read input from a terminal and write output messages back to the same terminal. Contrast with *alternate PCB*.

Isolation level. An attribute that shows the degree of independence one application has from another application. There are two levels of isolation: repeatable read and cursor stability. In a CSP/AD application isolation level is controlled using the EZESQISL special function word.

Item edit table. A table that contains the edit specifications for each data item defined for the maps.

J

Join. An SQL term only, join is a relational operation that allows retrieval of data from two or more tables by matching column values.

Justify. An attribute that positions data in a map field during output if data is moved to the map field with fewer characters or numbers than the defined length for the map field.

K

key. A data item or record ID used to identify a record and establish its order with respect to other records in an indexed file, a DL/I database, or a relational database.

key item. The name of an item that contains the segment key.

keyword. (1) A part of a processing statement or command operand that consists of a specific character string. (2) CSP/AE uses a type of keyword that consists of a set of alphanumeric characters to describe a CSP/AE program failure.

L

length. The number of characters or numeric digits for a data item or map field.

level. (1) The degree of subdivision of a data structure. In CSP/AD, level numbers subdivide a record, table, or working storage definition. See also

level-77. (2) On the Structure List panel, level is a number that represents the position of the process or statement group in the nested structure.

level-77. A level that can be defined for unstructured or single data items in working storage.

line editing. Allows an application developer to manipulate lines on a CSP/AD displayed list.

linkname. A name that consists of 1 to 8 characters that is assigned to the linked EZ-RUN application.

list processor facility. The CSP/AD facility that can be used to manage member definitions by displaying a directory list of the member specification library (MSL). See also *list processor options*.

list processor options. One-character codes that define the functions that are available to manage members in the member specification library (MSL).

load catalog module. A program unit that is suitable for loading into main storage for execution. See also *application load file*.

local system. A system that contains the files and definitions that are used to generate an application. See also *remote system*.

locking. An internal mechanism used to protect data. It ensures that no other user or application can change the data currently being used.

log file. A file that logs each command that runs, each message, and the return code from each command.

logic. The processing of data defined in processing blocks that access records and display maps.

logic design. A functional design that uses formal methods of description such as symbolic logic.

logical record. A record defined according to its content, function, and use rather than its physical attributes; that is, a record defined in terms of the information it contains.

logical unit of work (LUW). The unit of recovery for recoverable files and databases. At the completion of a logical unit of work (LUW), all database changes are guaranteed to be either complete (changes were committed) or not done at all (changes were undone by a rollback). Also, no database changes made during a logical unit of work are visible to other users until the changes are committed.

LUW. See *logical unit of work*.

M

main batch. A type of batch application that is the initial application that receives control from CSP/AE.

main storage database (MSDB). A type of IMS fast path database that uses fixed-length root segments that reside in virtual storage for quick access. The segments can be defined for access by a specific terminal or by all terminals.

main transaction. A type of application that a user interacts with at a display device.

map. In CSP/AD a definition of the format of all or part of what will appear on a screen or printer page. A map is used to define the organization and characteristics of information that is presented on a display or printer while running an application.

map definition. A facility that helps to create the panel and report formats as they will appear in a production environment.

map field. An area defined on a map where data can be entered or displayed. See also *constant field*, *data item*, *field attribute*, *input field*, and *variable field*.

map formatting. On the CSP/AD map presentation panel, a user is prompted to design the maps for an application. CSP/AD codes and commands can be used when defining a map.

map group. Contains all of the maps used by an application. Maps in a map group can be defined and used in other applications. See *map group name*.

map group name. A unique identification of a map group. The name must be 1 to 6 characters with the first character as alphabetic or national (A-Z, \$, #, @) and the remaining characters as alphanumeric or national (A-Z, 0-9, \$, #, @). No special characters, embedded blanks, or EZE prefixes are allowed.

map name. A unique identification of a map that can be 1 to 8 characters. The first character must be alphabetic or national (A-Z, \$, #, @) and the remaining characters must be alphanumeric or national (A-Z, 0-9, \$, #, @). No special characters, embedded blanks, or EZE prefixes are allowed.

map position. An attribute that defines the starting point for a map on a screen using lines and columns. Map position is specified at the same time as the map size. See also *floating map*.

map size. A parameter used to define the size of a map that will be displayed to a user using lines and columns. The line and column fields contain CSP/AD default values based on the sizes of the devices chosen

for the map. The line and column values can be changed, but they cannot exceed the maximum values available for a device.

map specification. All the parameters used to define and describe a map and how it appears on a screen.

map structure. All the parameters that define the fields on the map and the various edits that can be associated with those fields.

match invalid table. A type of table that can be used as an edit table. A match invalid table compares the input data with the first column of the table, which contains a list of the invalid values for a map field.

match valid table. A type of table that can be used as an edit table. A match valid table compares the input data with the first column of the table, which contains a list of valid values for a map field.

MDT. See *modified data tag*.

member. Part of a member specification library (MSL), consisting of the name and a complete definition. Examples of MSL members are map, record, and process definitions.

member print utility. A utility that produces a listing of specified members in a member specification library (MSL).

member specification library (MSL). A single library containing all of the information that is supplied to and defined by CSP/AD. The MSL contains all the CSP/AD definitions such as data items, maps, and application definitions that are used to generate an application.

message file. An optional, user-defined file that contains user-created messages and can be defined through the CSP/AD message file utility program. The message will be displayed in a defined field on a map while running an application when an error occurs.

message format services (MFS). An editing facility used with IMS/VS that allows programs to deal with only message data from a terminal. MFS allows the user to customize the panel presentation of the data, but shields the program from panel formats and device dependencies by providing access to only the data required from the terminal.

message processing program (MPP). An IMS/VS program used to process requests from terminals and from other applications. These messages are stored in message queues accessible to the MPP. The MPP can perform required database access and write new messages to output queues for further terminal and program processing. The MPP cannot access OS/VS or GSAM files. See also *conversational MPP* and *nonconversational MPP*.

message queue. A place in IMS/VS where information being sent to an alternate terminal or another transaction can be placed so that IMS/VS can handle the I/O to the terminal or schedule the other transaction to start processing.

MFS. See *Message Format Services*.

MIX. See *mixed data*.

mixed constant code (%). The default code symbol that specifies the starting position of a mixed character constant field made up of single-byte characters and double-byte characters. The default code symbol can be changed.

mixed data (MIX). A character string where portions are in single-byte (SBCS) and portions are in double-byte (DBCS) character sets. The DBCS portion is enclosed with a shift-out (SO) character and a shift-in (SI) character to distinguish it from SBCS.

mixed literal. Literal data enclosed in quotes, single or double, that has occurrences of SO/SI characters within the data.

mixed variable code (|). The default code symbol that specifies the starting position of a mixed-character variable field made up of single-byte characters and double-byte characters. The default code symbol can be changed.

modified data tag (MDT). A CSP/AD function that can be set either by the MDT characteristic or a SET MODIFIED statement. The modified data tag causes the contents of a map field to be passed to an application the next time the map that contains that field is displayed. The passed data is preset and the user can choose to accept the displayed data in the map field or change it before it is sent to the application.

MOVE statement. A processing statement that moves the contents of one item to another or that moves the contents of corresponding items in a data structure to another structure.

MOVEA statement. A processing statement that moves the contents of one array to another array or that initializes the elements of an array.

MPP. See *message processing program*.

MSDB. See *main storage database*.

MSL. See *member specification library*.

MSL maintenance. A facility that allows members in the member specification library (MSL) to be copied, deleted, or renamed.

MSL member. All of the CSP/AD definitions including record, table, program specification block, data item,

map, map group, process, statement group, and application definitions that are used to generate an application.

MSL selection facility. The facility that provides a way to select and change the number, order, and the way that member specification libraries (MSLs) are available for review and edit.

N

national characters. The characters #, \$, and @.

NLS. National language support.

nonconversational MPP. A type of message processing program (MPP) that processes a single input message, with at most a single response. Only the data on the panel or in a database is saved.

nonprocedural element. An element that has no syntax and is specified on preformatted panels or lists.

nonsegmented mode. See *CICS/VS conversational mode*.

normal. The default for the light intensity field attribute for the text displayed on a map.

null. (1) When used with a relational database, a NULL value shows that no data value has been assigned to the intersection of a row and a column in a relational table. (2) When used with a map data item, NULL is equivalent to pressing Erase EOF (end of file).

NUM. See *numeric data*.

number of occurrences item. Contains the actual number of occurrences of the variably occurring item.

numeric data. Data or physical quantities that are discrete representations of numbers. A mathematical entity that can suggest quantity or amount of units. See also *binary data, character data, data type, hexadecimal data, and packed data*.

O

object. A record or map that is used by a process option. The SQLEXEC and EXECUTE options are the only options that do not require an object to be specified.

occurs. The number of repetitions of a data item in a data structure.

operand. Information that is entered with a command name to define the data to be operated upon and executed within an application.

option. The function to be performed by a process. Only one option per process is allowed, and the EXECUTE option is the default. See also *ADD*, *CLOSE*, *CONVERSE*, *DELETE*, *DISPLAY*, *EXECUTE*, *INQUIRY*, *REPLACE*, *SCAN*, *SCANBACK*, *SETINQ*, *SETUPD*, *SQLEXEC*, and *UPDATE* options.

organization. The structure of the file or database associated with the record that determines how the record is to be accessed. Organization is specified during record specification. Organization can be defined as indexed, relative, serial, working storage, redefined record, DL/I segment, or SQL row.

output editing. Specified characteristics of how data should be formatted and appear when it is sent from the application to a display device. The following output editing formats are examples of what the user can specify: justify (right or left), fill character, decimal positions, sign, and currency symbol.

P

packed data. Decimal data that is packed when stored to save space.

panel. The formatted information presented on the viewing surface of a terminal display device.

parameter group. A facility that provides the ability to control the invocation parameters used to invoke CSP/AD or CSP/AE.

parent segment. A segment with a child segment related to it in a DL/I database hierarchy. See also *child segment*, *hierarchy*, and *segment*.

partial map. Fixed maps that do not have a depth equal to the size of the physical screen. See also *floating map*.

PCB. See *program communications block*.

PERFORM statement. An unconditional statement that specifies a process to be run as if it were included in-line in place of the PERFORM statement.

portability. The ability to use applications, data sets, or files on more than one computer without modifying them.

precompilation. A DB2 term, precompilation is the processing of programs containing DB2 requests that takes place before compilation. SQL statements are replaced with statements that will be recognized by the host language compiler. Output from this precompilation includes source code that can be submitted to the compiler and the DBRM that is input to the bind process.

preprocessor function. A function of the test and generation facilities that checks the syntax of and finds definition errors in CSP/AD statements that are defined for an application.

print utility. See *directory print utility* and *member print utility*.

procedural element. An element governed by a language structure or syntax.

process. A unit of an application. A process is divided into a process option and processing statements. Each process does one task of an application. Processes are connected by branching instructions in the flow stage of an application. Processes provide a modular design approach for application. See also *option*.

processing block. (1) A process statement that does a single input/output (I/O) operation. (2) A statement group is several processing statements, which can be used multiple times within an application.

processing load module. A module that contains the processing statements, map item edit tables, and the literal pool.

processing statement. A CSP/AD statement that is specified during application definition. Processing statements, when they run, perform the tasks for an application. Processing statements can be grouped in the following categories: arithmetic statements, data manipulation statements, external branching statements (unconditional), and internal branching statements (conditional and unconditional). See also *statement group*.

process name. A unique identification of a process in an application that can be 1 to 18 characters. The first character must be alphabetic or national (A-Z, \$, #, @). The remaining characters must be alphanumeric, national (A-Z, 0-9, \$, #, @), hyphen, or underscore. A process name can be a DBCS name. No special characters, embedded blanks, or EZE prefixes are allowed. See also *DBCS name*.

process object. The name of a record or map accessed by the process option.

process option. The function to be performed by a CSP/AD process. Process options except EXECUTE and SQLEXEC represent input/output operations.

process stage. A part of a process that contains processing statements that are to be performed before or after a process option.

process statement table (PST). A table that contains the processing statements used in the application for each process and statement group.

program. A sequence of instructions suitable for processing by a computer. See also *application*.

program communications block (PCB). An entry in a DL/I program specification block. Each PCB describes one hierarchical database structure that an application can access. In the IMS/VS environments, PCBs are also used for the IMS message queue and GSAM databases. See also *alternate PCB*, *database PCB*, *express PCB*, *I/O PCB*, and *GSAM PCB*.

program failure keyword. A CSP/AD and CSP/AE keyword that identifies the IBM licensed program that failed.

program specification block (PSB). A formal DL/I description of the hierarchical database structures that an application can access. CSP/AD supports the definition of a member in the member specification library (MSL) that contains a subset of the information in the DL/I PSB. The PSB shows the hierarchical relationship that exists between types of segments. In the IMS/VS environment, program communication blocks (PCBs) for the IMS message queue or GSAM databases are also included. See also *program communications block (PCB)*.

programmable workstation (PWS). A workstation that has some degree of processing capability and that allows a user to change its functions.

prologue. A user-written description of definitions that have been defined for an application, record, or table; its text is limited to 60 characters per line.

PSB. See *program specification block*.

PSB name. A unique identification of a program specification block (PSB) in an application that can be 1 to 8 characters. The first character must be alphabetic or national (A-Z, \$, #, @) and the remaining characters are alphanumeric or national (A-Z, 0-9, \$, #, @). No special characters, embedded blanks, or EZE prefixes are allowed.

pseudoconversational mode. The CICS term for running an application in segmented mode. A series of single CICS transactions designed to appear to the operator as a continuous conversation. See also *segmented mode*.

PST. See *process statement table*.

PTF. See *program temporary fix*.

PWS. See *programmable workstation*.

Q

qualification statement. A string in a segment search argument (SSA) that specifies a condition that must be satisfied if a segment is to be retrieved. The qualification statement consists of the name of a field in the segment, a relational operator, and a comparison value. DL/I compares the value in the segment field to the comparison value using the operator to determine if the condition is met.

R

range. A map field characteristic that specifies the minimum value and maximum value that are valid for a numeric data item when input is entered on a map. Tables can be defined with range values and used in edit checks of data that is input for both character and numeric data.

range match valid table. A type of table that can be used as an edit table. A range match valid table compares input data with the first and second columns of the table, which contain lists of valid ranges of values for a data item.

record. A collection of related data items treated as one unit. Records are the information units that form a file or database.

record ID. A data item whose contents show the location of a record in a relative or indexed file. A data item that is required to be part of a record and is used directly to access records. See also *indexed*, *key*, and *relative*.

record ID item. The name of a data item that contains a record ID.

record key. A field within the first block of each record in an indexed data set that is used in storing and retrieving records in the data set.

record length item. The record length item contains the current record length.

record name. A unique identification that can be 1 to 18 characters. The first character must be alphabetic or national (A-Z, \$, #, @); the remaining characters must be alphanumeric, national (A-Z, 0-9, \$, #, @), hyphen, or underscore. A record name can be a DBCS name. No special characters, embedded blanks, or EZE prefixes are allowed. See *DBCS name*.

record specification. All the parameters used to define the organization of the record along with other options such as file name and record ID item.

redefined record. (1) A record that allows you to define multiple structures for a record within a file. (2) An alternate item structure for an existing record.

reference file. A data set that lists the application that will identify where errors occurred during execution. This file can be saved or printed.

reference table. A table that stores information that can be used by an application.

relational database. (1) A data structure perceived by its users as a collection of tables. (2) A database that is organized and accessed according to relationships between data items. (3) A database where relationships between data items are explicitly specified as equally accessible attributes.

relative. A file organization specified in record definition file specification for records that are accessed by a record ID number. See also *indexed* and *serial*.

relative file. A file that contains records that are identified by record ID numbers. The record ID is not required to be a part of a record, but it is defined as the record ID item during file specification. See also *indexed file* or *serial file*.

relative record data set (RRDS). In VSAM, a type of data set that is divided into fixed length slots. Each slot corresponds to a relative record number. Each record that is added to an RRDS is assigned a relative record number and stored in its corresponding slot.

relative record number. The content of the record ID item used to identify a record in a file with relative organization.

remote system. A system that is not in a local system and may contain records that are used by an application that resides in a local system. See also *local system*.

rename utility. A utility that causes the name of a member in the member specification library (MSL) to be changed.

repeatable read. One of the two possible settings for isolation level in which a logical unit of work (LUW) is protected from accessing uncommitted updates of another LUW. Also, no other LUW can modify any row that has been read by this LUW. This ensures that within a LUW, the values remain the same unless the user commits the updates.

REPLACE option. A process option that takes an updated or changed record and puts it back in the file for the record. An UPDATE or SCAN for update process option must occur before a REPLACE option.

resultant size. The number of occurrences from the starting position specified and the end of the array.

return code. A code used to influence how succeeding instructions run.

R-O. Read only.

rollback. The process of restoring changed data to the state at its last commit point. CSP/AD applications rollback changed data by calling the EZEROLLB service.

root segment. The segment at the top of a DL/I database hierarchy.

row. A sequence of values so that the *n*th value is the value of the *n*th column of the table. The relational equivalent of a record.

RRDS. See *relative record data set*.

R/W. Read and write.

S

SAA. See *Systems Application Architecture*.

SBCS. See *single-byte character set*.

SCAN option. A process option that retrieves records in a file or database in sequential order.

SCANBACK option. A process option that retrieves records in a file in reverse sequential order.

scratchpad area (SPA). In IMS/VS, storage used to save data during conversational processing while the screen is in use displaying data.

screen. Face of a nonprogrammable display device or terminal where a user sees information or a panel.

segment. See *application segment* and *DL/I segment*.

segmented mode. The mode of running interactive applications in which storage resources are released each time a map is conversed and the application is waiting for a response from the user. See also *conversational MPP* and *pseudoconversational*.

segment field. Identifies the name of the field used for segment selection.

segment search argument (SSA). A character string passed during a DL/I call used to locate a specific position at a certain level in the database.

segment sequence field. The field in a DL/I segment that determines the order in which the segment is stored and retrieved from the database. The sequence

field is used as a key field for quick retrieval of specific segments from the database.

SENSEG. In DL/I, statements in a program communication block (PCB) that describe which segments a program will use and what type of processing options are valid for those segments.

sequence field. The search field in a database segment used to store segment occurrences in sequential ascending order.

serial. A file organization in which records are accessed sequentially. See also *indexed* and *relative*.

serial file. A file containing records that are arranged sequentially. No record ID is required, and the file is accessed by reading one record after another until the desired record is found. See also *indexed file* and *relative file*.

service routine. A routine that usually supports the processes of a computer; for instance, an input routine.

SET statement. A processing statement used to position the cursor in a map field, change the attribute byte being used for a map field, set record item NULL for an SQL row record, clear a display, clear a record, cause a printer to skip to a new page, or establish a position in a file.

SETINQ option. A process option that selects a set of rows from a relational table for later retrieval by the SCAN option one row at a time. The object must be an SQL row record.

SETUPD option. A process option that selects a set of rows from a relational table for later retrieval by the SCAN option one row at a time. A row selected by SETUPD can be replaced or deleted following the SCAN that retrieves it.

share mode. A mode in which a resource is available for more than one user at the same time.

shared storage. Areas of highly used storage that can be shared between users.

shift-in character (SI). A code extension character used to signal the end of a sequence of double-byte character set (DBCS) ideographic characters within data.

shift-out character (SO). A code extension character used to signal the beginning of a sequence of double-byte character set (DBCS) ideographic characters within data.

SI. See *shift-in character*.

SID. SQL/DS user identifier.

signed data. A type of data that is displayed with a negative or positive sign. See also *data type*.

single-byte character set (SBCS). A set of characters for which a single character requires one byte of data for each character in a record, table, or map.

SO. See *shift-out character*.

soft error. (1) An intermittent error on a network that requires transmission again. (2) A nonfatal error that allows the application to continue. Contrast with *hard error*.

source interface utility. A utility that allows you to load record and DL/I definitions from a sequential data set directly into an application load file (ALF).

SPA. See *scratchpad area*.

spacer code (/). The default code symbol that is used for centering or evenly spacing fields on lines that are entered on a map during map definition. The default code symbol can be changed.

special character. A character other than a letter, a digit, or #, \$, @. For example, %, &, and + are special characters.

special function word. A word used by CSP/AD processing statements to provide additional internal branching and processing functions for an application. All CSP/AD special function words are 6 to 8 characters and begin with the EZE prefix.

SQL. See *Structured Query Language*.

SQLCA. See *SQL communications area*.

SQL column name. The name by which the column is known to the database manager.

SQL communications area (SQLCA). A collection of variables that are used by DB2 or SQL/DS to provide an application with information about running its SQL statements.

SQLDA. See *SQL description area*.

SQL data code. A number that identifies the data type of the data item to the database manager.

SQL description area (SQLDA). A data structure used to pass variable values between a program and SQL/DS or DB2 when an SQL statement is run.

SQL/DS. See *Structured Query Language/Data System*.

SQLEXEC option. A process option that runs an SQL statement written by the application developer.

SQL row. A record organization that represents a row in a relational table to a CSP/AD application.

SQL table name. The name of the relational table that the SQL row record represent.

SSA. See *segment search argument*.

statement. See *processing statement*.

statement group. A set of CSP/AD processing statements that perform processing only (no input/output operations are performed). When a statement group finishes running, control is returned to the application statement that started running the statement group.

statement group name. A unique identification of a statement group that can be 1 to 18 characters. The first character must be alphabetic or national (A-Z, \$, #, @), and the remaining characters must be alphanumeric, national (A-Z, 0-9, \$, #, @), hyphen or underscore. A statement group name can be a DBCS name. No special characters, embedded blanks, or EZE prefixes are allowed. See also *DBCS name*.

statement sequence numbers. Numbers shown on application listings that match those sequence numbers that are produced by the preprocessor or generator. Statement sequence numbers are also used for statement identification on the cross reference listing and on the printout of the application structure.

static mode. A mode of running SQL applications in which the SQL statements in the application are coded in an assembler or high-level language program. The program is precompiled so the database manager can prepare the SQL statements to run. The program is then compiled and linked. The SQL statements are run by calling the program. **Note:** CSP/AE supports static mode execution for SQL applications on DB2 systems. On SQL/DS systems, extended dynamic mode is used.

structure. See *data structure*.

Structured Query Language (SQL). A language that can be used with applications or interactively to access data in relational databases.

Structured Query Language/Data System (SQL/DS). An IBM program that manages relational databases on VM/SP and VSE systems.

subscript. An attribute that makes it possible to define an array.

substructure. A subdivision of a data item.

SYNCPOINT. In CICS/VS a point in the processing of a task (marked by a SYNCPOINT command, end of task, or DL/I TERM call) at which changes to recoverable resources are regarded as committed (written to a file).

Systems Application Architecture (SAA). A collection of selected software interfaces, conventions, and protocols for development of consistent applications across the three major IBM computing environments: System/370, AS/400, and PS/2.

T

table. A collection of related data items structured as a two-dimensional array of columns and rows. Two kinds of tables are used in CSP/AD applications: (1) The word table by itself refers to a member in the member specification library (MSL) that can be used in validating map inputs or identifying related factors for standard computations and (2) A relational table refers to a table in a relational database accessed by the Structured Query Language (SQL). A relational table is represented in an application by a record with SQL row organization.

table label. A shortened version of the table name used in associating a column name to table names in the SQL statements built for the record.

table load modules. A module that contains the contents defined for the user tables.

table name. A unique identification for a table that can be 1 to 7 characters. The first character must be alphabetic (A-Z) and the remaining characters must be alphanumeric (A-Z, 0-9). No national or special characters, embedded blanks, EZE prefixes, or names ending in 0 are allowed.

table specification. A parameter that allows the user to specify the type of table being defined.

table type. Available types of tables are match invalid, match valid, range match valid, or unspecified. Type only has meaning when the table is used as an edit table for a map field.

target. A unique name of a data set that is to be used by the export utility.

target data set. (1) A data set that contains exported members of a member specification library such as members that were moved by use of the CSP/AD export utility. (2) The target of an indexed file contains the data records as opposed to the index part, which contains the record keys, of an indexed file.

test facility. A CSP/AD facility that is used to test an application's definitions for correct syntax, logic design, and the logical sequence of maps presented when the application is run. See also *preprocessor function* and *test function*.

test function. A function of the test facility that aids in finding mistakes in the logic design of an application.

The test function uses test data sets set up by the user, CSP/AD definitions contained in the member specification library (MSL), and optional trace options to display or print the maps and processing statements as they would be encountered if the application were actually started and run. See also *test facility* and *trace option*.

TEST statement. A conditional statement that can be used to determine the state of an operand.

trace option. Used with the test function to specify sections of an application that should be examined during testing. The sequence in which instructions are run can be recorded and, thus, isolate programming errors. See also *test facility* and *test function*.

transaction. (1) A type of application that runs while interacting with the user via maps and terminal input/output (I/O) operations. (2) A type of input message processed by IMS/VS that contains a transaction code and some data. IMS/VS maintains a message queue for the transaction, and when data is on the queue, a single, predefined application and PSB are scheduled to process the transaction data.

transaction-oriented BMP. A batch message processing (BMP) program that accesses the message queue, databases, and operating system files for input. Output can be sent to databases, operating system files, or message queues.

transient data queue. A file that is serially organized on CICS/VS.

tutorial facility. A help facility that provides online information about CSP/AD facilities, maps, and error messages. The tutorial consists of topics that can be viewed sequentially by selecting the tutorial facility or directly by pressing the Help Key (PF1) when a panel or message is presented.

type. A parameter that specifies the internal format of data and determines how the data can be used.

U

UIB. See *user interface block*.

unconditional statement. A statement that is always performed.

unprotect. A field attribute that allows data to be entered in a field on a map.

UPDATE option. A process option that obtains data in a record for changing or deleting the data. If an UPDATE option is not followed by a REPLACE option or a DELETE option, a record lock will be outstanding. See also *REPLACE option* or *DELETE option*.

user interface block (UIB). In CICS/VS, a control block that passes information for an interface routine. The interface routine acts as the interface between a CICS/VS program and DL/I.

user storage. Storage that is made up of data areas, read/write tables, and buffers that are unique for each user.

utility. A function that provides support for manipulating members in a member specification library (MSL). Utilities can be used to move (export) an MSL member to a target data set, to include (import) MSL members into one MSL from a target data set, and to print, copy, rename, or delete MSL members. See also *export utility*, *import utility*, and *member*.

V

value. The contents of a data item, field, or intersection of a row and column in a table.

variable code (⌈). The default code symbol that is used to begin a variable field on a map. The default code symbol can be changed.

variable field. A field that can be used for input data that is entered by the user and output data that is supplied by the application.

variable length item. The name of a data item in a variable length DL/I segment record that contains the length of the segment, including the length item.

variable length record. A record having a length independent of the length of other records with which it is logically or physically associated.

variably occurring item. The last item in a record that is not subordinate to any other item. The variably occurring item can be substructured and used with both indexed and serial records.


view. An alternate and temporary representation of data from one or more relational tables.

W

wait state. The interval between the instant at which an instruction control unit initiates a call for data and the instant at which the actual transfer of data begins.

where used. A list processor function that locates all references to the selected member and generates a new list of those members in which the reference occurs.


WHILE statement. A processing statement that provides for repetitively running a block of statements as long as a condition or multiple conditions are met.



work database. In IMS/VS, it supplements the scratchpad area (SPA). During conversational processing, a program may request a small SPA and supplement it by using a work database. IMS/VS does no special processing for a work database; it is managed by the program and used like any other database. A work database may be a DL/I database or a relational database.


working storage. A record definition used by an application to temporarily hold data. While running an application, working storage can be accessed and used to pass data to other applications.

X





XFER statement. A transfer statement that starts running another CSP/AD application, non-CSP/AD program, or transaction.

Z



zero edit. A parameter that specifies the display format for numeric fields that have zero values.



Special Characters

42nd-ward character. A double-byte character that contains X'42' in the first byte and has a corresponding EBCDIC value in the second byte.

#. See *constant code*.

+. See *DBCS constant code*.

@. See *DBCS variable code*.

%. See *mixed constant code*.

|. See *mixed variable code*.

/. See *spacer*.

┘. See *variable code*.

Index

A

ADD process option 3-2
AFTER tag 3-5
ALTSPEC attribute 5-1
ANSIGEN attribute 2-11
ANSIMOD attribute 2-11
APPL tag 2-1
AREA tag 10-2
ASYNC attribute 2-19
attributes
 ALTSPEC 5-1
 ANSIGEN 2-11
 ANSIMOD 2-11
 ASYNC 2-19
 BOOLOP 3-11
 BYPKEY
 on APPL tag 2-2
 on MAP tag 9-2
 BYTES
 on CFIELD tag 9-6
 on CONTITEM tag 6-8
 on DEFITEM tag 6-6
 on ITEM tag 7-1
 on RECDITEM tag 5-11
 on VFIELD tag 9-10
 CLAUSE 3-6
 CLAUSE TEXT
 on JOINCON tag 5-8
 on SQL tag 3-7
 CLIST 2-11
 CMDCODES 3-9
 COLNAME 5-11
 COLOR
 on CATTR tag 9-7
 on VATTR tag 9-17
 COLUMN
 on CFIELD tag 9-6
 on VFIELD tag 9-10
 COMPVAL 3-11
 CONSTANT 9-4
 CREATOR 5-6
 CREATREF 2-11
 CURRSYMB
 on MAPEDITS tag 7-3
 on VFIELD tag 9-12
 CURSOR
 on CATTR tag 9-7
 on VATTR tag 9-17
 DATA
 on CATTR tag 9-8
 on VATTR tag 9-18
 DATACODE 5-11
 DATE
 on APPL tag 2-2

attributes (*continued*)

 DATE (*continued*)
 on GROUP tag 4-1
 on ITEM tag 7-1
 on MAP tag 9-2
 on MAPG tag 10-1
 on PROCESS tag 3-1
 on PSB tag 8-1
 on RECORD tag 5-2
 on TBLE tag 6-1
 DATEFORM
 on MAPEDITS (item) tag 7-4
 on MAPEDITS (map) tag 9-12
 DBCSCONS 9-4
 DBCVAR 9-4
 DBDNAME 3-8
 DBNAME 8-2
 DB2GEN 2-11
 DB2MOD 2-11
 DDSGEN 2-12
 DECIMALS
 on CONTITEM tag 6-9
 on DEFITEM tag 6-6
 on ITEM tag 7-2
 on RECDITEM tag 5-11
 on VFIELD tag 9-10
 DEFFOLD 9-5
 DESC
 on DEFITEM tag 6-6
 on GROUP tag 4-1
 on ITEM tag 7-2
 on PROCESS tag 3-2
 on RECDITEM tag 5-11
 on VFIELD tag 9-10
 DETECT
 on CATTR tag 9-8
 on VATTR tag 9-18
 DEVICE 10-2
 DEVICES 9-2
 EDITMSG
 on MESSAGES tag 7-7
 on VFIELD tag 9-16
 EDITORDR 9-11
 EDITRTN
 on MAPEDITS (item) tag 7-4
 on MAPEDITS (map) tag 9-13
 ENTER
 on CATTR tag 9-8
 on VATTR tag 9-18
 ERRRTN 3-2
 EVENSQ
 on DEFITEM tag 6-6
 on ITEM tag 7-2
 on RECDITEM tag 5-12

attributes (continued)

EXECBLD 3-2
EXECMODE 2-12
field text 9-6
FILELOC 5-2
FILENAME
 on GENFILE tag 2-16
 on RECORD tag 5-2
FILETYPE 2-17
FILL
 on CATTR tag 9-8
 on VATTR tag 9-18
FILLCHAR
 on MAPEDITS (item) tag 7-4
 on MAPEDITS (map) tag 9-13
FIRSTMAP 2-2
FLDFOLD
 on MAPEDITS (item) tag 7-4
 on MAPEDITS (map) tag 9-13
flow statement 2-5
FOLD
 on GENOPTS tag 2-12
 on TBLE tag 6-1
GENGRP1 2-12
GENGRP2 2-12
GRPNAME
 on MAP tag 9-2
 on MAPG tag 10-1
HELPGRP 2-2
HELPKEY
 on APPL tag 2-2
 on MAP tag 9-2
HELPMAP 9-3
HEXEDIT
 on MAPEDITS (item) tag 7-4
 on MAPEDITS (map) tag 9-13
HILITE
 on CATTR tag 9-8
 on VATTR tag 9-18
HOSTTRAN 2-19
HOSTVAR
 on JOINCON tag 5-8
 on SQL tag 3-6
HTFERITM 2-19
HTFWKITM 2-20
IKEY 8-3
IMPLICIT 2-2
INDEX 9-11
initial value 9-11
INPUTREQ
 on MAPEDITS (item) tag 7-5
 on MAPEDITS (map) tag 9-13
INTENSE
 on CATTR tag 9-8
 on VATTR tag 9-18
INVALMSG
 on MESSAGES tag 7-7
 on VFIELD tag 9-16

attributes (continued)

JUSTIFY
 on MAPEDITS (item) tag 7-5
 on MAPEDITS (map) tag 9-14
KEEP 2-14
KEY
 on RECDITEM tag 5-12
 on RECORD tag 5-2
LABEL 5-6
LEVEL
 on DEFITEM tag 6-6
 on RECDITEM tag 5-12
LOADLIB
 on GENOPTS tag 2-12
 on TBLE tag 6-3
MAPGROUP 2-2
MAPGRP1 2-12
MAPGRP2 2-12
MAPNAME 9-3
MAPSIZE 9-3
MDT
 on CATTR tag 9-8
 on VATTR tag 9-18
MININMSG
 on MESSAGES tag 7-7
 on VFIELD tag 9-16
MININPUT
 on MAPEDITS (item) tag 7-5
 on MAPEDITS (map) tag 9-14
MIXCONS 9-5
MIXVAR 9-5
MODEL 3-2
MSGFILE 2-2
NAME
 on APPL tag 2-3
 on CALLPARM tag 2-7
 on CONTITEM tag 6-9
 on DEFITEM tag 6-7
 on GENTABLE tag 2-14
 on GROUP tag 4-2
 on ITEM tag 7-2
 on MAINPRC tag 2-5
 on PROCESS tag 3-2
 on PSB tag 8-1
 on RECDITEM tag 5-12
 on RECORD tag 5-3
 on TABREC tag 2-6
 on TBLE tag 6-2
 on VFIELD tag 9-11
NUMOCCUR 5-3
NUMSEP
 on MAPEDITS (item) tag 7-5
 on MAPEDITS (map) tag 9-14
OBJECT 3-2
OCCURS 5-13
OPTION 3-2
ORG 5-3
OUTLINE
 on CATTR tag 9-9

attributes (*continued*)

OUTLINE (*continued*)

- on VATTR tag 9-19
- PARENT 8-3
- PCBNO 2-17
- PFEQUATE 2-3
- PRINT 2-12
- PROTECT
 - on CATTR tag 9-9
 - on VATTR tag 9-19
- PSB
 - on APPL tag 2-3
 - on DLICALL tag 3-8
- RANGE
 - on MAPEDITS (item) tag 7-5
 - on MAPEDITS (map) tag 9-14
- RANGEMSG
 - on MESSAGES tag 7-7
 - on VFIELD tag 9-16
- READONLY 5-13
- REDEFREC 5-5
- REFINE
 - on GROUP tag 4-2
 - on PROCESS tag 3-3
- RELOP 3-12
- REQMSG
 - on MESSAGES tag 7-8
 - on VFIELD tag 9-16
- ROW
 - on CFIELD tag 9-6
 - on VFIELD tag 9-11
- SCANPAR 3-8
- SCANUPD 3-8
- SCOPE
 - on DEFITEM tag 6-7
 - on RECDITEM tag 5-13
 - on RECORD tag 5-5
 - on TBLE tag 6-2
- SEGFIELD 3-12
- SEGMENT 8-4
- SEGNAME 3-10
- SEGTRAN 2-13
- SIGN
 - on MAPEDITS (item) tag 7-5
 - on MAPEDITS (map) tag 9-14
- SINGROW 3-3
- SIZE 10-2
- SOSIPOS 9-3
- SPACER 9-5
- SQLBLOCK 2-13
- SQLGEN 2-13
- SQLID 2-13
- SQLMOD 2-13
- STARTPOS
 - on AREA tag 10-2
 - on MAP tag 9-3
- SYSNAME 2-17
- SYSTEM
 - on GENFILE tag 2-17

attributes (*continued*)

SYSTEM (*continued*)

- on TARGSYS tag 2-15
- TABLEGEN 2-14
- TABLEID 5-6
- TABPOS 9-5
- TABTYPE 6-2
- TBLENAM 5-6
- TIME
 - on APPL tag 2-3
 - on GROUP tag 4-2
 - on ITEM tag 7-2
 - on MAP tag 9-3
 - on MAPG tag 10-1
 - on PROCESS tag 3-3
 - on PSB tag 8-1
 - on RECORD tag 5-5
 - on TBLE tag 6-2
- TYPE
 - on APPL tag 2-3
 - on CALLPARM tag 2-8
 - on CFIELD tag 9-6
 - on CONTITEM tag 6-9
 - on DEFITEM tag 6-7
 - on ITEM tag 7-2
 - on PCB tag 8-2
 - on RECDITEM tag 5-13
 - on TABREC tag 2-6
 - on VFIELD tag 9-11
- TYPEUSE 6-3
- UPDPROC 3-3
- VALIDLOC 2-13
- VALIDSQL 2-13
- VARFOLD 9-5
- VARIABLE 9-5
- VARLENT 5-5
- WORKSTOR 2-4
- ZEROEDIT
 - on MAPEDITS (item) tag 7-6
 - on MAPEDITS (map) tag 9-14

B

batch commands

- CSP/AD A-3
- CSP/AE A-5
- batch log file A-3
- BEFORE tag 3-4
- BOOLOP attribute 3-11
- BYPKEY attribute
 - on APPL tag 2-2
 - on MAP tag 9-2
- BYTES attribute
 - on CFIELD tag 9-6
 - on CONTITEM tag 6-8
 - on DEFITEM tag 6-6
 - on ITEM tag 7-1
 - on RECDITEM tag 5-11

BYTES attribute (*continued*)
on VFIELD tag 9-10

C

CALLPARM tag 2-7
CATTR tag 9-7
CFIELD tag 9-6
CLAUSE attribute 3-6
CLAUSE TEXT attribute
on JOINCON tag 5-8
on SQL tag 3-7
CLIST attribute 2-11
CLOSE process option 3-2
CMDCODES attribute 3-9
COLNAME attribute 5-11
COLOR attribute
on CATTR tag 9-7
on VATTR tag 9-17
COLUMN attribute
on CFIELD tag 9-6
on VFIELD tag 9-10
command codes 3-9–3-10
COMPVAL attribute 3-11
CONSTANT attribute 9-4
CONTITEM tag 6-8
CONVERSE process option 3-2
CREATOR attribute 5-6
CREATREF attribute 2-11
CURRSYMB attribute
on MAPEDITS (item) tag 7-3
on MAPEDITS (map) tag 9-12
CURSOR attribute
on CATTR tag 9-7
on VATTR tag 9-17

D

DATA attribute
on CATTR tag 9-8
on VATTR tag 9-18
DATACODE attribute 5-11
DATE attribute
on APPL tag 2-2
on GROUP tag 4-1
on ITEM tag 7-1
on MAP tag 9-2
on MAPG tag 10-1
on PROCESS tag 3-1
on PSB tag 8-1
on RECORD tag 5-2
on TBLE tag 6-1
DATEFORM attribute
on MAPEDITS (item) tag 7-4
on MAPEDITS (map) tag 9-12
DBCS support B-1
DBCSONS attribute 9-4

DBCVAR attribute 9-4
DBDNAME attribute 3-8
DBNAME attribute 8-2
DB2GEN attribute 2-11
DB2MOD attribute 2-11
DCAIMPW A-3
DCAWORK A-3
DDSGEN attribute 2-12
DECIMALS attribute
on CONTITEM tag 6-9
on DEFITEM tag 6-6
on ITEM TAG 7-2
on RECDITEM tag 5-11
on VFIELD tag 9-10
DEFFOLD attribute 9-5
DEFITEM tag 6-5
DELETE process option 3-2
DESC attribute
on DEFITEM tag 6-6
on GROUP tag 4-1
on ITEM tag 7-2
on PROCESS tag 3-2
on RECDITEM tag 5-11
on VFIELD tag 9-10
DETECT attribute
on CATTR tag 9-8
on VATTR tag 9-18
DEVICE attribute 10-2
DEVICES attribute 9-2
DISPLAY process option 3-3
DLICALL tag 3-8

E

EAFTR tag 3-5
EAPPL tag 2-1
EBEFORE tag 3-4
ECFIELD tag 9-6
EDITMSG attribute
on MESSAGES tag 7-7
on VFIELD tag 9-16
EDITORDR attributes 9-11
EDITRTN attribute
on MAPEDITS (item) tag 7-4
on MAPEDITS (map) tag 9-13
EGROUP tag 4-1
EITEM tag 7-1
EJOINCON tag 5-8
EMAINPRC tag 2-5
EMAP tag 9-1
EMAPG tag 10-1
enhancements
batch generations options support xv
CSP/AD Programmable Workstation Feature xiii
external source format interactive support xiv
facility transfer command support xv
graphical representation for CSP/AD xiii
icons xiii

- enhancements (*continued*)
 - IMS/DC definition xiii
 - local data item support xiv
 - long names support xiv
 - new publications xv
 - overview facility xiii
 - SQL extensions xiv
 - tree structures xiii
 - VM/XA exploitation xiv
- 3.3 xiii
- ENTER attribute
 - on CATTR tag 9-8
 - on VATTR tag 9-18
- EPROCESS tag 3-1
- EPROL tag
 - with PROL tag 2-9
 - with RECORD tag 5-9
 - with TBLE tag 6-4
- EPSB tag 8-1
- ERECORD tag 5-1
- error processing A-4
- ERRRTN attribute 3-2
- ESQL tag 3-6
- ESTMTS tag 4-3
- ETBLE tag 6-1
- EVENSQLE attribute
 - on DEFITEM tag 6-6
 - on ITEM tag 7-2
 - on RECDITEM tag 5-12
- EVFIELD tag 9-10
- examples, syntax
 - See tag syntax, examples
- EXECBLD attribute 3-2
- EXECMODE attribute 2-12
- EXECUTE process option 3-3
- EXPORT command A-1
- export files A-2
- export of MSL members A-1
- external source format 1-1
- external source format data storage B-1
- external source format file A-2

F

- field text attribute 9-6
- FILELOC attribute 5-2
- FILENAME attribute
 - on GENFILE tag 2-16
 - on RECORD tag 5-2
- FILETYPE attribute 2-17
- FILL attribute
 - on CATTR tag 9-8
 - on VATTR tag 9-18
- FILLCHAR attribute
 - on MAPEDITS (item) tag 7-4
 - on MAPEDITS (map) tag 9-13
- FIRSTMAP attribute
 - on APPL tag 2-2

- FLDFOLD attribute
 - on MAPEDITS (item) tag 7-4
 - on MAPEDITS (map) tag 9-13
- floating area 10-2
- flow statement attribute 2-5
- FOLD attribute
 - on GENOPTS tag 2-12
 - on TBLE tag 6-1

G

- GENFILE tag 2-16
- GENGRP1 attribute 2-12
- GENGRP2 attribute 2-12
- GENOPTS tag
 - with APPL tag 2-10
 - with TBLE tag 6-3
- GENTABLE tag 2-14
- GET command A-1
- GROUP tag 4-1
- GRPNAME attribute
 - on MAP tag 9-2
 - on MAPG tag 10-1

H

- header record A-2
- HELPGRP attribute 2-2
- HELPKEY attribute
 - on APPL tag 2-2
 - on MAP tag 9-2
- HELPMAP attribute 9-3
- HEXEDIT attribute
 - on MAPEDITS (item) tag 7-4
 - on MAPEDITS (map) tag 9-13
- HILITE attribute
 - on CATTR tag 9-8
 - on VATTR tag 9-18
- HOSTTRAN attribute 2-19
- HOSTVAR attribute
 - on JOINCON tag 5-8
 - on SQL tag 3-6
- HTFERITM attribute 2-19
- HTFFILE tag 2-19
- HTFWKITM attribute 2-20

I

- IKEY attribute 8-3
- IMPLICIT attribute 2-2
- IMPORT command A-1
- import files A-2
- import of MSL members A-1
- import work file A-3
- INDEX attribute 9-11
- information library vi
 - Administering CSP/AD and CSP/AE on DPPX vi
 - Administering CSP/AD and CSP/AE on MVS vi

information library (*continued*)

Administering CSP/AD and CSP/AE on OS/2 and IBM DOS vi
Administering CSP/AD and CSP/AE on VM vi
Administering CSP/AD and CSP/AE on VSE vi
binders viii
Defining Applications on the S/370 vi
Developing Applications vii
External Source Format Reference viii
External Source Format Reference Summary viii
General Information Manual vi
Generating and Running IMS applications vii
LPS - CSP/AD for S/370 ix
LPS - CSP/AE for DPPX ix
LPS - CSP/AE for S/370 ix
LPS - CSP/370 Runtime Services ix
Messages, Codes, and Problem Determination vii
Planning viii
Reference vii
Reference Summary viii
initial value attribute 9-11
INPUTREQ attribute
on MAPEDITS (item) tag 7-5
on MAPEDITS (map) tag 9-13
INQUIRY process option 3-3
INTENSE attribute
on CATTR tag 9-8
on VATTR tag 9-18
internal format file A-3
Introduction 1-1
INVALMSG attribute
on MESSAGES tag 7-7
on VFIELD tag 9-16
ITEM tag 7-1

J

JOINCON tag 5-8
JUSTIFY attribute
on MAPEDITS (item) tag 7-5
on MAPEDITS (map) tag 9-14

K

KEEP attribute 2-14
KEY attribute
on RECDITEM tag 5-12
on RECORD tag 5-2

L

LABEL attribute 5-6
LEVEL attribute
on DEFITEM tag 6-6
on RECDITEM tag 5-12
LOADLIB attribute
on GENOPTS tag 2-12
on TBLE tag 6-3

M

MAINPRC tag 2-5
MAP tag 9-1
MAPEDITS tag
with ITEM tag 7-3
with VFIELD tag 9-12
MAPG tag 10-1
MAPGROUP attribute 2-2
MAPGRP1 attribute 2-12
MAPGRP2 attribute 2-12
MAPNAME attribute 9-3
MAPSIZE attribute 9-3
MDT attribute
on CATTR tag 9-8
on VATTR tag 9-18
message print file A-3
MESSAGES tag
with ITEM tag 7-7
with VFIELD tag 9-16
MININMSG attribute
on MESSAGES field 7-7
on VFIELD tag 9-16
MININPUT attribute
on MAPEDITS (item) tag 7-5
on MAPEDITS (map) tag 9-14
MIXCONS attribute 9-5
MIXVAR attribute 9-5
MODEL attribute 3-2
MSGFILE attribute 2-2

N

NAME attribute
on APPL tag 2-3
on CALLPARM tag 2-7
on CONTITEM tag 6-9
on DEFITEM tag 6-7
on GENTABLE tag 2-14
on GROUP tag 4-2
on ITEM tag 7-2
on MAINPRC tag 2-5
on PROCESS tag 3-2
on PSB tag 8-1
on RECDITEM tag 5-12
on RECORD tag 5-3
on TABREC tag 2-6
on TBLE tag 6-2
on VFIELD tag 9-11
NUMOCCUR attribute 5-3
NUMSEP attribute
on MAPEDITS (item) tag 7-5
on MAPEDITS (map) tag 9-14

O

OBJECT attribute 3-2

OCCURS attribute 5-13
OPTION attribute
 See process options
ORG attribute 5-3
OUTLINE attribute
 on CATTR tag 9-9
 on VATTR tag 9-19

P

PARENT attribute 8-3
PCB tag 8-2
PCBNO attribute 2-17
PFEQUATE attribute 2-3
PRESENT tag 9-4
PRINT attribute 2-12
process options
 ADD 3-2
 CLOSE 3-2
 CONVERSE 3-2
 DELETE 3-2
 DISPLAY 3-3
 EXECUTE 3-3
 INQUIRY 3-3
 REPLACE 3-3
 SCAN 3-3
 SCANBACK 3-3
 SETINQ 3-3
 SETUPD 3-3
 SQLEXEC 3-3
 UPDATE 3-3
PROCESS tag 3-1
PROL tag
 with EPROL tag 2-9
 with RECORD tag 5-9
 with TBLE tag 6-4
PROTECT attribute
 on CATTR tag 9-9
 on VATTR tag 9-19
PSB attribute
 on APPL tag 2-3
 on DLICALL tag 3-8
PSB tag 8-1
PUT command A-1

Q

QUAL tag 3-11

R

RANGE attribute
 on MAPEDITS (item) tag 7-5
 on MAPEDITS (map) tag 9-14
RANGEMSG attribute
 on MESSAGES tag 7-7
 on VFIELD tag 9-16

READONLY attribute 5-13
RECDITEM tag 5-10
RECORD tag 5-1
REDEFREC attribute 5-5
REFINE attribute
 on GROUP tag 4-2
 on PROCESS tag 3-3
RELOP attribute 3-12
REPLACE process option 3-3
report format C-1
REQMSG attribute
 on MESSAGES tag 7-8
 on VFIELD tag 9-16
ROW attribute
 on CFIELD tag 9-6
 on VFIELD tag 9-11
ROW tag 6-10

S

Sample Applications
sample printed messages C-1
SCAN process option 3-3
SCANBACK process option 3-3
SCANPAR attribute 3-8
SCANUPD attribute 3-8
SCOPE attribute
 on DEFITEM tag 6-7
 on RECDITEM tag 5-13
 on RECORD tag 5-5
 on TBLE tag 6-2
SEGFIELD attribute 3-12
SEGMENT attribute 8-4
SEGNAME attribute 3-10
SEGTRAN attribute 2-13
SENSE tag 8-3
SETINQ process option 3-3
SETUPD process option 3-3
SIGN attribute
 on MAPEDITS (item) tag 7-5
 on MAPEDITS (map) tag 9-14
SINGROW attribute 3-3
SIZE attribute 10-2
SOSIPOS attribute 9-3
SPACER attribute 9-5
SQL tag 3-6
SQLBLOCK attribute 2-13
SQLEXEC process option 3-3
SQLGEN attribute 2-13
SQLID attribute 2-13
SQLMOD attribute 2-13
SQLTABLE tag 5-6
SSA tag 3-9
STARTPOS attribute
 on AREA tag 10-2
 on MAP tag 9-3
STMTS tag 4-3

syntax

- See tag syntax, examples
- SYSNAME attribute 2-17
- SYSTEM attribute
 - on GENFILE tag 2-17
 - on TARGSYS tag 2-15

T

- TABLEGEN attribute 2-14
- TABLEID attribute 5-6
- TABPOS attribute 9-5
- TABREC tag 2-6
- TABTYPE attribute 6-2
- tag syntax
 - description 1-2-1-5
 - examples
 - application definition 2-21
 - data item definition 7-9
 - map definition 9-20
 - process definition 3-13
 - PSB definition 8-5
 - record definition 5-14
 - statement group definition 4-4
 - table definition 6-11

tags

- AFTER 3-5
- APPL 2-1
- AREA 10-2
- BEFORE 3-4
- CALLPARAM 2-7
- CATTR 9-7
- CFIELD 9-6
- CONTITEM 6-8
- DEFITEM 6-5
- DLICALL 3-8
- EAFTR 3-5
- EAPPL 2-1
- EBEFORE 3-4
- ECFIELD 9-6
- EGROUP 4-1
- EITEM 7-1
- EJOINCON 5-8
- EMAINPRC 2-5
- EMAP 9-1
- EMAPG 10-1
- EPROCESS 3-1
- EPROL
 - with PROL tag 2-9
 - with RECORD tag 5-9
 - with TBLE tag 6-4
- EPSB 8-1
- ERECORD 5-1
- ESQL 3-6
- ESTMTS 4-3
- ETBLE 6-1
- EVFIELD 9-10
- GENFILE 2-16

tags (continued)

- GENOPTS
 - with APPL tag 2-10
 - with TBLE tag 6-3
- GENTABLE 2-14
- GROUP 4-1
- HTFFILE 2-19
- ITEM 7-1
- JOINCON 5-8
- MAINPRC 2-5
- MAP 9-1
- MAPEDITS
 - with ITEM tag 7-3
 - with VFIELD tag 9-12
- MAPG 10-1
- MESSAGES
 - with ITEM tag 7-7
 - with VFIELD tag 9-16
- PCB 8-2
- PRESENT 9-4
- PROCESS 3-1
- PROL
 - with EPROL tag 2-9
 - with RECORD tag 5-9
 - with TBLE tag 6-4
- PSB 8-1
- QUAL 3-11
- RECDITEM 5-10
- RECORD 5-1
- ROW 6-10
- SENSEG 8-3
- SQL 3-6
- SQLTABLE 5-6
- SSA 3-9
- STMTS 4-3
- TABREC 2-6
- TARGSYS 2-15
- TBLE 6-1
- VATTR 9-17
- VFIELD 9-10
- TARGSYS 2-15
- TBLE tag 6-1
- TBLENAME attribute 5-6
- TIME attribute
 - on APPL tag 2-3
 - on GROUP tag 4-2
 - on ITEM tag 7-2
 - on MAP tag 9-3
 - on MAPG tag 10-1
 - on PROCESS tag 3-3
 - on PSB tag 8-1
 - on RECORD tag 5-5
 - on TBLE tag 6-2
- TYPE attribute
 - on APPL tag 2-3
 - on CALLPARAM tag 2-8
 - on CFIELD tag 9-6
 - on CONTITEM tag 6-9

TYPE attribute (*continued*)

- on DEFITEM tag 6-7
 - on ITEM tag 7-2
 - on PCB tag 8-2
 - on RECDITEM tag 5-13
 - on TABREC tag 2-6
 - on VFIELD tag 9-11
- TYPEUSE attribute 6-3

U

- UPDATE process option 3-3
- UPDPROC attribute 3-3

V

- VALIDLOC attribute 2-13
- VALIDSQL attribute 2-13
- VARFOLD attribute 9-5
- VARIABLE attribute 9-5
- Variable Field Definition, Format Example 1-5
- VARLENTH attribute 5-5
- VATTR tag 9-17
- Version 3 Release 3 enhancements xiii
- VFIELD tag 9-10

W

- WORKSTOR attribute 2-4

Z

- ZEROEDIT attribute
 - on MAPEDITS (item) tag 7-6
 - on MAPEDITS (map) tag 9-14

This manual is part of a library that serves as a reference source for systems 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 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: Copies of IBM publications are not stocked at the location to which this form is addressed. Please 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.

Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. 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 front cover or title page.)

Reader's Comment Form

Fold and tape

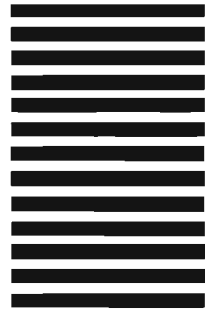
Please Do Not Staple

Fold and tape



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.



POSTAGE WILL BE PAID BY ADDRESSEE:
INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT T45 INFORMATION DEVELOPMENT
PO BOX 60000
CARY, NC 27512-9968



Fold and tape

Please Do Not Staple

Fold and tape



IBM System Product/Applicat
Development
External Source Format Reference Release Version 3 Release 3 Printed in USA SH20-6433-1





Program Number
5668 - 813

File Number
S370/4300/8100 - 50

SH20-6433-1

