

WebFOCUS

Describing Data
With WebFOCUS Language
Version 4 Release 3

Cactus, EDA/SQL, FIDEL, FOCCALC, FOCUS, FOCUS Fusion, FOCUS Vision, Hospital-Trac, Information Builders, the Information Builders logo, Parlay, PC/FOCUS, SmartMart, SmartMode, SNAPpack, TableTalk, WALDO, Web390, WebFOCUS and WorldMART are registered trademarks and EDA, iWay, and iWay Software are trademarks of Information Builders, Inc.

Acrobat and Adobe are registered trademarks of Adobe Systems Incorporated.

Allaire and JRun are trademarks of Allaire Corporation.

NOMAD is a registered trademark of Aonix.

UniVerse is a registered trademark of Ardent Software, Inc.

AvantGo is a trademark of AvantGo, Inc.

WebLogic is a registered trademark of BEA Systems, Inc.

SUPRA and TOTAL are registered trademarks of Cincom Systems, Inc.

Impromptu is a registered trademark of Cognos.

Alpha, DEC, DECnet, and NonStop are registered trademarks and Tru64, OpenVMS, and VMS are trademarks of Compaq Computer Corporation.

CA-ACF2, CA-Datcom, CA-IDMS, CA-Top Secret, and Ingres are registered trademarks of Computer Associates International, Inc.

MODEL 204 and M204 are registered trademarks of Computer Corporation of America.

Paradox is a registered trademark of Corel Corporation.

StorHouse is a registered trademark of FileTek, Inc.

HP MPE/iX is a registered trademark of Hewlett Packard Corporation.

Informix is a registered trademark of Informix Software, Inc.

Intel is a registered trademark of Intel Corporation.

ACF/VTAM, AIX, AS/400, CICS, DB2, DRDA, Distributed Relational Database Architecture, IBM, MQSeries, MVS/ESA, OS/2, OS/390, OS/400, RACF, RS/6000, S/390, VM/ESA, VSE/ESA and VTAM are registered trademarks and DB2/2, Hiperspace, IMS, MVS, QMF, SQL/DS, WebSphere, z/OS and z/VM are trademarks of International Business Machines Corporation.

INTERSOLVE and Q+E are registered trademarks of INTERSOLVE.

Orbit is a registered trademark of Iona Technologies Inc.

Approach and DataLens are registered trademarks of Lotus Development Corporation.

ObjectView is a trademark of Matesys Corporation.

ActiveX, FrontPage, Microsoft, MS-DOS, PowerPoint, Visual Basic, Visual C++, Visual FoxPro, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

Motorola is a trademark of Motorola, Inc.

Teradata is a registered trademark of NCR International, Inc.

Netscape, Netscape FastTrack Server, and Netscape Navigator are registered trademarks of Netscape Communications Corporation.

ServletExec is a trademark of New Atlanta Communications, LLC.

Nextel is a registered trademark of Nextel Communications.

NetWare and Novell are registered trademarks of Novell, Inc.

CORBA is a trademark of Object Management Group, Inc.

Oracle is a registered trademark and Rdb is a trademark of Oracle Corporation.

Palm is a trademark and Palm OS is a registered trademark of Palm Inc.

INFOAccess is a trademark of Pioneer Systems, Inc.

Plumtree is a trademark of Plumtree Software, Inc.

Progress is a registered trademark of Progress Software Corporation.

BlackBerry is a trademark and RIM is a registered trademark of Research In Motion Limited.

Red Brick Warehouse is a trademark of Red Brick Systems.

SAP and SAP R/3 are registered trademarks and SAP Business Information Warehouse and SAP BW are trademarks of SAP AG.

Silverstream is a trademark of Silverstream Software.

ADABAS is a registered trademark of Software A.G.

CONNECT:Direct is a trademark of Sterling Commerce.

Java and all Java-based marks, NetDynamics, Solaris, SunOS, and iPlanet are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

PowerBuilder, Powersoft, and Sybase are registered trademarks and SQL Server is a trademark of Sybase, Inc.

Unicode is a trademark of Unicode, Inc.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2001, by Information Builders, Inc. All rights reserved. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Printed in the U.S.A.

Preface

This documentation describes how to create the metadata for the data sources that your WebFOCUS applications will access. It is intended for database administrators, application developers, or other information technology professionals who will create the metadata used by WebFOCUS to access corporate data. This documentation is part of the WebFOCUS documentation set.

How This Manual Is Organized

This manual is organized as follows:

Chapter/Appendix		Description
1	<i>Understanding a Data Source Description</i>	Introduces data source descriptions and explains how to use them.
2	<i>Identifying a Data Source</i>	Documents how to describe general aspects of a data source.
3	<i>Describing a Group of Fields</i>	Documents how to describe groups of related fields or segments of a data source.
4	<i>Describing an Individual Field</i>	Documents how to describe specific field level information of a data source.
5	<i>Describing a Sequential, VSAM, or ISAM Data Source</i>	Provides supplementary information specific to sequential, VSAM, and ISAM data sources.
6	<i>Describing a FOCUS Data Source</i>	Provides information specific to FOCUS data sources.
7	<i>Defining a Join in a Master File</i>	Describes how to create a new relationship between any two segments that have at least one field in common by joining them.
8	<i>Checking and Changing a Master File: CHECK</i>	Describes how to use the CHECK command to validate a data source description.
9	<i>Providing Data Source Security: DBA</i>	Describes how you can use DBA security features to provide security for any FOCUS data source.
10	<i>Creating and Rebuilding a Data Source</i>	Describes utilities to create new FOCUS data sources and to refresh existing data sources after the structure has changed.
A	<i>Master Files and Diagrams</i>	Contains Master Files and diagrams of sample data sources used in the documentation examples.
B	<i>Error Messages</i>	Describes how to obtain information about error messages.

Documentation Conventions

The following conventions apply throughout this manual:

Convention	Description
<code>THIS TYPEFACE</code> or <code>this typeface</code>	Denotes syntax that you must enter exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
<i>this typeface</i>	Represents a placeholder (or variable) in a text paragraph, indicates a cross-reference, or emphasizes an important term.
this typeface	Highlights file names and commands (in a text paragraph) that must be lowercase.
this typeface	Indicates buttons, menu items, and dialog box options you can click or select.
Key + Key	Indicates keys that must be pressed simultaneously.
{ }	Indicates two choices from which you must choose one. You type one of these choices, not the braces.
[]	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the information within the brackets, not the brackets.
	Separates two mutually exclusive choices in a syntax line. You type one of these choices, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameters, not the ellipsis points (...).
.	Indicates that there are (or could be) intervening or additional commands.

Related Publications

See the *Information Builders Technical Publications Catalog* for the most up-to-date listing and prices of technical publications, plus ordering information. To obtain a catalog, contact the Publications Order Department at (800) 969-4636.

You can also visit our World Wide Web site, <http://www.informationbuilders.com>, to view a current listing of our publications and to place an order.

Customer Support

Do you have questions about WebFOCUS?

Call Information Builders Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your WebFOCUS questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code number (*xxxx.xx*) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, <http://www.informationbuilders.com>. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of www.informationbuilders.com also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Information You Should Have

To help our consultants answer your questions most effectively, be ready to provide the following information when you call:

- Your six-digit site code number (*xxxx.xx*).
- Your WebFOCUS configuration:
 - The front-end you are using, including vendor and release.
 - The communications protocol (for example, TCP/IP or HLLAPI), including vendor and release.
 - The software release.
 - The server you are accessing, including release (for example, 4.3.6).
- The stored procedure (preferably with line numbers) or commands being used in server access.
- The name of the Master File and Access File.

- The exact nature of the problem:
 - Are the results or the format incorrect? Are the text or calculations missing or misplaced?
 - The error message and return code, if applicable.
 - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, WebFOCUS, your security system, communications protocol, or front-end software changed?
- Is this problem reproducible? If so, how?
- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing the code to access a single data source?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

In an effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Document Enhancement Request Form on our Web site, <http://www.informationbuilders.com>.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our World Wide Web site (<http://www.informationbuilders.com>) or call (800) 969-INFO to speak to an Education Representative.

Contents

- 1 Understanding a Data Source Description1-1**
 - A Note About Data Source Terminology 1-2
 - What Is a Data Source Description?..... 1-2
 - How an Application Uses a Data Source Description 1-3
 - What Does a Master File Describe? 1-3
 - Identifying a Data Source..... 1-3
 - Identifying and Relating a Group of Fields 1-3
 - Describing a Field 1-4
 - Creating a Data Source Description 1-4
 - Creating a Master File and Access File Using an Editor..... 1-4
 - What Is in a Master File?..... 1-5
 - Improving Readability..... 1-6
 - Adding a Comment 1-7
 - Editing and Validating a Master File..... 1-7

- 2 Identifying a Data Source2-1**
 - Specifying a Data Source Name: FILENAME..... 2-2
 - Identifying a Data Source Type: SUFFIX 2-2
 - Providing Descriptive Information for a Data Source: REMARKS 2-6
 - Specifying a Physical File Name: DATASET..... 2-6
 - DATASET Behavior in a FOCUS Data Source 2-7
 - DATASET Behavior in a Fixed-Format Sequential Data Source 2-10
 - DATASET Behavior in a VSAM Data Source 2-12

- 3 Describing a Group of Fields3-1**
 - Defining a Single Group of Fields..... 3-2
 - Understanding a Segment..... 3-2
 - Understanding a Segment Instance 3-3
 - Understanding a Segment Chain 3-3
 - Identifying a Key Field..... 3-4
 - Identifying a Segment: SEGNAME 3-4
 - Identifying a Logical View: Redefining a Segment 3-5
 - Relating Multiple Groups of Fields 3-7
 - Facilities for Specifying a Segment Relationship..... 3-7
 - Identifying a Parent Segment: PARENT 3-8
 - Identifying the Type of Relationship: SEGTYPE 3-8

- Logical Dependence: The Parent-Child Relationship3-9
 - A Simple Parent-Child Relationship3-9
 - A Parent-Child Relationship With Multiple Segments3-10
 - Types of Parent and Child Segments.....3-10
 - Understanding a Root Segment.....3-11
 - Understanding a Descendant Segment3-11
 - Understanding an Ancestral Segment3-12
- Logical Independence: Multiple Paths3-12
 - Understanding a Single Path3-12
 - Understanding Multiple Paths3-13
 - Understanding Logical Independence3-14
- Cardinal Relationships Between Segments3-14
- The One-to-One Relationship.....3-15
 - Understanding a One-to-One Relationship.....3-16
 - Where to Use a One-to-One Relationship3-17
 - Implementing a One-to-One Relationship in a Relational Data Source.....3-17
 - Implementing a One-to-One Relationship in a Sequential Data Source.....3-17
 - Implementing a One-to-One Relationship in a FOCUS Data Source.....3-17
- The One-to-Many Relationship.....3-18
 - Understanding a One-to-Many Relationship.....3-19
 - Implementing a One-to-Many Relationship in a Relational Data Source3-19
 - Implementing a One-to-Many Relationship in a VSAM or Sequential Data Source3-20
 - Implementing a One-to-Many Relationship in a FOCUS Data Source.....3-20
- The Many-to-Many Relationship3-20
 - Implementing a Many-to-Many Relationship Directly3-21
 - Implementing a Many-to-Many Relationship Indirectly3-22
- Recursive Relationships3-24
- Relating Segments From Different Types of Data Sources.....3-26
- Rotating a Data Source: An Alternate View.....3-28

4 Describing an Individual Field	4-1
Field Characteristics	4-2
The Field's Name: FIELDNAME	4-3
Using a Long and Qualified Field Name	4-4
Using a Duplicate Field Name	4-6
Rules for Evaluating a Qualified Field Name	4-7
The Field's Synonym: ALIAS	4-10
Implementing a Field Synonym	4-11
The Displayed Data Type: USAGE	4-11
Data Type Formats	4-13
Integer Format	4-13
Floating-Point Double-Precision Format	4-14
Floating-Point Single-Precision Format	4-15
Packed-Decimal Format	4-16
Numeric Display Options	4-17
Rounding	4-19
Alphanumeric Format	4-21
Date Formats	4-21
Date Display Options	4-22
Controlling the Date Separator	4-26
Date Translation	4-26
Using a Date Field	4-27
Numeric Date Literals	4-28
Date Fields in Arithmetic Expressions	4-29
Converting a Date Field	4-29
How a Date Field Is Represented Internally	4-30
Displaying a Non-Standard Date Format	4-31
Date Format Support	4-31
Alphanumeric and Numeric Formats With Date Display Options	4-32
Date-Time Formats	4-33
Describing a Date-Time Field	4-33
Specifying a Date-Time Value	4-38
Text Field Format	4-39
The Stored Data Type: ACTUAL	4-39
Null or MISSING Values: MISSING	4-40
Using a Missing Value	4-41
Defining a Dimension: WITHIN	4-42
Validating Data: ACCEPT	4-48
Specifying Acceptable Values for a Dimension	4-49
Alternative Report Column Titles: TITLE	4-51
Documenting the Field: DESCRIPTION	4-52

- Describing a Virtual Field: DEFINE 4-53
 - Using a Virtual Field 4-54
- 5 Describing a Sequential, VSAM, or ISAM Data Source 5-1**
 - Sequential Data Source Formats..... 5-2
 - What Is a Fixed-Format Data Source? 5-2
 - What Is a Free-Format Data Source? 5-4
 - Rules for Maintaining a Free-Format Data Source..... 5-4
 - Standard Master File Attributes for a Sequential Data Source 5-5
 - Standard Master File Attributes for a VSAM or ISAM Data Source 5-6
 - Group Fields..... 5-7
 - Describing a Multiply Occurring Field in a Free-Format Data Source 5-8
 - Describing A Multiply Occurring Field in a Fixed-Format, VSAM, or ISAM Data Source..... 5-9
 - The OCCURS Attribute 5-10
 - A Parallel Set of Repeating Fields..... 5-13
 - A Nested Set of Repeating Fields..... 5-14
 - The POSITION Attribute 5-16
 - Specifying the ORDER Field 5-17
 - Redefining a Field in a Non-FOCUS Data Source 5-18
 - Extra-Large Record Length Support 5-19
 - Describing Multiple Record Types..... 5-20
 - Describing a RECTYPE Field..... 5-22
 - Describing Positionally Related Records 5-23
 - Order of Records in the Data Source..... 5-24
 - Describing Unrelated Records..... 5-26
 - Using a Generalized Record Type..... 5-30
 - Using an ALIAS in a Report Request 5-32
 - Combining Multiply Occurring Fields and Multiple Record Types..... 5-33
 - Describing a Multiply Occurring Field and Multiple Record Types..... 5-34
 - A VSAM Repeating Group With RECTYPES 5-36
 - Describing a Repeating Group Using MAPFIELD 5-37
 - VSAM Data and Index Buffers 5-39
 - A VSAM Alternate Index..... 5-40

6	Describing a FOCUS Data Source	6-1
	Designing a FOCUS Data Source	6-2
	Data Relationships.....	6-2
	Join Considerations	6-3
	General Efficiency Considerations.....	6-3
	Changing a FOCUS Data Source	6-4
	Describing a Single Segment.....	6-5
	Maximum Number of Segments	6-5
	Describing Keys, Sort Order, and Segment Relationships: SEGTYPE.....	6-5
	Describing a Key Field.....	6-7
	Describing Sort Order	6-7
	Understanding Sort Order	6-8
	Describing Segment Relationships.....	6-8
	Storing a Segment in a Different Location: LOCATION	6-9
	Separating Large Text Fields	6-11
	Limits on the Number of LOCATION Files	6-12
	The ACCEPT Attribute	6-12
	The INDEX Attribute	6-13
	Joins and the INDEX Attribute	6-15
	FORMAT and MISSING: Internal Storage Requirements.....	6-16
7	Defining a Join in a Master File	7-1
	Join Types.....	7-2
	Static Joins Defined in the Master File: SEGTYPE = KU and KM	7-3
	Describing a Unique Join: SEGTYPE = KU.....	7-3
	Using a Unique Join for Decoding	7-6
	Describing a Non-Unique Join: SEGTYPE = KM	7-7
	Using Cross-Referenced Descendant Segments: SEGTYPE = KL and KLU	7-9
	Hierarchy of Linked Segments.....	7-13
	Dynamic Joins Defined in the Master File: SEGTYPE = DKU and DKM.....	7-14
	Comparing Static and Dynamic Joins.....	7-15
	Joining to One Cross-Referenced Segment From Several Host Segments.....	7-17
	Joining From Several Segments in One Host Data Source	7-17
	Joining From Several Segments in Several Host Data Sources: Multiple Parents	7-19
	Recursive Reuse of a Segment	7-21

8	Checking and Changing a Master File: CHECK	8-1
	CHECK Command Display	8-3
	Determining Common Errors	8-4
	The PICTURE Option	8-5
	The HOLD Option.....	8-8
	Specifying an Alternate File Name With the HOLD Option.....	8-10
	TITLE, HELPMESSAGE, and TAG Attributes	8-10
	Virtual Fields in the Master File.....	8-10
9	Providing Data Source Security: DBA	9-1
	Introduction to Data Source Security	9-2
	Implementing Data Source Security.....	9-3
	Identifying the DBA: The DBA Attribute.....	9-5
	Including the DBA Attribute in a HOLD File.....	9-6
	Identifying Users With Access Rights: The USER Attribute.....	9-6
	Establishing User Identity	9-7
	Specifying an Access Type: The ACCESS Attribute.....	9-9
	Types of Access	9-10
	Limiting Data Source Access: The RESTRICT Attribute.....	9-12
	Restricting Access to a Field or a Segment.....	9-13
	Restricting Access to a Value.....	9-15
	Placing Security Information in a Central Master File.....	9-17
	File Naming Requirements for DBAFILE	9-19
	Connection to an Existing DBA System With DBAFILE.....	9-19
	Combining Applications With DBAFILE.....	9-20
	Summary of Security Attributes.....	9-21
	Hiding Restriction Rules: The ENCRYPT Command	9-22
	Encrypting Data.....	9-22
	Performance Considerations for Encrypted Data	9-23
	Restricting an Existing FOCUS Data Source	9-23
	Setting a Password Externally	9-24
	WebFOCUS FOCEXEC Security	9-24
	Encrypting and Decrypting a FOCEXEC.....	9-24
10	Creating and Rebuilding a Data Source.....	10-1
	Creating a New Data Source: The CREATE Command.....	10-2
	Rebuilding a Data Source: The REBUILD Command.....	10-4
	Controlling the Frequency of REBUILD Messages.....	10-6
	Optimizing File Size: The REBUILD Subcommand	10-7
	Changing Data Source Structure: The REORG Subcommand.....	10-9
	Indexing Fields: The INDEX Subcommand.....	10-13

Creating an External Index: The EXTERNAL INDEX Subcommand.....	10-15
Concatenating Index Databases.....	10-18
Positioning Indexed Fields.....	10-18
Activating an External Index.....	10-18
Checking Data Source Integrity: The CHECK Subcommand.....	10-21
Confirming Structural Integrity Using ? FILE and TABLEF.....	10-23
Changing the Data Source Creation Date and Time: The TIMESTAMP Subcommand.....	10-25
Converting Legacy Dates: The DATE NEW Subcommand.....	10-27
How DATE NEW Converts Legacy Dates.....	10-28
What DATE NEW Does Not Convert.....	10-30
Action Taken on a Date Field During REBUILD/DATE NEW.....	10-31
Migrating to a Fusion Data Source: The MIGRATE Subcommand.....	10-32
Creating a Fusion Multi-Dimensional Index: The MDINDEX Subcommand.....	10-32
A Master Files and Diagrams.....	A-1
EMPLOYEE Data Source.....	A-2
EMPLOYEE Master File.....	A-3
EMPLOYEE Structure Diagram.....	A-4
JOBFILE Data Source.....	A-5
JOBFILE Master File.....	A-5
JOBFILE Structure Diagram.....	A-6
EDUCFILE Data Source.....	A-7
EDUCFILE Master File.....	A-7
EDUCFILE Structure Diagram.....	A-7
SALES Data Source.....	A-8
SALES Master File.....	A-8
SALES Structure Diagram.....	A-9
CAR Data Source.....	A-10
CAR Master File.....	A-11
CAR Structure Diagram.....	A-12
LEDGER Data Source.....	A-13
LEDGER Master File.....	A-13
LEDGER Structure Diagram.....	A-13
FINANCE Data Source.....	A-14
FINANCE Master File.....	A-14
FINANCE Structure Diagram.....	A-14
REGION Data Source.....	A-15
REGION Master File.....	A-15
REGION Structure Diagram.....	A-15

COURSE Data Source..... A-16
 COURSE Master File..... A-16
 COURSE Structure Diagram..... A-16
EMPDATA Data Source A-17
 EMPDATA Master File A-17
 EMPDATA Structure Diagram A-17
TRAINING Data Source A-18
 TRAINING Master File A-18
 TRAINING Structure Diagram..... A-18
VideoTrk, MOVIES, and ITEMS Data Sources..... A-19
 VideoTrk Master File..... A-19
 VideoTrk Structure Diagram..... A-20
 MOVIES Master File A-21
 MOVIES Structure Diagram A-21
 ITEMS Master File..... A-22
 ITEMS Structure Diagram A-22
VIDEOTR2 Data Source A-23
 VIDEOTR2 Master Files..... A-23
 VIDEOTR2 Structure Diagram..... A-24
Gotham Grinds Data Sources A-25
 GGDEMOG Master File A-25
 GGDEMOG Structure Diagram A-26
 GGORDER Master File A-26
 GGORDER Structure Diagram A-27
 GGPRODS Master File A-27
 GGPRODS Structure Diagram..... A-28
 GGSALES Master File..... A-28
 GGSALES Structure Diagram A-29
 GGSTORES Master File A-29
 GGSTORES Structure Diagram..... A-29
Century Corp Data Sources A-30
 CENTCOMP Data Source..... A-31
 CENTCOMP Structure Diagram..... A-31
 CENTFIN Data Source A-32
 CENTFIN Structure Diagram A-32
 CENTHR Data Source A-33
 CENTHR Structure Diagram A-35
 CENTINV Data Source..... A-35
 CENTINV Structure Diagram..... A-36
 CENTORD Data Source A-36
 CENTORD Structure Diagram..... A-37
 CENTQA Data Source A-38
 CENTQA Structure Diagram A-39

B	Error Messages	B-1
	Displaying Messages	B-2
Index		I-1

CHAPTER 1

Understanding a Data Source Description

Topics:

- A Note About Data Source Terminology
- What Is a Data Source Description?
- How an Application Uses a Data Source Description
- What Does a Master File Describe?
- Creating a Data Source Description
- What Is in a Master File?

Information Builders products provide a flexible data description language, which you can use with many types of data sources, including:

- Relational, such as DB2, Oracle, Sybase, and Teradata.
- Hierarchical, such as IMS and FOCUS.
- Network, such as CA-IDMS.
- Indexed, such as ISAM and VSAM.
- Sequential, both fixed-format and free-format.
- Multi-dimensional, such as Fusion.

You can also use the data description language and related facilities to:

- Join different types of data sources to create a temporary structure from which your request can read or write.
- Define a subset of fields or columns to be available to users.
- Logically rearrange a data source to access the data in a different order.

A Note About Data Source Terminology

Different types of data sources make use of similar concepts, but refer to them differently. For example, the smallest meaningful element of data is called a *field* by many hierarchical database management systems and indexed data access methods, but called a *column* by relational database management systems.

There are other cases in which a common concept is identified by a number of different terms. For simplicity, we use a single set of standardized terms. For example, we usually refer to the smallest meaningful element of data as a *field*, regardless of the type of data source. However, when required for clarity, we use the term specific to a given data source. Each time we introduce a new standard term, we define it and compare it to equivalent terms used with different types of data sources.

What Is a Data Source Description?

When your application accesses a data source, it needs to know how to interpret the data that it finds. Your application needs to know about:

- The overall structure of the data. For example, is the data relational, hierarchical, or sequential? Depending upon the structure, how is it arranged or indexed?
- The specific data elements. For example, what fields are stored in the data source, and what is the data type of each field—character, date, integer, or some other type?

To obtain the necessary information, your application reads a data source description. The primary component of a data source description is called a Master File. A Master File describes the structure of a data source and its fields. For example, it includes information such as field names and data types.

For some data sources, an Access File supplements a Master File. An Access File includes additional information that completes the description of the data source. For example, it includes the full data source name and location. You need one Master File—and, for some data sources, one Access File—to describe a data source.

How an Application Uses a Data Source Description

Master Files and Access Files are stored separately, apart from the associated data source. Your application uses a data source's Master File (and if required, the corresponding Access File) to interpret the data source in the following way:

1. Identifies, locates, and reads the Master File for the data source named in a request.
If the Master File is already in memory, your application uses the memory image and then proceeds to Step 4. If the Master File is not in memory, the application locates the Master File on a storage device and loads it into memory, replacing any existing Master File in memory. If your Master File references other data sources as cross-referenced segments, or if a JOIN command is in effect for this file, the cross-referenced Master Files are also read into memory.
2. Reads the security rules if Information Builders data source security (DBA) has been specified for the data source and ensures that user access is based on any DBA security specified.
3. Locates and reads the Access File for the data source named in the request, if that data source requires an Access File.
4. Locates and reads the data source.
The data source contents are interpreted based on the information in the Master File and, if applicable, the Access File.

What Does a Master File Describe?

A Master File enables you to:

- Identify the name and type of a data source.
- Identify and relate groups of fields.
- Describe individual fields.

Identifying a Data Source

In order to interpret data, your application needs to know the name you are using to identify the data source and what type of data source it is. For example, is it a DB2 data source, an Oracle data source, or a FOCUS data source?

For more information see Chapter 2, *Identifying a Data Source*.

Identifying and Relating a Group of Fields

A Master File identifies and relates groups of fields that have a one-to-one correspondence with each other—in Master File terms, a segment; in relational terms, a table.

You can join data sources of the same type (using a Master File or a JOIN command) and data sources of different types (using a JOIN command). For example, you can join two DB2 data sources to a FOCUS data source, and then to a VSAM data source.

For more information, about defining and relating groups of fields, see Chapter 3, *Describing a Group of Fields*.

Describing a Field

Every field has several characteristics that you must describe in a Master File, such as type of data and length or scale. A Master File can also indicate optional field characteristics. For example, a Master File can specify if the field can have a null value, and can provide descriptive information for the field.

A Master File usually describes all of the fields in a data source. In some cases, however, you can create a logical view of the data source in which only a subset of the fields is available, and then describe only those fields in your Master File.

For more information, see Chapter 4, *Describing an Individual Field*.

Creating a Data Source Description

You can create a Master File and Access File for a data source in several ways. If the data source:

- **Has an existing description**—such as a native schema or catalog, or a COBOL File Description—you can use a tool to automatically generate the Master File and Access File from the existing description. For example, if you want to use WebFOCUS to read a DB2 data source, you can use the Synonym Wizard to generate a Master File and Access File from the DB2 catalog.
- **Does not have an existing description**, you can create a Master File and (if required) an Access File by coding them using Information Builders' data source description language, and specify their attributes using any text editor.

For more information about coding or specifying a Master File and Access File, see *Creating a Master File and Access File Using an Editor* on page 1-4.

Creating a Master File and Access File Using an Editor

You can create a Master File and an Access File by:

- **Coding** them using a text editor. You can do this in all Information Builders products. The information that you need about Master File syntax is contained in this documentation. For information about Access File syntax, see your data adapter documentation. See your documentation for more information about the Text Editor.
After editing a Master File, issue the CHECK FILE command to validate the new Master File and to refresh your session's image of it.
- **Specifying** their attributes using the Master File Editor. The Master File Editor is available in the WebFOCUS (Windows version) and WebFOCUS Maintain.

What Is in a Master File?

A Master File describes a data source using a series of declarations:

- A data source declaration.
- A segment declaration for each segment within the data source.
- A field declaration for each field within a segment.

The specifications for an Access File are similar, although the details vary by type of data source. The appropriate documentation for your data adapter indicates whether you require an Access File and, if so, what the Access File attributes are.

Syntax

How to Specify a Declaration

Each declaration specifies a series of attributes in the form

```
attribute = value, attribute = value, ... , $
```

where:

attribute

Is a Master File keyword that identifies a file, segment, or field property. You can specify any Master File attribute by its full name, its alias, or its shortest unique truncation. For example, you can use the full attribute FILENAME or the shorter form FILE.

value

Is the value of the attribute.

A comma follows each attribute assignment, and each field declaration ends with a dollar sign (\$). Commas and dollar signs are optional at the end of data source and segment declarations.

Each declaration should begin on a new line. You can extend a declaration across as many lines as you wish. For a given declaration you can put each attribute assignment on a separate line, combine several attributes on each line, or include the entire declaration on a single line. Each line can be a maximum of 80 characters long.

For more information on data source declarations, see Chapter 2, *Identifying a Data Source*. For more information on segment declarations, see Chapter 3, *Describing a Group of Fields*. For more information on field declarations, see Chapter 4, *Describing an Individual Field*.

Note: In a Master File, the attribute name must be in English; the attribute value can be in any supported national language.

Improving Readability

You can begin each attribute assignment in any position that you wish. You can include blank spaces between the elements in a declaration. This makes it easy for you to indent segment or field declarations to make the Master File easier to read. To position text, use blank spaces, not the Tab character.

You can also include blank lines to separate declarations from each other. Blank spaces and lines are not required and are ignored by the application.

Example

Improving Readability With Blank Spaces and Blank Lines

The following declarations show how to improve readability by adding blank spaces and blank lines within and between declarations:

- `SEGNAME=EMPINFO, SEGTYPE=S1 , $
FIELDNAME=EMP_ID, ALIAS=EID, USAGE=A9 , $`
- `SEGNAME=EMPINFO, SEGTYPE=S1 , $
FIELDNAME = EMP_ID, ALIAS = EID, USAGE = A9 , $`
- `SEGNAME=EMPINFO, SEGTYPE=S1 , $
FIELDNAME = EMP_ID, ALIAS = EID, USAGE = A9 , $`

Example

Improving Readability by Extending a Declaration Across Lines

The following example extends a field declaration across several lines:

```
FIELDNAME = MEMBERSHIP, ALIAS = BELONGS, USAGE = A1, MISSING = ON,  
DESCRIPTION = This field indicates the applicant's membership status,  
ACCEPT = Y OR N, FIELDTYPE = I,  
HELPMESSAGE = 'Please enter Y for Yes or N for No' , $
```

Adding a Comment

You can add comments to any declaration by:

- Typing a comment in a declaration line after the terminating dollar sign.
- Creating an entire comment line by placing a dollar sign at the beginning of the line.

Adding a comment line terminates the previous declaration if it has not already been terminated. Everything on a line following the dollar sign is ignored.

Comments placed after a dollar sign are useful only for those who view the Master File source code. They are not displayed in graphical tools such as the Database Description Editor, Report Assistant, and Graph Assistant. For information about providing descriptions for display in graphical tools using the REMARKS or DESCRIPTION attribute, see Chapter 2, *Identifying a Data Source*, and Chapter 4, *Describing an Individual Field*.

Example

Adding a Comment in a Master File

The following example contains two comments. The first comment follows the dollar sign on the data source declaration. The second comment is on a line by itself after the data source declaration.

```
FILENAME = EMPLOYEE, SUFFIX = FOC , $ This is the personnel data source.  
$ This data source tracks employee salaries and raises.  
SEGNAME = EMPINFO, SEGTYPE = S1 , $
```

Editing and Validating a Master File

After you manually create or edit a Master File, you should issue the CHECK FILE command to validate it. CHECK FILE reads the new or revised Master File into memory and highlights any errors in your Master File so that you can correct them before reading the data source.

The CHECK FILE PICTURE command displays a diagram illustrating the structure of a data source. You can also use this command to view information in the Master File, such as names of segments and fields, and the order in which information is retrieved from the data source when you run a request against it.

For more information, see Chapter 8, *Checking and Changing a Master File: CHECK*.

CHAPTER 2

Identifying a Data Source

Topics:

- Specifying a Data Source Name: FILENAME
- Identifying a Data Source Type: SUFFIX
- Providing Descriptive Information for a Data Source: REMARKS
- Specifying a Physical File Name: DATASET

In order to interpret data, your application needs to know the name you are using to identify the data source and what type of data source it is. For example, is it a DB2 data source, a Teradata data source, or a FOCUS data source?

In a Master File, you identify the name and the type of data source in a data source declaration. A data source declaration can include the following attributes:

- FILENAME, which identifies the name of the data source.
- SUFFIX, which identifies the type of data source.
- REMARKS, which allows you to provide descriptive information about the data source for display in graphical tools such as Report Assistant and Graph Assistant.
- ACCESSFILE, which identifies the name of the optional Access File for a FOCUS data source. See Chapter 6, *Describing a FOCUS Data Source*.
- DATASET, which identifies the physical file name if your data source has a non-standard name.

You can optionally specify a sliding date window that assigns a century value to dates stored with two-digit years, using these data source attributes:

- FDEFCENT, which identifies the century.
- FYRTHRESH, which identifies the year.

For more information on the sliding window technique, see the *Developing Reporting Applications* manual.

Specifying a Data Source Name: FILENAME

The FILENAME attribute specifies the name of the data source described by the Master File. This is the first attribute specified in a Master File. You can abbreviate the FILENAME attribute to FILE.

Syntax

How to Specify a Data Source Name

```
FILE[NAME] = data_source_name
```

where:

data_source_name

Is the name of the data source that the Master File describes. The name can be a maximum of eight characters.

The file name must start with a letter, and the remaining characters can be any combination of letters, numbers, and underscores (_).

Example

Specifying a Data Source Name

The following example specifies the data source name EMPLOYEE:

```
FILENAME = EMPLOYEE
```

Identifying a Data Source Type: SUFFIX

The SUFFIX attribute identifies the type of data source you are using—for example, a DB2 data source or a FOCUS data source. Based on the value of SUFFIX, the appropriate data adapter is used to access the data source.

The SUFFIX attribute is required for most types of data sources. It is optional for a fixed-format sequential data source. However, if you refer to a fixed-format sequential data source in a JOIN command, then the SUFFIX attribute must be declared in the Master File.

You can use the full attribute name FILESUFFIX, or the shorter form SUFFIX.

Syntax

How to Identify a Data Source Type

```
[FILE]SUFFIX = data_source_type
```

where:

data_source_type

Indicates the type of data source or the name of a customized data access module. The default value is FIX, which represents a fixed-format sequential data source.

Example

Specifying the Type for a FOCUS Data Source

The following example specifies the data source type FOC, which represents a FOCUS data source:

```
SUFFIX = FOC
```

Reference

SUFFIX Values

The following table indicates the SUFFIX value for each data source type:

Data Source Type	SUFFIX Value
ADABAS	ADBSIN or ADBSINX ADABAS (OpenVMS EDA 2.x and FOCUS 6.x)
ALLBASE/SQL	SQLALB or ALLBASE
CA-Datcom/DB	DATACOM
CA-IDMS/DB	IDMSR
CA-IDMS/SQL	SQLIDMS
C-ISAM	C-ISAM
DB2	DB2 or SQLDS
DB2/2	SQLDBM
DB2/400	SQL400 (SQL access) DBFILE (native access)
DB2/6000	DB2
DBMS	DBMS
DMS	VSAM (keyed access) FIX (non-keyed access)
Digital Standard MUMPS	DSM
Enscribe	ENSC
Fixed-format sequential	FIX This value is the default. PRIVATE (for FOCSAM user exit)
FOCUS	FOC
Free-format (also known as comma-delimited) sequential	COM
Fusion	FUSION

Data Source Type	SUFFIX Value
Image/SQL	IMAGE
IMS	IMS
INFOAccess	SQLIAX
Information/Management	INFOMAN
Informix	SQLINF
Ingres	SQLING
KSAM	KSAM
Micronetics Standard MUMPS	MSM
Millennium	CPMILL or CPMILL2 (Release 2) CPMILL3 (Release 3)
Model 204	M204IN
Native Interface	SQLSAP
NETISAM	C-ISAM
NOMAD	NMDIN
NonStop SQL	NSSQL
Nucleus	SQLNUC
ODBC	SQLODBC
OpenIngres	SQLING
Open M/SQL	SQLMSQ
Oracle	SQLORA
PACE	VSAM
Progress	SQLPRO
Rdb	SQLRDB RDB (OpenVMS EDA 2.x and FOCUS 6.x)
Red Brick	SQLRED

Data Source Type	SUFFIX Value
RMS	RMS ISAM (OpenVMS EDA 2.x and FOCUS 6.x)
SQL/DS	SQLDS
SQL Server	SQLMSS
StorHouse	SQLSTH
SUPRA	SUPRA
Sybase	SQLSYB
System 2000	S2K
Teradata	SQLDBC
TOTAL	TOTIN
TurboIMAGE	IMAGE OMNIDEX (using OMNIDEX IMS indices)
Unify	SQLUNIFY
uniVerse	UNIVERSE
VSAM	VSAM PRIVATE (for FOCSAM user exit)
Non-standard data source	Name of the customized data access routine.

Providing Descriptive Information for a Data Source: REMARKS

The optional REMARKS attribute provides descriptive information about the data source. This descriptive information displays in graphical tools such as Report Assistant and Graph Assistant.

You can also include descriptive information as a comment following a \$ symbol in the Master File. For more information, see Chapter 1, *Understanding a Data Source Description*. These descriptions, however, are useful only for those who view the Master File source code. They are *not* displayed in graphical tools such as Report Assistant and Graph Assistant.

Example

Providing Descriptive Information for a Sybase Table

The following example shows the data source declaration for the Sybase table ORDERS. The data source declaration includes descriptive information about the table.

```
FILENAME = ORDERS, SUFFIX = SQLSYB,  
REMARKS = 'This Sybase table tracks daily, weekly, and monthly orders.', $
```

Since the descriptive information would not fit on the same line as the other data source attributes, the REMARKS attribute appears on a line by itself.

Specifying a Physical File Name: DATASET

You can add the DATASET attribute to the Master File to specify a physical location for the data source to be allocated. In addition, the DATASET attribute permits you to bypass the search mechanism for default data source location. DATASET eliminates the need to allocate data sources using JCL, FILEDEF, DYNAM, and USE commands.

User allocation and system specific behavior is as follows:

Platform	User Allocation Command
CMS	FILEDEF
TSO	DYNAM ALLOC or TSO ALLOC
UNIX/WinNT/OpenVMS	FILEDEF

Note: You cannot use both the ACCESSFILE attribute and the DATASET attribute in the same Master File. For information on the ACCESSFILE attribute, see Chapter 6, *Describing a FOCUS Data Source*.

DATASET Behavior in a FOCUS Data Source

The DATASET attribute can be used only at the file declaration level of the Master File. If the Master File's name is present in the USE list, or the user explicitly allocated the data file, a warning is issued and the DATASET attribute is ignored.

If DATASET is used in a Master File whose data source is managed by the FOCUS Database Server, the DATASET attribute is ignored on the server side, because the FOCUS Database Server does not read Master Files for servicing table requests.

The DATASET attribute in the Master File has the lowest priority:

- A user's explicit allocation overrides the DATASET attribute if you first issue the allocation command and then issue a CHECK FILE command to clear the previous DATASET allocation.
- The USE command for FOCUS data sources overrides DATASET attributes and explicit allocations.

An alternative to the DATASET attribute for allocating FOCUS data sources is an Access File. For detailed information, see Chapter 6, *Describing a FOCUS Data Source*.

Note: If a DATASET allocation is in effect, a CHECK FILE command must be issued in order to override it by an explicit allocation command. The CHECK FILE command will undo the allocation created by DATASET.

Syntax

How to Use the DATASET Attribute

```
{DATASET|DATA}='filename [ON sinkname]'
```

where:

filename

Is the platform-dependent physical name of the data source.

sinkname

Indicates that the data source is located on the FOCUS Database Server. This attribute is valid only for FOCUS data sources.

In MVS, the syntax is:

```
{DATASET|DATA}='qualifier.qualifier ...'
```

or

```
{DATASET|DATA}='ddname ON sinkname'
```

In CMS, the syntax is:

```
{DATASET|DATA}='filename filetype filemode [ON sinkname]'
```

In UNIX, the syntax is:

```
{DATASET|DATA}='/filesystem/filename.foc [ON sinkname]'
```

In NT, the syntax is:

```
{DATASET|DATA}='drive:\directory\filename.foc [ON sinkname]'
```

In OpenVMS, the syntax is:

```
{DATASET|DATA}='[device:[directory]]filename[.foc] [ON sinkname]'
```

Example Allocating a FOCUS Data Source Using the DATASET Attribute

The following example illustrates how to allocate a FOCUS data source using the DATASET attribute:

For MVS,

```
FILENAME=CAR,SUFFIX=FOC
DATASET='USER1.CAR.FOCUS'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

For CMS,

```
FILENAME=CAR,SUFFIX=FOC
DATASET='CAR FOCUS A'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

For UNIX,

```
FILENAME=CAR,SUFFIX=FOC
DATASET='/filesystem/filename.foc'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

For NT,

```
FILENAME=CAR,SUFFIX=FOC
DATASET='drive:\directory\filename.foc'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

For OpenVMS,

```
FILENAME=CAR,SUFFIX=FOC
DATASET='device:[directory]filename.foc'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

Example

Allocating a Data Source For the FOCUS Database Server

The following example illustrates how to allocate a FOCUS data source with the DATASET attribute using ON *sink*:

For MVS,

```
FILENAME=CAR,SUFFIX=FOC
DATASET='CAR ON SINK1'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

Note: The ddname CAR is allocated by the FOCUS Database Server JCL.

For CMS,

```
FILENAME=CAR,SUFFIX=FOC
DATASET='CAR FOCUS A ON SINK1'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

For UNIX,

```
FILENAME=CAR,SUFFIX=FOC
DATASET='filename ON sink'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

For NT,

```
FILENAME=CAR,SUFFIX=FOC
DATASET='filename ON sink'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

For OpenVMS,

```
FILENAME=CAR,SUFFIX=FOC
DATASET='filename ON sink'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

DATASET Behavior in a Fixed-Format Sequential Data Source

The DATASET attribute can be used only at the file declaration level of the Master File and cannot contain ON *sink*. If the DATASET attribute contains ON *sink*, a message is issued and the operation is terminated.

An explicit allocation of data for this Master File is checked for when the DATASET attribute is detected. If an explicit allocation exists, a warning message is issued informing the user that the DATASET value has been overridden, and the DATASET attribute is ignored. If this Master File name is not allocated, an internal command is issued to perform the allocation. This allocation is stored temporarily and is released when a new Master File is used or when the session terminates.

There is a second parameter for DATASET, required for the UNIX, NT, and OpenVMS platforms, which when used with SUFFIX=FIX data sources, tells whether it contains binary or text data. This parameter provides information on whether there are line break characters in the data source. This distinguishes a text data source from a binary data source. The default is binary.

Syntax**How to Use the DATASET Attribute With a Fixed-Format Data Source**

```
{DATASET|DATA}='filename'
```

```
{DATASET|DATA}='filename' {BINARY|TEXT}'
```

where:

filename

Is the platform-dependent physical name of the data source.

BINARY

Indicates that it is a binary data source. BINARY is the default value.

TEXT

Indicates that it is a text data source.

The DATASET attribute in the Master File has the lowest priority:

- A user's explicit allocation overrides DATASET attributes.

Note: If a DATASET allocation is in effect, a CHECK FILE command must be issued in order to override it by an explicit allocation command. The CHECK FILE command will undo the allocation created by DATASET.

Example**Allocating a Fixed-Format Data Source Using the DATASET Attribute**

The following examples illustrate how to allocate a fixed-format data source using the DATASET attribute:

```
1. FILE=XX, SUFFIX=FIX, DATASET='SEQFILE1 DATA A'
```

```
.
```

```
.
```

```
.
```

```
2. FILE=XX, SUFFIX=FIX, DATASET='USER1.SEQFILE1'
```

```
.
```

```
.
```

```
.
```

```
3. FILE=XX, SUFFIX=FIX, DATASET='C:\DATA\FILENAME.FTM TEXT'
```

```
.
```

```
.
```

```
.
```

```
4. FILE=XX, SUFFIX=FIX, DATASET='/u22/class/data/filename.ftm'
```

```
.
```

```
.
```

```
.
```

```
5. FILE=XX, SUFFIX=FIX, DATASET='/u22/class/data/filename.ftm BINARY'
```

DATASET Behavior in a VSAM Data Source

The DATASET attribute can be used on the file declaration level of the Master File and cannot contain ON *sink*. If the DATASET attribute contains ON *sink*, a message is issued and the operation is terminated.

An explicit allocation of data for this Master File is checked for when the DATASET attribute is detected. If an explicit allocation is found, a warning message is issued informing the user that the DATASET value has been overridden, and the DATASET attribute is ignored. If this Master File name is not allocated, an internal command is issued to perform the allocation. This allocation is stored temporarily and is released when a new Master File is used or when the session terminates.

The DATASET attribute may also appear on the field declaration level of the Master File to specify where to find an alternate index. Because of VSAM naming conventions (names are truncated to 8 characters), the name of the field alias will be used as the ddname. If a user allocation is found for the Master File or alternate index ddname, the DATASET attribute is ignored and a warning message issued.

Note: There is no limit on how many alternate indices you may have. It is also acceptable for some alternate indices to have the DATASET attribute while others do not. However, if a file level DATASET attribute is missing, the field level DATASET will be ignored.

Syntax

How to Use the DATASET Attribute With a VSAM Data Source

```
{DATASET|DATA}='filename'
```

where:

filename

Is the platform-dependent physical name of the data source or alternate index.

The DATASET attribute in the Master File has the lowest priority:

- A user's explicit allocation overrides DATASET attributes.

Note: If a DATASET allocation is in effect, a CHECK FILE command must be issued in order to override it by an explicit allocation command. The CHECK FILE command will de-allocate the allocation created by DATASET.

Example

Allocating a VSAM Data Source Using the DATASET Attribute

The following example illustrates how to allocate a VSAM data source on the file declaration level and for an alternate index:

```
FILE=EXERVSM1, SUFFIX=VSAM, DATASET='VSAM1.CLUSTER1', $
SEGNAME=ROOT, SEGTYPE=S0, $
GROUP=KEY1, ALIAS=KEY, FORMAT=A4, ACTUAL=A4, $
FIELD=FLD1, ALIAS=F1, FORMAT=A4, ACTUAL=A4, $
FIELD=FLD2, ALIAS=F2, FORMAT=A4, ACTUAL=A4, $
FIELD=FLD3, ALIAS=DD1, FORMAT=A4, ACTUAL=A4, FIELDTYPE = I,
DATASET='VSAM1.INDEX1', $
FIELD=FLD4, ALIAS=F4, FORMAT=A4, ACTUAL=A4, $
FIELD=FLD5, ALIAS=F5, FORMAT=A4, ACTUAL=A4, $
FIELD=FLD6, ALIAS=F6, FORMAT=A4, ACTUAL=A4, $
FIELD=FLD7, ALIAS=F7, FORMAT=A4, ACTUAL=A4, $
```

CHAPTER 3

Describing a Group of Fields

Topics:

- Defining a Single Group of Fields
- Identifying a Logical View: Redefining a Segment
- Relating Multiple Groups of Fields
- Logical Dependence: The Parent-Child Relationship
- Logical Independence: Multiple Paths
- Cardinal Relationships Between Segments
- The One-to-One Relationship
- The One-to-Many Relationship
- The Many-to-Many Relationship
- Recursive Relationships
- Relating Segments From Different Types of Data Sources
- Rotating a Data Source: An Alternate View

In a data source, certain fields may have a one-to-one correspondence and form a group. You can relate different groups to each other. For some types of data sources you can even define a logical view of a group—that is, a subset. You identify these groups, and relationships between these groups, by using attributes in the Master and Access File, as well as related facilities such as the JOIN command.

Defining a group of fields is described in *Defining a Single Group of Fields* on page 3-2. Relating these groups to each other is described in *Relating Multiple Groups of Fields* on page 3-7.

These topics describe the general concepts and explain how to implement them using Master File attributes. If your type of data source also requires an Access File, see the appropriate data adapter documentation for supplementary information about defining groups and group relations in the Access File.

For Maintain:

- The JOIN command, the MATCH FILE command, alternate views, the SET OPTIMIZATION command, dynamic joins (SEGTYPE of DKU), and recursive joins are not supported.
- Static joins (SEGTYPE of KU) are supported.

Defining a Single Group of Fields

In a data source, certain fields may have a one-to-one correspondence: for each value of a field, the other fields will have exactly one corresponding value. For example, consider the EMPLOYEE data source:

- Each employee has one ID number and the number is unique to that employee.
- For each ID number—that is, for each employee—there is one first and last name, one date hired, one department, and one current salary.

In the data source, a field represents each of these employee characteristics. The group of fields represents the employee. In Master File terms, this group is called a *segment*.

Understanding a Segment

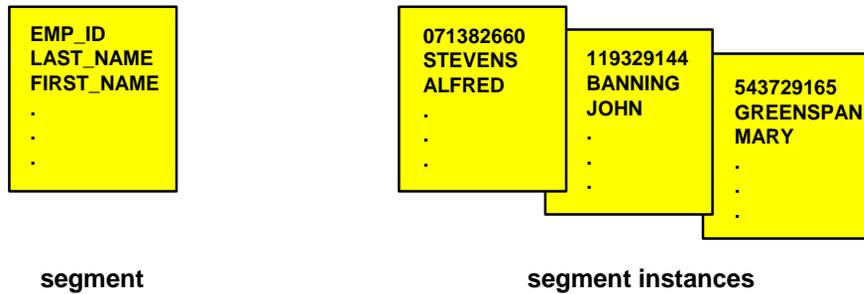
While the term *segment* may not be familiar to you, the concept behind it is universal. A segment is a group of fields that have a one-to-one correspondence with each other and usually describe a group of related characteristics. In a relational data source, a segment is equivalent to a table. Segments are the building blocks of larger data structures. You can relate different segments to each other, and describe the new structures, as discussed in *Relating Multiple Groups of Fields* on page 3-7.

 <p>ID# 199329144 John Banning . . .</p> <p>works in Production dept. earns \$29,700</p> <p>Real World</p>	<p>EMP_ID LAST_NAME FIRST_NAME . . . DEPARTMENT CURR_SAL . . .</p> <p>Segment (describes the real world)</p>	<p>SEGNAME=EMPINFO FIELDNAME=EMP_ID, ... FIELDNAME=LAST_NAME, ... FIELDNAME=FIRST_NAME, FIELDNAME=DEPARTMENT, ... FIELDNAME=CURR_SAL,</p>
--	---	--

Understanding a Segment Instance

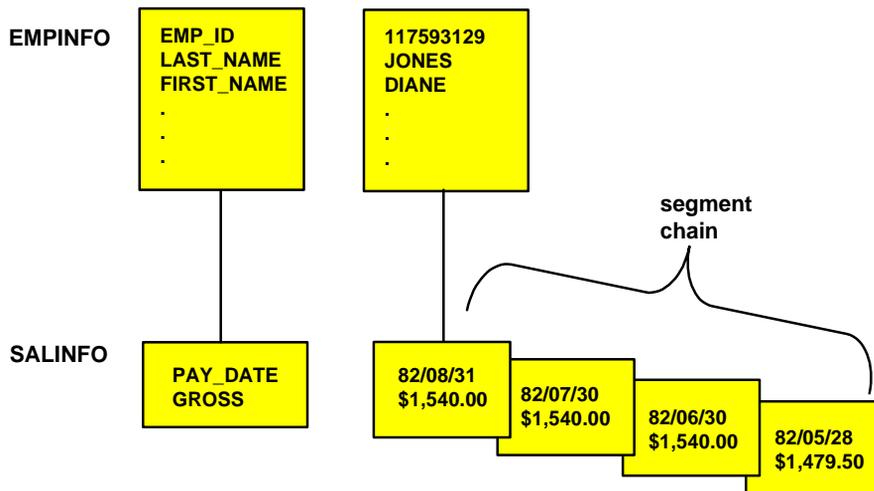
While a segment is abstract—a description of data—the segment instances that correspond to it are the actual data. Each instance is an occurrence of segment values found in the data source. For a relational data source, an instance is equivalent to a row in a table. In a single segment data source, a segment instance is the same as a record.

The relationship of a segment to its segment instances is illustrated in the following diagram:



Understanding a Segment Chain

The instances of a segment that are descended from a single parent instance are collectively known as a segment chain. In the special case of a root segment, which has no parent, all of the root instances form a chain. The parent-child relationship is discussed in *Logical Dependence: The Parent-Child Relationship* on page 3-9.



You describe a segment using the SEGNAME and SEGTYPE attributes in the Master File. The SEGNAME attribute is described in *Identifying a Segment: SEGNAME* on page 3-4.

Identifying a Key Field

Most segments also have key fields—that is, one or more fields that uniquely identify each segment instance. In the EMPLOYEE data source, the ID number is the key because each employee has one ID number, and no other employee has the same number. The ID number is represented in the data source by the EMP_ID field.

If your data source uses an Access File, you may need to specify which fields serve as keys by identifying them in the Access File. If the data source also happens to be a relational data source, the fields constituting the primary key in the Master File should be the first fields described for that segment—that is, their field declarations should come before any others in that segment.

For FOCUS data sources, you identify key fields and their sorting order using the SEGTYPE attribute in the Master File, as shown in Chapter 6, *Describing a FOCUS Data Source*. You position the key fields as the first fields in their segment.

Identifying a Segment: SEGNAME

The SEGNAME attribute identifies the segment. It is the first attribute you specify in a segment declaration. SEGNAME has an alias of SEGMENT.

You can give the segment any name consisting of up to eight characters. You will probably want to make the Master File self-documenting by setting SEGNAME to something meaningful to the user or the native file manager. For example, if you are describing a DB2 table, you might want to assign the table name (or an abbreviation) to SEGNAME.

In a Master File, each segment name must be unique. The only exception to this rule is in a FOCUS data source, where cross-referenced segments in Master File defined joins can have the same name as other cross-referenced segments in Master File defined joins. If cross-referenced segments do have identical names, you can still refer to them uniquely by using the CRSEGNAME attribute. See Chapter 7, *Defining a Join in a Master File* for more information.

In a FOCUS data source, you cannot change the value of SEGNAME once data has been entered into the data source. For all other types of data sources, you can change SEGNAME as long as you also change all references to it—for example, any references in the Master and Access File.

If your data source uses an Access File as well as a Master File, you must specify the same segment name in both.

Syntax

How to Identify a Segment

```
{SEGNAME|SEGMENT} = segment_name
```

where:

segment_name

Is the name you want to use to identify this segment. It can be up to a maximum of eight characters long.

The first character must be a letter, and the remaining characters can be any combination of letters, numbers, and underscores (_). Information Builders does not recommend using other characters, because they may cause problems in some operating environments or when resolving expressions.

Example

Identifying a Segment

If a segment corresponds to a relational table named TICKETS, and you want to give the segment the same name, you could use the SEGNAME attribute in the following way:

```
SEGNAME = TICKETS
```

Identifying a Logical View: Redefining a Segment

The segments that you define usually correspond to underlying groups in your data source. For example, a segment could be a table in a relational data source.

However, you are not limited to using the segment as it was originally defined in the native data source. You can define a logical view in which you include only a subset of the segment's fields (similar to a relational view), or else define the unwanted fields as one or more filler fields. This technique can be helpful if, for example, you want to make only some of the segment's fields available to an application or its users.

You can use the following methods with the following types of data sources:

- **Relational data sources.** You can omit unwanted fields from the segment description in the Master File.
- **Sequential and FOCUS data sources.** You can define unwanted fields as one or more filler fields. You would define the field's format to be alphanumeric, its length to be the number of bytes making up the underlying fields, and its name and alias to be blank. Field declarations and length are discussed in detail in Chapter 4, *Describing an Individual Field*.

If you want to explicitly restrict access at the file, segment, or field level based on user ID, field values, and other characteristics, you can use the DBA facility, as described in Chapter 9, *Providing Data Source Security: DBA*.

Example

Omitting a Field: Creating a Segment Subset

Consider the following Master File for an Oracle table named EMPFACTS:

```
FILENAME = EMPFACTS, SUFFIX = SQLORA , $
SEGNAME = EMPFACTS, SEGTYPE = S0 , $
    FIELDNAME = EMP_NUMBER, ALIAS = ENUM, USAGE = A9, ACTUAL = A9 , $
    FIELDNAME = LAST_NAME, ALIAS = LNAME, USAGE = A15, ACTUAL = A15 , $
    FIELDNAME = FIRST_NAME, ALIAS = FNAME, USAGE = A10, ACTUAL = A10 , $
    FIELDNAME = HIRE_DATE, ALIAS = HDT, USAGE = I6YMD, ACTUAL = DATE , $
    FIELDNAME = DEPARTMENT, ALIAS = DPT, USAGE = A10, ACTUAL = A10 , $
    FIELDNAME = SALARY, ALIAS = SAL, USAGE = D12.2M, ACTUAL = D8 , $
    FIELDNAME = JOBCODE, ALIAS = JCD, USAGE = A3, ACTUAL = A3 , $
    FIELDNAME = OFFICE_NUM, ALIAS = OFN, USAGE = I8, ACTUAL = I4 , $
```

If you develop an application that refers to only an employee's ID and name, and you want this to be reflected in the application's view of the segment, you can code an alternative Master File that names only the desired fields:

```
FILENAME = EMPFACTS, SUFFIX = SQLORA , $
SEGNAME = EMPFACTS, SEGTYPE = S0 , $
    FIELDNAME = EMP_NUMBER, ALIAS = ENUM, USAGE = A9, ACTUAL = A9 , $
    FIELDNAME = LAST_NAME, ALIAS = LNAME, USAGE = A15, ACTUAL = A15 , $
    FIELDNAME = FIRST_NAME, ALIAS = FNAME, USAGE = A10, ACTUAL = A10 , $
```

Example

Redefining a Field: Creating a Filler Field

Consider the EMPINFO segment of the EMPLOYEE data source:

```
SEGNAME = EMPINFO, SEGTYPE = S1 , $
    FIELDNAME = EMP_ID, ALIAS = EID, USAGE = A9 , $
    FIELDNAME = LAST_NAME, ALIAS = LN, USAGE = A15 , $
    FIELDNAME = FIRST_NAME, ALIAS = FN, USAGE = A10 , $
    FIELDNAME = HIRE_DATE, ALIAS = HDT, USAGE = I6YMD , $
    FIELDNAME = DEPARTMENT, ALIAS = DPT, USAGE = A10 , $
    FIELDNAME = CURR_SAL, ALIAS = CSAL, USAGE = D12.2M , $
    FIELDNAME = CURR_JOBCODE, ALIAS = CJC, USAGE = A3 , $
    FIELDNAME = ED_HRS, ALIAS = OJT, USAGE = F6.2 , $
```

If you develop an application that refers to only an employee's ID and name, and you want this to be reflected in the application's view of the segment, you can code an alternative Master File that explicitly names only the desired fields:

```
SEGNAME = EMPINFO, SEGTYPE = S1 , $
    FIELDNAME = EMP_ID, ALIAS = EID, USAGE = A9 , $
    FIELDNAME = LAST_NAME, ALIAS = LN, USAGE = A15 , $
    FIELDNAME = FIRST_NAME, ALIAS = FN, USAGE = A10 , $
    FIELDNAME = , ALIAS = , USAGE = A29 , $
```

Note that the filler field is defined as an alphanumeric field of 29 bytes, which is the combined internal length of the fields it replaces: HIRE_DATE (4 bytes), DEPARTMENT (10 bytes), CURR_SAL (8 bytes), CURR_JOBCODE (3 bytes), and ED_HRS (4 bytes).

Relating Multiple Groups of Fields

Once you have described a group of fields—that is, a segment—you can relate segments to each other to build more sophisticated data structures. You can:

- **Describe physical relationships.** If groups of fields are already physically related in your data source, you can describe the relationship.
- **Describe logical relationships.** You can describe a logical relationship between any two segments that have at least one field in common by joining them. The underlying data structures remain physically separate, but they are treated as if they were part of a single structure. The new structure can include segments from the same or different types of data sources.

Note that if you are creating a new FOCUS data source, you can implement segment relationships in several ways, depending upon your design goals, as described in Chapter 6, *Describing a FOCUS Data Source*.

To describe a data structure containing several segments—whether it is a multi-segment data source or several data sources that have been joined together—you should be aware of the following:

- Logical dependence between related segments.
- Logical independence between unrelated segments.

Facilities for Specifying a Segment Relationship

There are several facilities for specifying relationships between segments. The use of a Master and Access File to specify a relationship is fully documented in this chapter. The JOIN command, which joins segments into a structure from which you can report, is fully described in the *Creating Reports With WebFOCUS Language* manual.

Note that a related facility, the MATCH FILE command, enables you to implement many different types of sophisticated relationships by first describing the relationship as a series of extraction and merging conditions, and then merging the related data into a new single segment data source. The result is not a joined structure, but an entirely new data source that can be further processed. The original data sources themselves remain unchanged. The MATCH FILE command is documented in the *Creating Reports With WebFOCUS Language* manual.

Identifying a Parent Segment: PARENT

The PARENT attribute identifies a segment's parent. You specify the PARENT attribute in the segment declaration of the Master File. Because a root segment has no parent, you do not specify the PARENT segment when declaring a root.

Note that a parent segment must be declared in the Master File before any of its child segments.

If the parent-child relationship is permanently implemented within the structure of the data source, as, for example, within a FOCUS data source, then you cannot change the parent attribute without changing the underlying structure of the data source. However, if the parent-child relationship is temporary, as, for example, when you join several relational tables in the Master File, then you can change the PARENT attribute.

Syntax

How to Identify the Parent Segment

`PARENT = segment_name`

where:

`segment_name`

Is the name of the segment's parent as previously declared in the Master File.

If you do not specify the PARENT attribute, it defaults to the value of the most recently specified segment. If the PARENT attribute has not been specified in any prior segment declarations in this Master File, the previous segment becomes the parent.

Information Builders recommends using the PARENT attribute for unique segments with a SEGTYPE of U.

Example

Identifying a Parent Segment

In the EMPLOYEE data source, DEDUCT's parent is SALINFO, and so the segment declaration for DEDUCT includes the following attribute:

`PARENT = SALINFO`

Identifying the Type of Relationship: SEGTYPE

The SEGTYPE attribute specifies the type of relationship that a segment has to its parent. SEGTYPE is part of the segment declaration and is used in different ways with different types of data sources. For sequential, VSAM, and ISAM data sources, see Chapter 5, *Describing a Sequential, VSAM, or ISAM Data Source*. For FOCUS data sources, see Chapter 6, *Describing a FOCUS Data Source*. For other types of data sources, see the appropriate data adapter documentation for details.

Logical Dependence: The Parent-Child Relationship

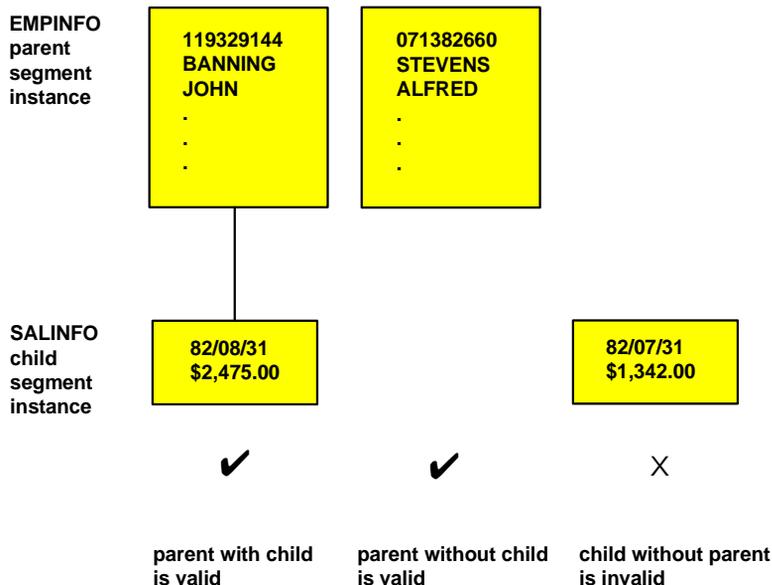
Logical dependence between segments is expressed in terms of the parent-child relationship: a child segment is dependent upon its parent segment. This means that an instance of the child segment can exist only if a related instance of the parent segment exists. The parent segment has logical precedence in the relationship, and is retrieved first when the data source is accessed.

Note that if the parent-child relationship is logical and not physical—that is, if it is implemented as a join—it is possible to have a child instance without a related parent instance. In this case, the child instance will not be accessible through the join, although it will still be accessible independently.

If a join relates the parent and child segments, the parent is known as the host segment, and the child is known as the cross-referenced segment. The fields on which the join is based—that is, the matching fields in the host and cross-referenced segments—are known respectively as the host and cross-referenced fields.

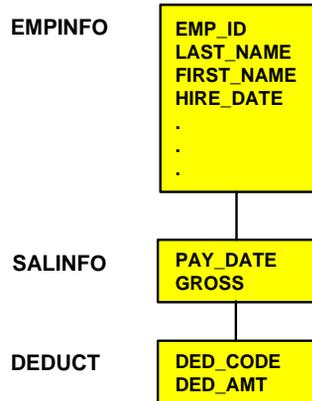
A Simple Parent-Child Relationship

In the EMPLOYEE data source, the EMPINFO and SALINFO segments are related: EMPINFO identifies an employee by ID number, while SALINFO contains the employee's pay history. EMPINFO is the parent segment, and SALINFO is a child segment dependent upon it. This relationship is illustrated by the fact that it is possible to have an employee identified by ID and name for whom no salary information has been entered—that is, the parent instance without the child instance. However, it is meaningless to have salary information for an employee if we do not know who the employee is—that is, a child instance without the parent instance.



A Parent-Child Relationship With Multiple Segments

Consider the following diagram of a portion of the EMPLOYEE data source, containing the EMPINFO, SALINFO, and DEDUCT segments. DEDUCT contains payroll deduction information for each paycheck.



EMPINFO is related to SALINFO, and in this relationship EMPINFO is the parent segment and SALINFO is the child segment. SALINFO is also related to DEDUCT. In this second relationship, SALINFO is the parent segment and DEDUCT is the child segment. Just as SALINFO is dependent upon EMPINFO, DEDUCT is dependent upon SALINFO.

Types of Parent and Child Segments

A segment can be one of the following types:

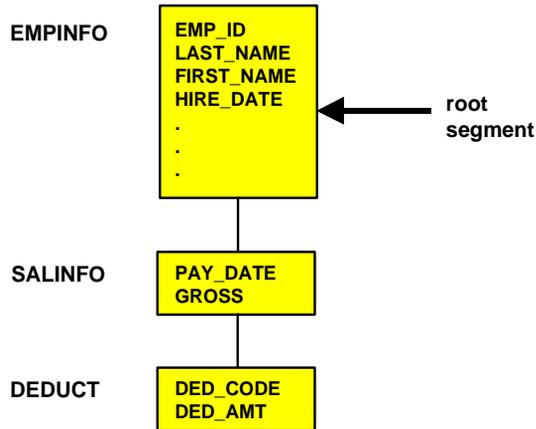
Root segment. The segment that has logical precedence over the entire data structure—the parent of the entire structure—is called the *root* segment. The term *root* is used because a data structure can branch like a tree, and the root segment, like the root of a tree, is the source of the structure.

Descendant segment. We refer to a segment's direct and indirect children collectively as its descendant segments.

Ancestral segment. We refer to a segment's direct and indirect parents as ancestral segments.

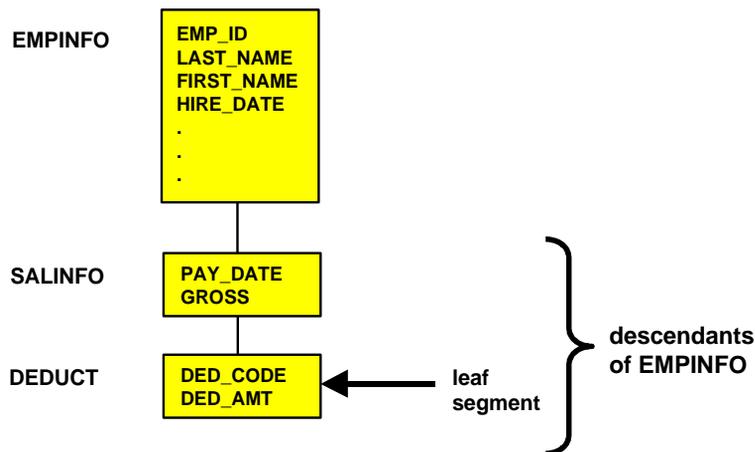
Understanding a Root Segment

In the EMPLOYEE data source, EMPINFO is the root; it has no parent, and all other segments in the structure are its children directly (SALINFO) or indirectly (DEDUCT).



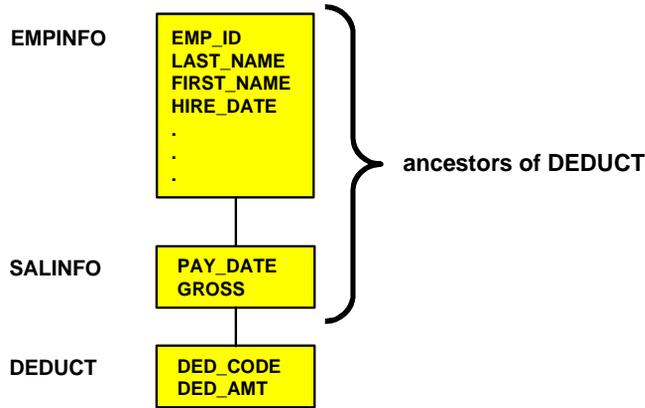
Understanding a Descendant Segment

SALINFO and DEDUCT are descendants of EMPINFO. DEDUCT is also a descendant of SALINFO. A descendant segment that has no children is called a leaf segment (because the branching of the data structure tree ends with the leaf). DEDUCT is a leaf.



Understanding an Ancestral Segment

In the EMPLOYEE data source, SALINFO and EMPINFO are ancestors of DEDUCT.

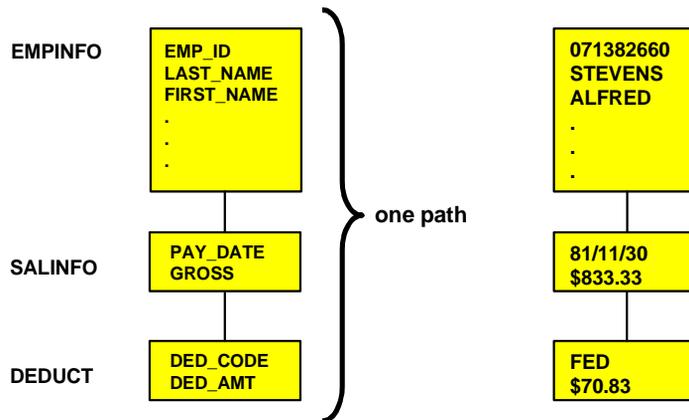


Logical Independence: Multiple Paths

A group of segments that are related to each other as a sequence of parent-child relationships, beginning with the root segment and continuing down to a leaf, is called a path. Because the path is a sequence of parent-child relationships, each segment is logically dependent upon all of the segments higher in the path.

Understanding a Single Path

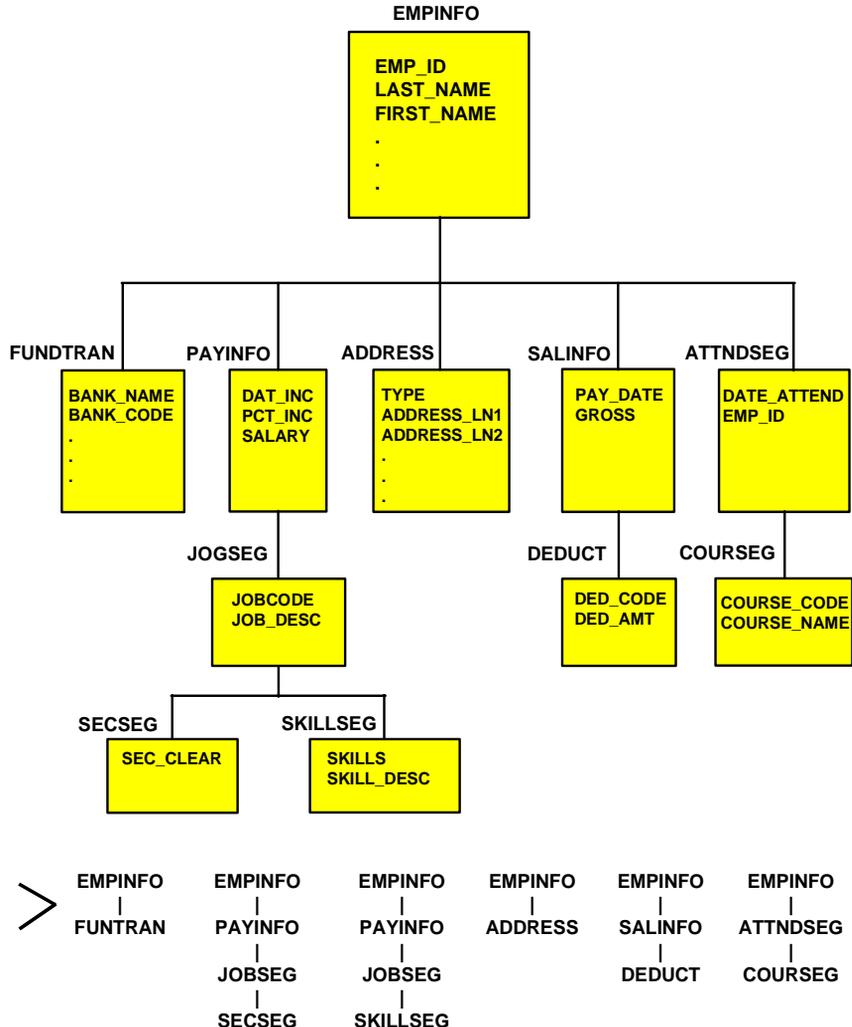
In the following view of the EMPLOYEE data source, EMPINFO, SALINFO, and DEDUCT form a path. An instance of DEDUCT (paycheck deductions) can exist only if a related instance of SALINFO (the paycheck) exists, and the instance of SALINFO (the employee's paycheck) can exist only if a related instance of EMPINFO (the employee) exists.



Understanding Multiple Paths

Consider the full EMPLOYEE structure, which includes the EMPLOYEE data source and the JOBFILE and EDUCFILE data sources that have been joined to it.

This is a multi-path data structure; there are several paths, each beginning with the root segment and ending with a leaf. Every leaf segment in a data structure is the end of a separate path.



Understanding Logical Independence

The EMPLOYEE data structure has six paths. The paths begin with the EMPINFO segment (the root), and end with:

- The FUNDTRAN segment
- The SECSEG segment
- The SKILLSEG segment
- The ADDRESS segment
- The DEDUCT segment
- The COURSEG segment

Each path is logically independent of the others. For example, an instance of DEDUCT is dependent upon its ancestor segment instances SALINFO and EMPINFO; but the ADDRESS segment lies in a different path, so DEDUCT is independent of ADDRESS.

An employee's deductions are identified by the paycheck from which they came, so deduction information can be entered into the data source only if the paycheck from which the deduction was made is entered first. However, deductions are not identified by the employee's address; an employee's paycheck deduction can be entered without the employee's address being known, and conversely the employee's address can be entered before any paychecks and deductions have been entered into the data source.

Cardinal Relationships Between Segments

The following types of cardinal relationships between groups of data are supported:

- One-to-one (1:1)
- One-to-many (1:M)
- Many-to-many (M:M)

You can define these relationships between:

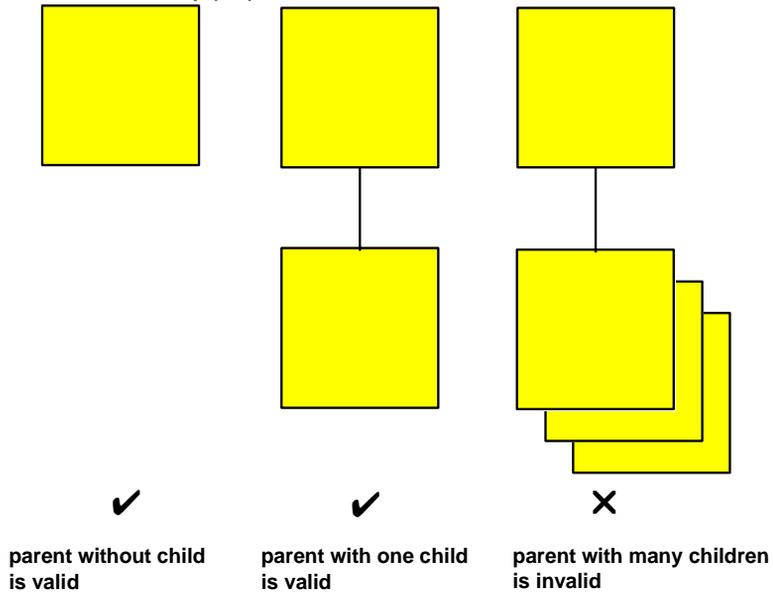
- Instances of different segments.
- Instances of the same segment—that is, a recursive or bill-of-materials relationship.
- Segments from the same type of data source.
- Segments from different types of data sources. For example, you can define the relationship between an Oracle table and a FOCUS data source. Note that you can join different types of data sources only by using the JOIN command, not by defining the join in the Master or Access File.
- If you are using a network data source, you can also “rotate” the data source after you have defined it, creating an alternate view that reverses some of the data relationships and enables you to access the segments in a different order.

The One-to-One Relationship

The fields in a segment have a one-to-one relationship with each other. Segments can also exhibit a one-to-one relationship; each instance of a parent segment can be related to one instance of a child segment, as shown in the following diagram. Because the relationship is one-to-one, the parent instance will never be related to more than one instance of the child. Of course, not every parent instance must have a matching child instance.

The child in a one-to-one relationship is referred to as a unique segment. The term refers to the fact that there can never be more than a single child instance.

One-to-One Relationship (1:1)

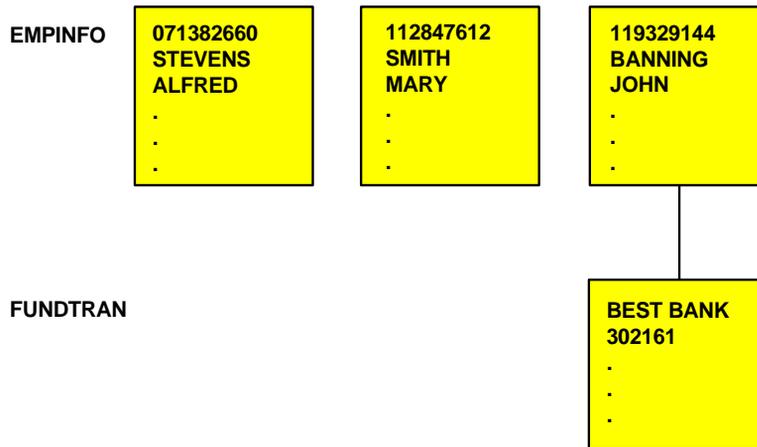


Understanding a One-to-One Relationship

In the EMPLOYEE data source, each EMPINFO segment instance describes one employee's ID number, name, current salary, and other related information. Some employees have joined the Direct Deposit program, which deposits their paychecks directly into their bank accounts each week. For these employees, the data source also contains the name of their bank and their account number.

Because only one set of bank information is needed for each employee (since each employee's paycheck is deposited into only one account), a one-to-one relationship exists between employee ID fields and bank information fields. But because participation in the Direct Deposit program is limited, only some employee data sources contain bank information; the other employees' data sources do not need bank fields.

The data source was designed with storage efficiency in mind, so the bank fields have been put into a separate segment called FUNDTRAN. Space will be used for the banking information only if it is needed—that is, an instance of FUNDTRAN will be created only as necessary. If banking fields were included in the parent segment (EMPINFO), the EMPINFO segment for each employee would reserve space for the banking fields, yet those fields would be empty in most cases.



Where to Use a One-to-One Relationship

You can specify a segment as unique to enforce a one-to-one relationship when you retrieve data. When you retrieve data from a segment described as unique, the request treats the unique segment as an extension of its parent. If the unique segment has multiple instances, the request retrieves only one. If the unique segment has no instances, the request substitutes default values for the missing segment's fields: zero (0) for numeric fields, blank () for alphanumeric fields, and the missing value for fields that have the MISSING attribute specified. The MISSING attribute is described in Chapter 4, *Describing an Individual Field*.

Implementing a One-to-One Relationship in a Relational Data Source

You can describe this relationship by joining the tables in the Master File and specifying a SEGTYPE of U for the child table. For more information on joining the tables in a Master File, see the appropriate data adapter documentation. Alternatively, you can join the tables by issuing the JOIN command without the ALL phrase and turning off the SQL Optimization facility with the SET OPTIMIZATION command.

Implementing a One-to-One Relationship in a Sequential Data Source

You can specify this relationship between two records by issuing the JOIN command without the ALL phrase.

Implementing a One-to-One Relationship in a FOCUS Data Source

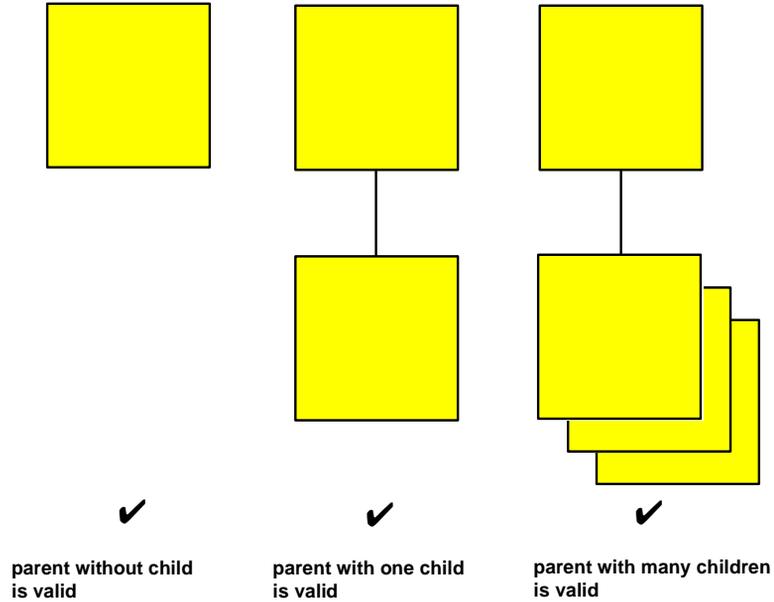
You can describe this relationship by specifying a SEGTYPE of U for the child segment. Alternately, you can join segments by issuing the JOIN command without the ALL phrase, or by specifying a unique join in the Master File using a SEGTYPE of KU (for a static join) or DKU (for a dynamic join). All of these SEGTYPE values are described in Chapter 6, *Describing a FOCUS Data Source*.

You can also describe a one-to-one segment relationship, in the Master File or using the JOIN command, as a one-to-many relationship. This technique gives you greater flexibility but does not enforce the one-to-one relationship when reporting or entering data and does not use resources as efficiently.

The One-to-Many Relationship

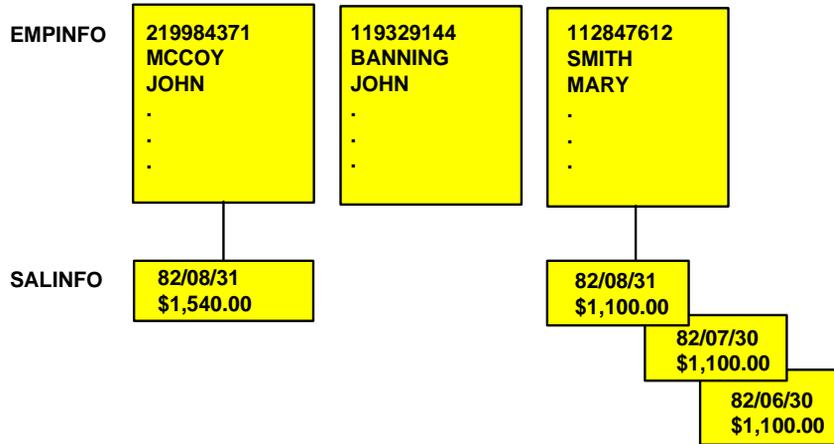
The most common relationship between two segments is the one-to-many relationship; each instance of a parent segment can be related to one or more instances of a child segment, as shown in the following diagram. Of course, not every parent instance needs to have matching child instances.

One-to-Many Relationship (1:M)



Understanding a One-to-Many Relationship

In the EMPLOYEE data source, each EMPINFO segment instance describes one employee's ID number, name, current salary, and other related information. Each SALINFO segment contains an employee's gross salary for each month. Most employees work for many months, so the relationship between EMPINFO and SALINFO is one-to-many.



Implementing a One-to-Many Relationship in a Relational Data Source

You can describe this relationship by joining the tables in the Master File and specifying a SEGTYPE of S0 for the child table. For more information on joining the tables in a Master File, see the appropriate data adapter documentation. Alternately, you can join the tables by issuing the JOIN command with the ALL phrase.

Implementing a One-to-Many Relationship in a VSAM or Sequential Data Source

You can describe a one-to-many relationship between a record and a group of multiply occurring fields within the record.

- The OCCURS attribute specifies how many times the field (or fields) occur.
- The POSITION attribute specifies where in the record the field (or fields) occur if they are not at the end of the record.
- The ORDER field determines the sequence number of an occurrence of a field.
- The PARENT attribute indicates the relationship between the singly occurring and multiply occurring fields.

The OCCURS and POSITION attributes and the ORDER field are all described in Chapter 5, *Describing a Sequential, VSAM, or ISAM Data Source*.

You can describe a one-to-many relationship between different records by using a RECTYPE field to indicate the type of each record, and the PARENT attribute to indicate the relationship between the different records. RECTYPE fields are described in Chapter 5, *Describing a Sequential, VSAM, or ISAM Data Source*.

You can also specify a one-to-many relationship between two records in different data sources by issuing the JOIN command with the ALL phrase or defining the join in the Master File. See the *Creating Reports With WebFOCUS Language* manual for information about the JOIN command, and see Chapter 7, *Defining a Join in a Master File*, for information about joins in a Master File.

Implementing a One-to-Many Relationship in a FOCUS Data Source

You can describe this relationship by specifying a SEGTYPE of Sn or SHn for the child segment. Alternatively, you can join the segments by issuing the JOIN command with the ALL phrase or by specifying a join in the Master File with a SEGTYPE of KM (for a static join) or DKM (for a dynamic join). All of these SEGTYPE values are described in Chapter 6, *Describing a FOCUS Data Source*.

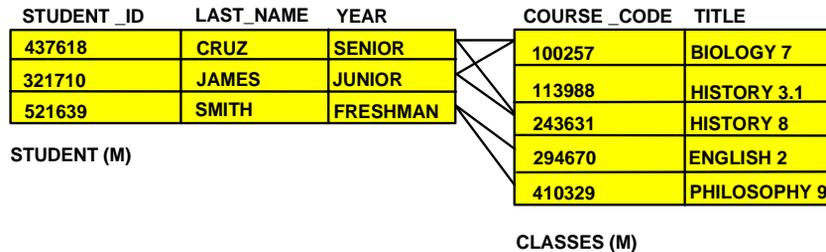
The Many-to-Many Relationship

A less commonly used relationship is many-to-many; each instance of one segment can be related to one or more instances of a second segment, and each instance of the second segment can be related to one or more instances of the first segment. It is possible to implement this relationship:

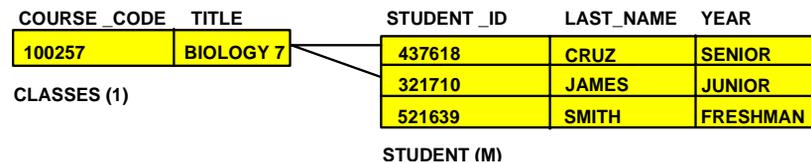
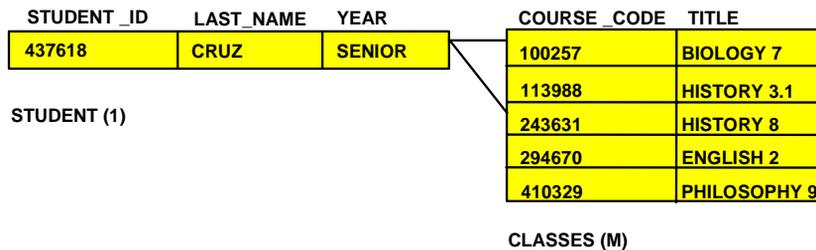
- **Directly** between two relational tables.
- **Indirectly** between segments of other types of data sources. Some non-relational data sources cannot represent a many-to-many relationship directly. However, they can represent it indirectly, and you can describe it as such.

Implementing a Many-to-Many Relationship Directly

The STUDENT table contains one row for each student enrolled at a college, and the CLASSES table contains one row for each class offered at the college. Each student can take many classes, and many students can take each class. This type of relationship is illustrated in the following diagram:



When the M:M relationship is seen from the perspective of either of the two tables, it looks like a 1:M relationship: one student taking many classes (1:M from the perspective of STUDENT), or one class taken by many students (1:M from the perspective of CLASSES).



When you report from or update the tables, at any one time the M:M relationship is seen from the perspective of one of the tables—that is, it sees a 1:M relationship. You decide which table's perspective to use by making that table the parent (host) segment, in the Master File or JOIN command. You describe the join in the Master File or JOIN command as you would for a standard one-to-many relationship.

To describe the relationship from the perspective of the STUDENT table, use the JOIN command as follows:

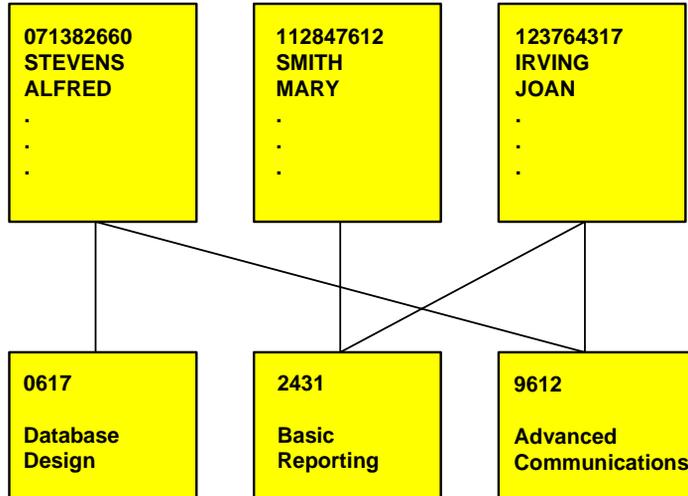
```
JOIN STUDENT_ID IN STUDENT TO ALL STUDENT_ID IN CLASSES
```

Describe the relationship from the perspective of the CLASSES table as follows:

```
JOIN COURSE_CODE IN CLASSES TO ALL COURSE_CODE IN STUDENT
```

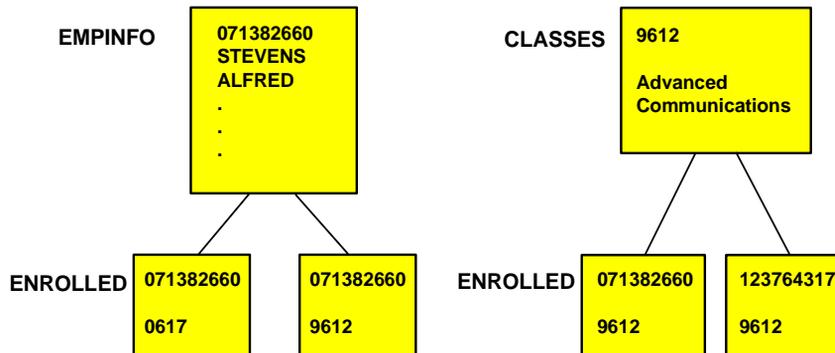
Implementing a Many-to-Many Relationship Indirectly

Consider the EMPINFO segment in the EMPLOYEE data source and the CLASSES segment in a hypothetical SCHOOL data source. Each instance of EMPINFO describes one employee, and each instance of CLASSES would describe one course. Each employee can take many courses, and many employees can take each course, so this is a many-to-many relationship.

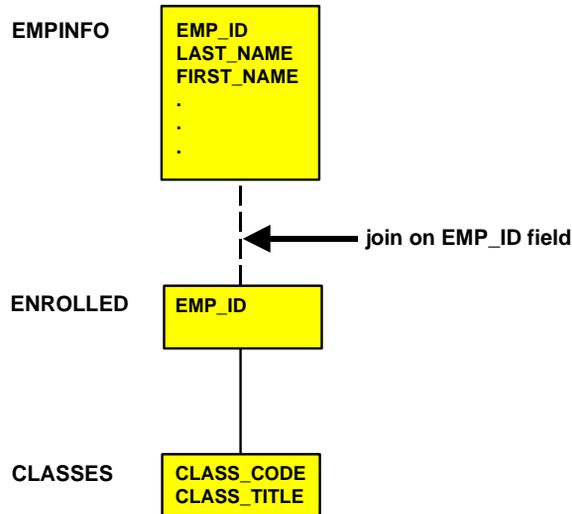


However, because some types of data sources cannot represent such a relationship directly, we need to introduce a mediating segment called ENROLLED. This new segment contains the keys from both of the original segments, EMP_ID and CLASS_CODE, and, in a sense, it represents the relationship between the two original segments. The new segment breaks up the M:M relationship into two 1:M relationships. Each instance of EMPINFO can be related to many instances of ENROLLED (because one employee can be enrolled in many classes), and each instance of CLASSES can be related to many instances of ENROLLED (because one class can have many employees enrolled in it).

These relationships are illustrated in the following diagram.



The next step is to make the mediating segment a child of one of the two original segments. You can design the SCHOOL data source so that CLASSES is the root and ENROLLED is the child of CLASSES. Note that when ENROLLED was an unattached segment it explicitly contained the keys (EMP_ID and CLASS_CODE) from both original segments. Yet as part of the SCHOOL data source, CLASS_CODE is implied by the parent-child relationship with CLASSES, and it can be removed from ENROLLED. You can then join EMPINFO and ENROLLED together:



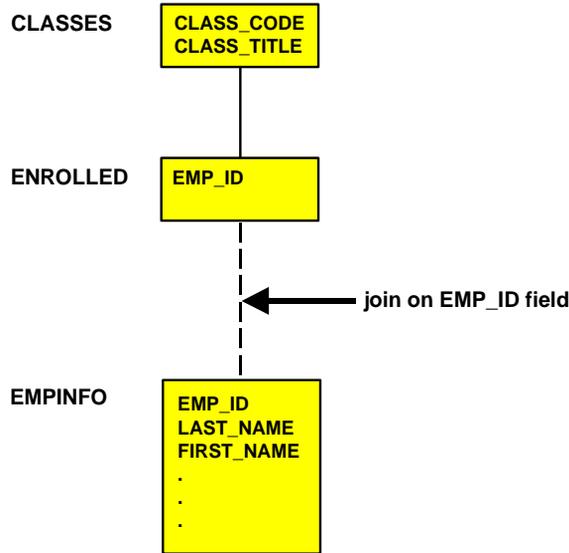
When the original M:M relationship is seen from this perspective, it looks like a 1:M:1 relationship. That is, one employee (EMPINFO) is enrolled many times (ENROLLED), and each enrollment is for a single class (CLASSES).

When you report from or update the new structure, the relationship is seen from the perspective of one of the original segments—in this case, from EMPINFO or CLASSES. You determine which segment's perspective is used by making that segment the parent in the join. You describe the join using the JOIN command, or for FOCUS data sources, alternately in the Master File. If you make the mediating segment, in this case ENROLLED, the child (cross-referenced) segment in the join, you implement the relationship as a standard one-to-many; if you make it the parent (host) segment, you implement the relationship as a standard one-to-one join.

For example, you can use the JOIN command to describe the relationship from the perspective of the CLASSES segment—that is, making ENROLLED the join’s host—as follows:

```
JOIN EMP_ID IN ENROLLED TO EMP_ID IN EMPINFO
```

The new structure looks like the following:



Another example that uses a join defined in the Master File is illustrated by the sample FOCUS data sources EMPLOYEE and EDUCFILE. Here, ATTNDSEG is the mediating segment between EMPINFO and COURSESEG.

Recursive Relationships

Generally, you use one-to-one and one-to-many relationships to join two different segments, usually in two different data sources. However, you can also join the same data source to itself, and even join the same segment to itself. This technique, which has many useful applications, is called a recursive join.

Recursive joins are described in more detail in the *Creating Reports With WebFOCUS Language* manual.

Example

A Recursive Join With a Single Segment

Assume that you have a single-segment data source called **MANAGER**, which includes the ID number of an employee, the employee’s name, and the ID number of the employee’s manager:



If you want to generate a report showing every employee's ID number, name, and manager's ID number and name, you would need to join the segment to itself. You would issue the following command:

```
JOIN MANAGER_ID IN MANAGER TO ID IN MANAGER AS BOSS
```

which would create the following structure:



Note that you can uniquely refer to fields in cross-referenced segments by prefixing them with the first four letters of the join name (BOSS in this example). The only exception is the cross-referenced field, for which the alias is prefixed instead of the field name.

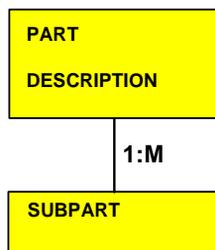
Once you have issued the join, you can generate an answer that looks like this:

ID	NAME	MANAGER_ID	BOSSNAME
---	----	-----	-----
026255	JONES	837172	CRUZ
308743	MILBERG	619426	WINOKUR
846721	YUTANG	294857	CAPRISTI
743891	LUSTIG	089413	SMITH
585693	CAPRA	842918	JOHNSON

Example

A Recursive Join With Multiple Segments

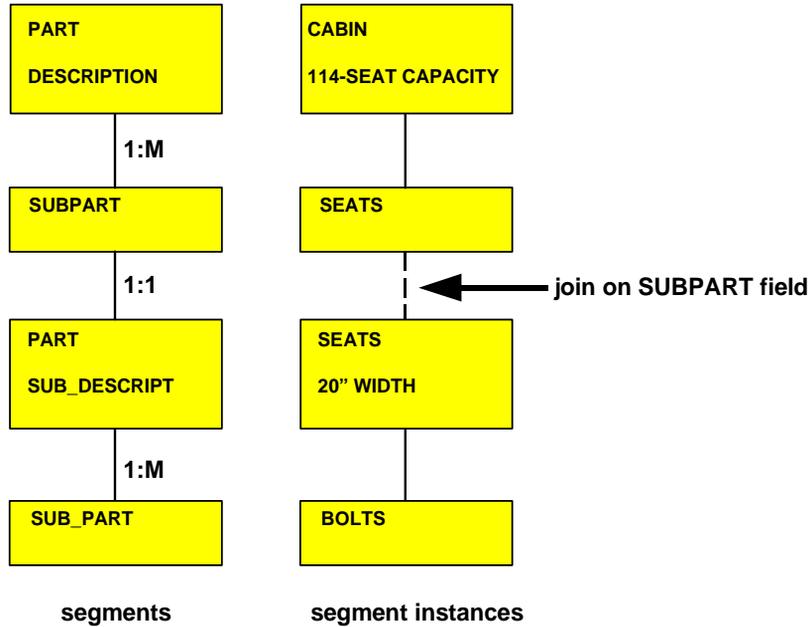
You can recursively join larger structures as well. For example, consider a two-segment data source called AIRCRAFT that stores a bill-of-materials for an aircraft company. The root segment has the name and description of a part, and the child segment has the name of a subpart. For each part, there can be many subparts.



While many of the larger parts are constructed of several levels of subparts, some of these subparts, such as bolts, are used throughout aircraft at many different levels. To give each occurrence of a subpart its own segment instance would produce redundancy. Instead, we can use the two-segment design shown previously and then join the data source to itself:

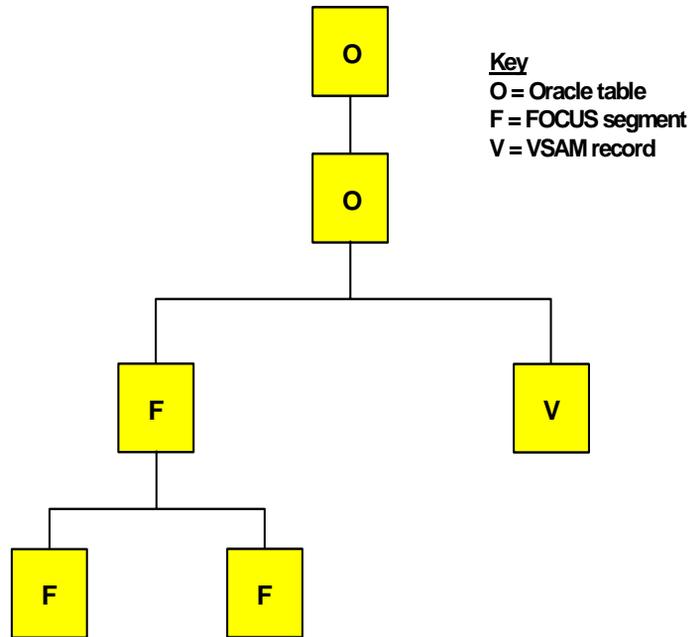
```
JOIN SUBPART IN AIRCRAFT TO PART IN AIRCRAFT AS SUB_PART
```

This produces the following data structure:



Relating Segments From Different Types of Data Sources

The JOIN command enables you to join segments from different types of data sources, creating temporary data structures that contain related information from otherwise incompatible sources. For example, you could join two Oracle data sources to a FOCUS data source to a VSAM data source, as illustrated in the following diagram.

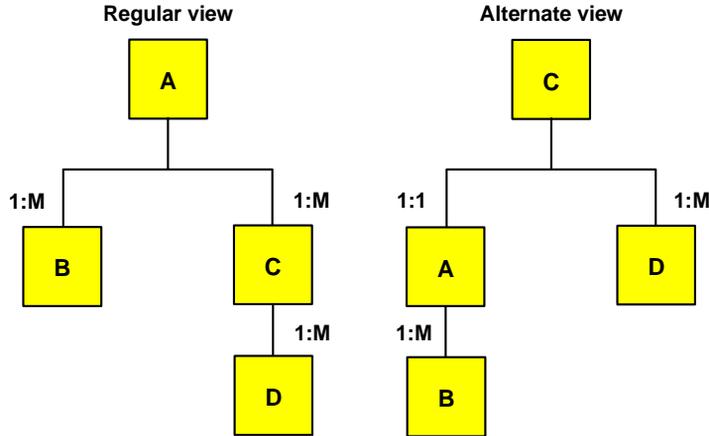


Joins between VSAM and fixed-format data sources are also supported in a Master File, as described in Chapter 7, *Defining a Join in a Master File*.

For detailed information on using the JOIN command with different types of data sources, see the *Creating Reports With WebFOCUS Language* manual.

Rotating a Data Source: An Alternate View

If you are using a network data source or certain hierarchical data sources such as FOCUS, you can rotate the data source after you have described it. This creates an alternate view that changes some of the segment relationships and enables you to access the segments in a different order. Customizing the access path in this way makes it more easily accessible for a given application.



You can even join hierarchical and/or network data sources together and then create an alternate view of the joined structure, selecting the new root segment from the host data source.

Using an alternate view can be very helpful when you want to generate a report using record selection criteria based on fields found in a lower segment (such as segment C in the previous diagram). You could report from an alternate view that makes this the root segment; FOCUS will begin its record selection based on the relevant segment, and avoid reading irrelevant ancestral segments.

When you report from a data source using an alternate view, the data is accessed more efficiently if both of the following conditions are satisfied:

- The field on which the alternate view is based is indexed. For FOCUS data sources, the alternate view field must include INDEX = I in the Master File.
- You use the field in a record selection test, using the WHERE or IF phrases, and make the selection criteria an equality or range test.

An alternate view can be requested on any segment in a data source, except a cross-referenced segment. You request an alternate view with the TABLE command by naming a field from the segment you wish to view as the new root segment. This field may not be a virtual field. The only restriction on requesting an alternate view is that the field on which it is requested must be a real field in the data source.

Syntax

How to Specify an Alternate View

To specify an alternate view, simply append a field name to the file name in the reporting command, using the syntax:

```
TABLE FILE filename.fieldname
```

where:

filename

Is the name of the data source on which you are defining the alternate view.

fieldname

Is a field located in the segment that you are defining as the alternate root. The field must be a real field, not a temporary field defined with the DEFINE attribute or the DEFINE or COMPUTE commands.

If the field is declared in the Master File with the FIELDTYPE attribute set to I, and you use the alternate view in a report, you must use the field in an equality selection test (such as EQ) or range test.

Example

Specifying an Alternative View

To report from the EMPLOYEE data source using an alternate view that makes the DEDUCT segment an alternate root, you could issue the following TABLE FILE command:

```
TABLE FILE EMPLOYEE.DED_CODE
```

CHAPTER 4

Describing an Individual Field

Topics:

- Field Characteristics
- The Field's Name: FIELDNAME
- The Field's Synonym: ALIAS
- The Displayed Data Type: USAGE
- The Stored Data Type: ACTUAL
- Null or MISSING Values: MISSING
- Defining a Dimension: WITHIN
- Validating Data: ACCEPT
- Specifying Acceptable Values for a Dimension
- Alternative Report Column Titles: TITLE
- Documenting the Field: DESCRIPTION
- Describing a Virtual Field: DEFINE

A field is the smallest meaningful element of data in a data source, but it can exhibit a number of complex characteristics. Master File attributes are used to describe these characteristics.

Field Characteristics

The Master File describes the following field characteristics:

- The name of the field described by the FIELDNAME attribute.
- Another name for the field—either its original name as defined to its native data management system, or (for some types of data sources) a synonym of your own choosing, or (in some special cases) a pre-defined value that tells how to interpret the field—that you can use as an alternative name in requests. This alternative name is defined by the ALIAS attribute.
- How the field stores and displays data, specified by the ACTUAL, USAGE, and MISSING attributes.

The ACTUAL attribute describes the type and length of the data as it is actually stored in the data source. For example, a field might be alphanumeric and 15 characters in length. Note that FOCUS data sources do not use the ACTUAL attribute, and instead use USAGE to describe the data both as it is stored in the data source and as it is formatted, since these are identical.

The USAGE attribute, which is also known by its alias, FORMAT, describes how you want a field to be formatted when it displays in reports. You can also specify edit options such as date formats, floating dollar signs, and zero suppression.

The MISSING attribute enables null values to be entered into and read from a field in data sources that support null data, such as FOCUS data sources and most relational data sources.

- The option that a field is virtual—that is, not stored in the data source—and has its value derived from information already in the data source. Virtual fields are specified by the DEFINE attribute.
- Optional field documentation for the developer, contained in the DESCRIPTION attribute.
- Acceptable data-entry values for the field, specified by the ACCEPT attribute.
- An alternative report column title for the field, described by the TITLE attribute.
- A 100-year window that assigns a century value to a two-digit year stored in the field. Two attributes define this window, DEFCENT and YRTHRESH. For detailed information, see the *Developing Reporting Applications* manual.

The Field's Name: FIELDNAME

You identify a field using FIELDNAME, the first attribute specified in a field declaration in the Master File. You can assign any name to a field, regardless of its name in its native data source. Likewise, for FOCUS data sources, you can assign any name to a field in a new data source. The name of a field should be unique within the data source.

Your reporting applications may influence your choice of field name. When you generate a report, each column title in the report defaults to the name of the field displayed in that column, so it will help report readers if you assign meaningful field names. Of course, you do not need to rely upon this default. You can specify a different column title within a given report by using the AS phrase in that report request—as described in the *Creating Reports With WebFOCUS Language* manual—or a different default column title for all reports by using the TITLE attribute in the Master File, as described in *Alternative Report Column Titles: TITLE* on page 4-2.

Syntax

How to Identify the Field Name

`FIELD[NAME] = field_name`

where:

`field_name`

Is the name you want to use to identify this field. It can be a maximum of 66 characters. Some restrictions apply to names longer than 12 characters, as described below. The name can include any combination of letters, digits, and underscores (_), and should begin with a letter. Other characters are not recommended and may cause problems in some operating environments or when resolving expressions.

It is recommended that you not use field names of the type Cn, En, and Xn (where n is any sequence of one or two digits) because these can be used to refer to report columns, HOLD file fields, and other special objects.

If you need to use special characters because of a field's report column title, consider using the TITLE attribute in the Master File to specify the title, as described in *Alternative Report Column Titles: TITLE* on page 4-2.

Reference

Usage Notes for FIELDNAME

Note the following rules when using FIELDNAME:

- **Alias.** FIELDNAME has an alias of FIELD.
- **Changes.** In a FOCUS data source, if the INDEX attribute has been set to I—that is, if an index has been created for the field—you cannot change the field name. In all other situations you can change the field name.

Using a Long and Qualified Field Name

In Master Files, field names and aliases can have a maximum length of 66 characters. However, before defining a field name longer than 48 characters, you must consider how the name will be referenced in requests.

Requests can qualify all referenced field names and aliases with file and/or segment names. This technique is useful when duplicate field names exist across segments in a Master File or in data sources that are joined. But, although the qualifiers and qualification characters are valid only in requests, not in Master Files, the 66-character maximum includes any qualifiers and qualification characters used with the field name in requests. Therefore, if you define a 66-character name in the Master File, you cannot use qualifiers with the name in a request.

The maximum of 66 characters includes the name of the field or alias, plus an eight-character maximum for each field qualifier (Master File name and segment name), plus a qualification character (usually a period) for each qualifier. You may use a unique truncation of a 66-character name with a qualifier.

Temporary field names may also contain up to 66 characters. Text fields and indexed fields in Master Files are limited to 12 characters. However, the aliases for text and indexed fields may be up to 66 characters. Field names up to 66 characters are displayed as column titles in TABLE reports if there is no TITLE attribute or AS phrase.

The default value for the SET FIELDNAME command, SET FIELDNAME=NEW, activates long and qualified field names. The syntax is described in the *Developing Reporting Applications* manual.

Syntax

How to Specify a Qualified Field Name in a Request

`[filename.][segname.]fieldname`

where:

`filename`

Is the one- to eight-character name of the Master File or tag name. Tag names are used with the JOIN and COMBINE commands.

`segname`

Is the one- to eight-character name of the segment in which the field resides.

`fieldname`

Is the name of the field.

Example

Qualifying a Field Name

The fully qualified name of the field EMP_ID in the EMPINFO segment of the EMPLOYEE data source is:

`EMPLOYEE.EMPINFO.EMP_ID`

Syntax

How to Change the Qualifying Character

`SET QUALCHAR = qualcharacter`

The period (.) is the default qualifying character. For further information about the SET QUALCHAR command and valid qualifying characters (. : ! % | \) see the *Developing Reporting Applications* manual.

Reference

Restrictions for Long and Qualified Field Names

The following restrictions apply to field names and aliases longer than 12 characters (that is, long names):

- Joins
You cannot use a long name to specify a join:
 - In a JOIN command, you cannot use it for a cross-referenced field in a FOCUS data source.
 - In a multi-table Master File for a relational data source, you cannot use it for the KEYFLD and IXFLD attributes in the Access File.
- Indexed fields and text fields in FOCUS data sources cannot have field names longer than 12 characters. They can have long ALIAS names.
- The SQL Translator supports field names up to 48 characters.
- A field name specified in an alternate file view cannot be long or qualified.
- CHECK FILE
The CHECK FILE command's PICTURE and HOLD options display the first 11 characters of long names within the resulting diagram or HOLD file. A caret (>) in the 12th position indicates that the name is longer than the displayed portion.
- ?FF, ? HOLD, ? DEFINE
These display up to 31 characters of the name and display a caret (>) in the 32nd character to indicate a longer field name.

Using a Duplicate Field Name

Field names are considered duplicates when you can reference two or more fields with the same field name or alias. Duplication may occur:

- If a name appears multiple times within a Master File.
- In a JOIN between two or more Master Files, or in a recursive JOIN.
- If you issue a COMBINE and do not specify a prefix.

Duplicate fields (those having the same field name and alias) are not allowed in the same segment. The second occurrence is never accessed, and the following message is generated when you issue CHECK and CREATE FILE:

```
(FOC1829) WARNING. FIELDNAME IS NOT UNIQUE WITHIN A SEGMENT: fieldname
```

Duplicate field names may exist across segments in a Master File. To retrieve such a field, you must qualify its name with the segment name in a request. If a field that appears multiple times in a Master File is not qualified in a request, the first field encountered in the Master File is retrieved.

Reports can include qualified names as column titles. The SET QUALTITLES command, discussed in the *Developing Reporting Applications* manual, determines whether reports display qualified column titles for duplicate field names. With SET QUALTITLES=ON, reports display qualified column titles for duplicate field names even when the request itself does not specify qualified names. The default value, OFF, disables qualified column titles.

Rules for Evaluating a Qualified Field Name

The following rules are used to evaluate qualified field names:

- The maximum field name qualification is filename.segmentname.fieldname. For example:

```
TABLE FILE EMPLOYEE
PRINT EMPLOYEE.EMPINFO.EMP_ID
END
```

includes EMP_ID as a fully qualified field. The file name, EMPLOYEE, and the segment name, EMPINFO, are the field qualifiers.

Qualifier names can also be duplicated. For example:

```
FILENAME=CAR, SUFFIX=FOC
SEGNAME=ORIGIN, SEGTYPE=S1
FIELDNAME=COUNTRY, COUNTRY, A10, $
SEGNAME=COMP, SEGTYPE=S1, PARENT=ORIGIN
FIELDNAME=CAR, CARS, A16, $
.
.
.
TABLE FILE CAR
PRINT CAR.COMP.CAR
END
```

This request prints the field with alias CARS. Both the file name and field name are CAR.

- A field name can be qualified with a single qualifier, either its file name or its segment name. For example:

```
FILENAME=CAR, SUFFIX=FOC
SEGNAME=ORIGIN, SEGTYPE=S1
FIELDNAME=COUNTRY, COUNTRY, A10, $
SEGNAME=COMP, SEGTYPE=S1, PARENT=ORIGIN
FIELDNAME=CAR, CARS, A16, $
.
.
.
TABLE FILE CAR
PRINT COMP.CAR AND CAR.CAR
END
```

This request prints the field with alias CARS twice.

When there is a single qualifier, segment name takes precedence over file name. Therefore, if the file name and segment name are the same, the field qualified by the segment name is retrieved.

- If a field name begins with characters that are the same as the name of a prefix operator, it may be unclear whether a request is referencing that field name or a second field name prefixed with the operator. The value of the first field is retrieved, not the value calculated by applying the prefix operator to the second field. In the next example, there is a field whose unqualified field name is CNT.COUNTRY and another whose field name is COUNTRY:

```
FILENAME=CAR, SUFFIX=FOC
  SEGNAME=ORIGIN, SEGTYPE=S1
    FIELDNAME=CNT.COUNTRY, ACNTRY, A10, $
    FIELDNAME=COUNTRY, BCNTRY, A10, $
```

```
TABLE FILE CAR
SUM CNT.COUNTRY
END
```

In this request, the string CNT.COUNTRY is interpreted as a reference to the field named CNT.COUNTRY, not as a reference to the prefix operator CNT. applied to the field named COUNTRY. Therefore, the request sums the field whose alias is ACNTRY. Although the field name CNT.COUNTRY contains a period as one of its characters, it is an unqualified field name. It is not a qualified name or a prefix operator acting on a field name, neither of which is allowed in a Master File. The request does not count instances of the field whose alias is BCNTRY.

- If a Master File has either a file name or segment name that is the same as a prefix operator, the value of the field within the segment is retrieved in requests, not the value calculated by applying the prefix operator to the field. For example:

```
FILENAME=CAR, SUFFIX=FOC
  SEGNAME=ORIGIN, SEGTYPE=S1
    FIELDNAME=COUNTRY, COUNTRY, A10, $
  SEGNAME=PCT, SEGTYPE=S1, PARENT=ORIGIN
    FIELDNAME=CAR, CARS, I2, $
```

```
TABLE FILE CAR
SUM PCT.CAR PCT.PCT.CAR
BY COUNTRY
END
```

This request sums the field with alias CARS first and then the percent of CARS by COUNTRY.

- When a qualified field name can be evaluated as a choice between two levels of qualification, the field name with the higher level of qualification takes precedence. In the following example, the choice is between an unqualified field name (the field named ORIGIN.COUNTRY in the ORIGIN segment) and a field name with segment name qualification (the field named COUNTRY in the ORIGIN segment). The field with segment name qualification is retrieved:

```
FILENAME=CAR, SUFFIX=FOC
  SEGNAME=ORIGIN, SEGTYPE=S1
    FIELDNAME=ORIGIN.COUNTRY, OCNTRY, A10, $
    FIELDNAME=COUNTRY, CNTRY, A10, $
```

```
TABLE FILE CAR
PRINT ORIGIN.COUNTRY
END
```

This request prints the field with alias CNTRY. To retrieve the field with alias OCNTRY, qualify its field name, ORIGIN.COUNTRY, with its segment name, ORIGIN:

```
PRINT ORIGIN.ORIGIN.COUNTRY
```

- When a qualified field name can be evaluated as a choice between two field names with the same level of qualification, the field with the shortest basic field name length is retrieved. For example:

```
FILENAME=CAR, SUFFIX=FOC
  SEGNAME=CAR, SEGTYPE=S1
    FIELDNAME=CAR.CAR, CAR1, A10, $
  SEGNAME=CAR.CAR, SEGTYPE=S1, PARENT=CAR
    FIELDNAME=CAR, CAR2, A10, $
```

```
TABLE FILE CAR
PRINT CAR.CAR.CAR
END
```

In this example, it is unclear if you intend CAR.CAR.CAR to refer to the field named CAR.CAR in the CAR segment or the field named CAR in the CAR.CAR segment. (In either case, the name CAR.CAR is an unqualified name that contains a period, not a qualified name. Qualified names are not permitted in Master Files.)

No matter what the intention, the qualified field name is exactly the same and there is no obvious choice between levels of qualification.

Since the field with alias CAR2 has the shortest basic field name length, CAR2 is printed. This is different from the prior example where the choice was between two levels of qualification. To retrieve the CAR1 field, you must specify its alias.

The Field's Synonym: ALIAS

You can assign every field an alternative name, or alias. A field's alias may be its original name as defined to its native data source, any name of your choosing, or in special cases, a pre-defined value. The way in which you assign the alias is determined by the type of data source and, in special cases, the role the field plays in the data source. Once it has been assigned, you can use this alias in requests as a synonym for the regular field name. You assign this alternative name using the ALIAS attribute.

Example

Using a Field Synonym

In the EMPLOYEE data source, the name CURR_SAL is assigned to a field using the FIELDNAME attribute, and the alternative name CSAL is assigned to the same field using the ALIAS attribute:

```
FIELDNAME = CURR_SAL, ALIAS = CSAL, USAGE = D12.2M, $
```

Both names are equally valid within a request. The following TABLE requests illustrate this—they are functionally identical, refer to the same field, and produce the same result:

```
TABLE FILE EMPLOYEE  
PRINT CURR_SAL BY EMP_ID  
END
```

```
TABLE FILE EMPLOYEE  
PRINT CSAL BY EMP_ID  
END
```

Note: In extract files (HOLD, PCHOLD), the field name is used to identify fields, not the ALIAS.

Implementing a Field Synonym

The value you assign to ALIAS must conform to the same naming conventions to which the FIELDNAME attribute is subject, unless stated otherwise. You assign a value to ALIAS in the following way for the following types of data sources:

- **Relational data sources.** ALIAS describes the field's original column name as defined in the relational table.
- **Sequential data sources.** ALIAS describes a synonym, or alternative name, that you can use in a request to identify the field. You can assign any name as the alias; many users choose a shorter version of the field's primary name—for example, if the field name is LAST_NAME, the alias might be LN. ALIAS is optional.

Note that ALIAS is used in a different way for sequenced repeating fields, where its value is ORDER, as well as for RECTYPE and MAPVALUE fields when the data source includes multiple record types. See Chapter 5, *Describing a Sequential, VSAM, or ISAM Data Source*, for more information about using ALIAS.

- **FOCUS data sources.** ALIAS describes a synonym, or alternative name, that you can use in a request to identify the field. You can assign any name as the alias; many users choose a shorter version of the field's primary name—for example, if the field name is LAST_NAME, the alias might be LN. ALIAS is optional. See Chapter 6, *Describing a FOCUS Data Source*, for more information about using ALIAS. Aliases can be changed without rebuilding the data source. If an alias is referred to in other data sources, similar changes may be needed in those Master Files.

The Displayed Data Type: USAGE

This attribute, which is also known as FORMAT, describes how you want a field to be formatted when it displays in reports or is used in calculations.

For FOCUS data sources, which do not use the ACTUAL attribute, USAGE also specifies how the field is to be stored. For other types of data sources, you will usually want to assign a USAGE value that corresponds to the ACTUAL value, to identify the field as the same data type used to store it in the data source. For instructions on which ACTUAL values correspond to which USAGE values, see the documentation for the specific data adapter. For sequential data sources, see Chapter 5, *Describing a Sequential, VSAM, or ISAM Data Source*. For other types of data sources, see the iWay documentation for your server.

In addition to selecting the data type and length, you can also specify display options such as date formatting, floating dollar signs, and zero suppression. You can use these options to customize how the field is displayed in reports.

Syntax

How to Specify a Display Format

`USAGE = t1[d]`

where:

t

Is the data type. Valid values are A (alphanumeric), F (floating-point single-precision), D (floating-point double-precision), I (integer), P (packed decimal), D, W, M, Q, or Y used in a valid combination (date), and TX (text).

l

Is a length specification. Different data types have different length specifications. See the section for each data type for more information. Note that you do not specify a length for date format fields.

d

Is one or more display options. Different data types offer different display options. See the section for each data type for more information.

The complete USAGE value cannot exceed eight characters.

The values that you specify for type and field length determine the number of print positions allocated for displaying or storing the field. Display options only affect displayed or printed fields. They are not active for non-display retrievals, such as extract files.

Note: If a numeric field cannot be displayed with the USAGE format given (for example, the result of aggregation is too large) asterisks are displayed.

Examples and additional information about each format type are provided in the section for that type.

Reference

Usage Notes for USAGE

Note the following rules when using USAGE:

- **Alias.** USAGE has an alias of FORMAT.
- **Changes.** For most data sources, you can change the type and length specifications of USAGE only to other types and lengths valid for that field's ACTUAL attribute. You can change display options at any time.

For FOCUS data sources, you cannot change the type specification. You can change the length specification for I, F, D, and P fields, because this affects only display, not storage. You cannot change the decimal part of the length specification for P fields. You can change the length specification of A (alphanumeric) fields only if you use the REBUILD facility. You can change display options at any time.

Data Type Formats

You can specify several types of formats:

- **Numeric.** There are four types of numeric formats: integer, floating-point single-precision, floating-point double-precision, and packed decimal. See *Numeric Display Options* on page 4-2 for additional information about numeric formats.
- **Alphanumeric.**
- **Date.** The date format enables you to define date components such as year, quarter, month, day, and day of week; to sort by date; to do date comparisons and arithmetic with dates; and to automatically validate dates in transactions. Note that for some applications, such as assigning a date value using the DECODE function, you may wish to instead use alphanumeric, integer, or packed-decimal fields with date display options which provide partial date functionality.
- **Text.**

Integer Format

You can use integer format for whole numbers—that is, any value composed of the digits zero to nine, without a decimal point.

You can also use integer fields with date display options to provide limited date support. This use of integer fields is described in the *Alphanumeric and Numeric Formats with Date Display Options* on page 4-2.

The integer USAGE type is I. Display options are described in *Numeric Display Options* on page 4-2. The format of the length specification is

n

where:

n

Is the maximum number of digits. The maximum integer size is 10 digits with I11 reserved for a negative sign. The maximum integer value displayed is 2147483647.

For example:

Format	Display
I6	4316
I2	22
I4	-617

Floating-Point Double-Precision Format

You can use floating-point double-precision format for any number, including numbers with decimal positions—that is, for any value composed of the digits zero to nine and an optional decimal point.

The floating-point double-precision USAGE type is D. The compatible display options are described in *Numeric Display Options* on page 4-2. The length specification format is

`t[.s]`

where:

`t`

Is the maximum number of characters to be displayed, up to a maximum of 16, including digits, a leading minus sign if the field will contain any negative values, and an optional decimal point if you want one to be displayed.

`s`

Is the number of digits that will follow the decimal point.

For example:

Format	Display
<code>D8.2</code>	3,187.54
<code>D8</code>	416

In the case of D8.2, the 8 represents the maximum number of places including the decimal point and decimal places. The 2 represents how many of these eight places are decimal places. The commas are automatically included in the display, and are not counted in the total.

Floating-Point Single-Precision Format

You can use floating-point single-precision format for any number, including numbers with decimal positions—that is, for any value composed of the digits 0 to 9, including an optional decimal point. This format is intended for use with smaller decimal numbers. Unlike floating-point double-precision format, its length cannot exceed nine positions.

The floating-point single-precision USAGE type is F. The compatible display options are described in *Numeric Display Options* on page 4-2. The length specification format is

`t[.s]`

where:

`t`

Is the maximum number of characters to be displayed, up to a maximum of 9, including digits, a leading minus sign if the field will contain any negative values, and an optional decimal point if you want one to be displayed.

`s`

Is the number of digits that will follow the decimal point.

For example:

Format	Display
F5.1	614.2
F4	318

Packed-Decimal Format

You can use packed-decimal format for any number, including decimal numbers—that is, for any value composed of the digits zero to nine, including an optional decimal point.

You can also use packed-decimal fields with date display options to provide limited date support. This use of packed-decimal fields is described in *Alphanumeric and Numeric Formats with Date Display Options* on page 4-2.

The packed-decimal USAGE type is P. The compatible display options are described in *Numeric Display Options* on page 4-2.

The length specification format is

m.n

where:

m

Is the maximum number of characters to be displayed, up to a maximum of 33 positions (which include a position for the sign and decimal point).

n

Is the number of digits that will follow the decimal point. It can be up to 31 digits.

For example:

Format	Display
P9.3	4168.368
P7	617542

Numeric Display Options

Display options may be used to edit numeric formats in various ways. Display options affect only how the data in the field is printed or displays on the screen. Display options do not affect how the data is stored in your data source.

Edit Option	Meaning	Effect
%	Percent sign	Displays a percent sign along with numeric data. Does not calculate the percent.
B	Bracket negative	Encloses negative numbers in parentheses.
c	Comma suppress	Suppresses the display of commas. Used with numeric format options M and N (floating and non-floating dollar sign) and data format D (floating-point double-precision).
C	Comma edit	Inserts a comma after every third significant digit, or a period instead of a comma if continental decimal notation is in use.
DMY	Day-Month-Year	Displays alphanumeric or integer data as a date in the form day/month/year.
E	Scientific notation	Displays only significant digits.
L	Leading zeroes	Adds leading zeroes.
M	Floating \$	Places a floating dollar sign \$ to the left of the highest significant digit.
MDY	Month-Day-Year	Displays alphanumeric or integer data as a date in the form month/day/year.
N	Fixed \$	Places a dollar sign \$ to the left of the field.
R	Credit (CR) negative	Places CR after negative numbers.
S	Zero suppress	If the data value is zero, prints a blank in its place.
T	Month translation	Displays the month as a three-character abbreviation.
YMD	Year-Month-Day	Displays alphanumeric or integer data as a date in the form year/month/day.

Example **Using Numeric Display Options**

The following table shows examples of the display options that are available for numeric fields.

Option	Format	Data	Display
Percent sign	I2%	21	21%
	D7%	6148	6,148%
	F3.2%	48	48.00%
Comma suppression	D6c	41376	41376
	D7Mc	6148	\$6148
	D7Nc	6148	\$ 6148
Comma inclusion	I6C	41376	41,376
Zero suppression	D6S	0	
Bracket negative	I6B	-64187	(64187)
Credit negative	I8R	-3167	3167 CR
Leading zeroes	F4L	31	0031
Floating dollar	D7M	6148	\$6,148
Non-floating dollar	D7N	5432	\$ 5,432
Scientific notation	D12.5E	1234.5	0.123456D+04
Year/month/day	I6YMD	980421	98/04/21
	I8YYMD	19980421	1998/04/21
Month/day/year	I6MDY	042198	04/21/98
	I8MDYY	04211998	04/21/1998
Day/month/year	I6DMY	210498	21/04/98
	I8DMYY	21041998	21/04/1998
Month translation	I2MT	07	JUL

Several display options can be combined, as shown:

Format	Data	Display
I5CB	-61874	(61,874)

All of the options may be specified in any order. Options M and N (floating and non-floating dollar sign) and data format D (floating-point double-precision) automatically invoke option C (comma). Options L and S cannot be used together. Option T (translate month) can be included anywhere in an alphanumeric or integer USAGE specification that includes the M (month) display option. Date display options (D, M, T, and Y), which cannot be used with floating-point fields, are described in *Alphanumeric and Numeric Formats with Date Display Options* on page 4-2.

Rounding

When a value with decimal places is assigned to a numeric field, if there are more decimal places in the value than are specified in the field's length description, the value is rounded to an acceptable size before either storing or displaying it. The value is rounded down when the first extra decimal digit is less than five, and rounded up when it is five or greater (although an additional consideration is introduced for floating-point values).

For example, consider a packed-decimal field with two decimal places

```
FIELDNAME = PRESSURE, FORMAT = P8.2, $
```

to which you assign a value with *four* decimal places:

```
PRESSURE = 746.1289
```

The first extra digit—that is, the first one past the specified length of two decimal places—is 8. Since 8 is greater than or equal to five, the value is rounded up, and PRESSURE becomes:

```
746.13
```

The details of rounding is handled in the following way for the following numeric formats:

- **Integer format.** When a value with decimal places is assigned to an integer field, the value is rounded before it is stored. If the value is assigned using a DEFINE or COMPUTE command, the decimal portion of the value is truncated before it is stored.
- **Packed-decimal format.** When a value is assigned to a packed-decimal field, and the value has more decimal places than the field's format specifies, the value is rounded before it is stored.

- **Floating-point single- and double-precision formats.** When a value is assigned to one of these fields, and the value has more decimal places than the field's format specifies, the full value is stored in the field (up to the limit of precision determined by the field's internal storage type). When this value is later displayed, however, it is rounded.

Note that if the decimal portion of a floating-point value *as it is internally represented in hexadecimal floating-point notation* is repeating—that is, non-terminating—the repeating hexadecimal number is resolved as a non-repeating slightly lower number, and this lower number is stored as the field's value. In these situations, if in the original value of the digit to be rounded had been a five (which would be rounded up), in the stored lower value it would become a four (which is rounded down).

For example, consider a floating-point double-precision field with one decimal place

```
FIELDNAME = VELOCITY, FORMAT = D5.1, $
```

to which you assign a value with *two* decimal places:

```
VELOCITY = 1.15
```

This value is stored as a slightly smaller number due to the special circumstances of floating-point arithmetic, as previously described:

```
1.149999
```

While the original number, 1.15, would have been rounded upward to 1.2 (since the first extra digit was 5 or greater), the number as stored is slightly less than 1.15 (1.149999) and, as the first extra digit is now less than 5 (4 in this case), it is rounded down to 1.1. To summarize the process:

```
format:    D5.1
entered:   1.15
stored:    1.149999
rounded:   1.1
displayed: 1.1
```

Alphanumeric Format

You can use alphanumeric format for any value to be interpreted as a sequence of characters and composed of any combination of digits, letters, and other characters.

You can also use alphanumeric fields with date display options to provide limited date support. This use of alphanumeric fields is described in *Alphanumeric and Numeric Formats with Date Display Options* on page 4-2.

The alphanumeric USAGE type is A. The format of the length specification is n, where n is the maximum number of characters in the field, up to 256 characters.

For example:

Format	Display
A115	The minutes of today's meeting were submitted...
A2	B3
A24	127-A429-BYQ-49

The standard numeric display options are not available for the alphanumeric data format. However, alphanumeric data can be printed under the control of a pattern that is supplied at run time. For instance, if a product code is to be displayed in parts, with each part separated by a “-“, the following could be included in a DEFINE command:

```
PRODCODE/A11 = EDIT (fieldname, '999-999-999') ;
```

where:

fieldname

Is the existing field name, not the newly defined field name.

If the value is 716431014, PRODCODE will be displayed as 716-431-014. See the *Creating Reports With WebFOCUS Language* manual for more information.

Date Formats

Date format enables you to define a field as a date and manipulate the field's value and display that value in ways appropriate to a date. Using date format, you can:

- Define date components such as year, quarter, month, day, and day of week, and extract them easily from date fields.
- Sort reports into date sequence, regardless of how the date is displayed.
- Perform arithmetic with dates and compare dates without resorting to special date-handling functions.
- Refer to dates in a natural way, such as JAN 1 1995, without regard to display or editing formats.
- Automatically validate dates in transactions.

Date Display Options

The date format does not specify type or length. Instead, it specifies date component options (D, W, M, Q, Y, and YY) and display options. These options are shown in the following chart.

Display Option	Meaning	Effect
D	Day	Prints a value from 1 to 31 for the day.
M	Month	Prints a value from 1 to 12 for the month.
Y	Year	Prints a two-digit year.
YY	Four-digit year	Prints a four-digit year.
T	Translate month or day	Prints a three-letter abbreviation for months in uppercase, if M is included in the USAGE specification.
t	Translate month or day	Functions the same as uppercase T (described above), except that the first letter of the month or day is uppercase and the following letters are lowercase.*
TR	Translate month or day	Functions the same as uppercase T (described above), except that the entire month or day name is printed instead of an abbreviation.
tr	Translate month or day	Functions the same as lowercase t (described above), except that the entire month or day name is printed instead of an abbreviation.*
Q	Quarter	Prints the quarter (1 - 4 if Q is specified by itself, or Q1 - Q4 if it is specified together with other date format items such as Y).
W	Day-of-Week	If it is included in a USAGE specification with other date component options, prints a three-letter abbreviation of the day of the week in uppercase. If it is the only date component option in the USAGE specification, it prints the number of the day of the week (1-7, Mon=1).
w	Day-of-Week	Functions the same as uppercase W (described above), except that the first letter is uppercase and the following letters are lowercase.*

Display Option	Meaning	Effect
WR	Day-of-Week	Functions the same as uppercase W (described above), except that the entire day name is printed instead of an abbreviation.*
wr	Day-of-Week	Functions the same as lowercase w (described above), except that the entire day name is printed instead of an abbreviation.*
JUL	Julian format	Prints date in Julian format.
YYJUL	Julian format	Prints a Julian format date in the format YYYYDDD. The 7-digit format displays the four-digit year and the number of days counting from January 1. For example, January 3, 2001 in Julian format is 2001003.

***Note:** When using these display options, be sure they are actually stored in the Master File as lowercase letters.

The following combinations of date components are not supported in date formats:

I2D, A2D, I2M, A2M, I2MD, A2MD

Reference

How Field Formats Y, YY, M, and W Are Stored

The Y, YY, and M formats are not smart dates. Smart date formats YMD and YYMD, are stored as an offset from the base date of 12/31/1900. Smart date formats YM, YQ, YYM, and YYQ are stored as an offset from the base date 01/1901. W formats are stored as integers with a display length of one, containing values 1-7 representing the days of the week. Y, YY, and M formats are stored as integers. Y and M have display lengths of two. YY has a display length of four. When using Y and YY field formats, keep in mind these two important points:

- The Y formats will not sort based on DEFCENT and YRTHRESH settings. A field with a format of Y will not equal a YY field, as this is not a displacement, but a 4-digit integer.
- It is possible to use DEFCENT and YRTHRESH to convert a field from Y to YY format.

Reference

Date Literals Interpretation Table

This table illustrates the behavior of date formats. The columns indicate the number of input digits for a date format. The rows indicate the usage or format of the field. The intersection of row and column describes the result of input and format.

Date Format	1	2	3	4
YYMD	*	*	CC00/0m/dd	CC00/mm/dd
MDYY	*	*	*	*
DMYY	*	*	*	*
YMD	*	*	CC00/0m/dd	CC00/mm/dd
MDY	*	*	*	*
DMY	*	*	*	*
YYM	CC00/0m	CC00/mm	CC0y/mm	CCyy/mm
MYY	*	*	*	*
YM	CC00/0m	CC00/mm	CC0y/mm	CCyy/mm
MY	*	*	0m/CCyy	mm/CCyy
M	0m	mm	*	*
YYQ	CC00/q	CC0y/q	CCyy/q	0yyy/q
QYY	*	*	q/CCyy	*
YQ	CC00/q	CC0y/q	CCyy/q	0yyy/q
QY	*	*	q/CCyy	*
Q	q	*	*	*
JUL	CC00/00d	CC00/0dd	CC00/ddd	CC0y/ddd
YY	000y	00yy	0yyy	yyyy
Y	0y	yy	*	*
D	0d	dd	*	*
W	w	*	*	*

Date Format	5	6	7	8
YYMD	CC0y/mm/dd	CCyy/mm/dd	0yyy/mm/dd	yyyy/mm/dd
MDYY	0m/dd/CCyy	mm/dd/CCyy	0m/dd/yyyy	mm/dd/yyyy
DMYY	0d/mm/CCyy	dd/mm/CCyy	0d/mm/yyyy	dd/mm/yyyy
YMD	CC0y/mm/dd	CCyy/mm/dd	0yyy/mm/dd	yyyy/mm/dd
MDY	0m/dd/CCyy	mm/dd/CCyy	0m/dd/yyyy	mm/dd/yyyy
DMY	0d/mm/CCyy	dd/mm/CCyy	0d/mm/yyyy	dd/mm/yyyy
YYM	0yyy/mm	yyyy/mm	*	*
MYY	0m/yyyy	mm/yyyy	*	*
YM	0yyy/mm	yyyy/mm	*	*
MY	0m/yyyy	mm/yyyy	*	*
M	*	*	*	*
YYQ	yyyy/q	*	*	*
QYY	q/yyyy	*	*	*
YQ	yyyy/q	*	*	*
QY	q/yyyy	*	*	*
Q	*	*	*	*
JUL	CCyy/ddd	*	*	*
YY	*	*	*	*
Y	*	*	*	*
D	*	*	*	*
W	*	*	*	*

Note:

- CC stands for two century digits provided by DFC/YRT settings.
- * stands for message FOC177 (invalid date constant).
- Date literals are read from right to left.

Controlling the Date Separator

You can control the date separators when the date is displayed. In basic date format, such as YMD and MDYY, the date components are displayed separated by a slash character (/). The same is true for the year-month format. Year-quarter format is displayed with the year and quarter separated by a blank (for example, 94 Q3 or Q3 1994). The single component formats display just the single number or name.

The separating character can be changed to a period, a dash, or a blank, or can even be eliminated entirely. The following table shows the USAGE specifications that you can use to change the separating character.

Format	Display
YMD	93/12/24
Y.M.D	93.12.24
Y-M	93-12
YBMBD	93 12 24 (The letter B signifies blank spaces.)
Y M D	931224 (The concatenation symbol () eliminates the separation character.)

Date Translation

Numeric months and days can be replaced by a translation, such as JAN, January, Wed, or Wednesday. The translated month or day can be abbreviated to three characters or fully spelled out. It can appear in either uppercase or lowercase. In addition, the day of the week (for example, Monday) can be appended to the beginning or end of the date. All of these options are independent of each other.

Translation	Display
MT	JAN
Mt	Jan
MTR	JANUARY
Mtr	January
WR	MONDAY
wr	Monday

Example Using a Date Format

The following chart shows some sample USAGE and ACTUAL formats for data stored in a non-FOCUS data source. The Value column shows the actual data value and the Display column shows how the data is displayed.

USAGE	ACTUAL	Value	Display
wrMtrDYY	A6YMD	990315	Monday, March 15, 1999
YQ	A6YMD	990315	99 Q1
QYY	A6YMD	990315	Q1 1999
YMD	A6	990315	99/03/15
MDYY	A6YMD	990315	03/15/1999

Note that the date attributes in the ACTUAL format specify the order in which the date is stored in the non-FOCUS data source. If the ACTUAL format does not specify the order of the month, day, and year, it will be inferred from the USAGE format.

Using a Date Field

A field formatted as a date is automatically validated when it is entered. It can be entered as a natural date literal (for example, JAN 12 1999) or as a numeric date literal (for example, 011299).

Natural date literals, by including spaces between date components and using abbreviations of month names, enable you to specify a date in a natural, easily understandable way. For example, April 25, 1999 can be specified as any of the following natural date literals:

```
APR 25 1999
25 APR 1999
1999 APR 25
```

Natural date literals can be used in all date computations and all methods of data source updating. Examples are shown in the following chart.

In WHERE screening	WHERE MYDATE IS 'APR 25 1999'
In arithmetic expressions	MYDATE - '1999 APR 25'
In computational date comparisons	IF MYDATE GT '25 APR 1999'
In comma-delimited data	...,MYDATE = APR 25 1999, ...

The following chart describes the format of natural date literals.

Literal	Format
Year-month-day	Four-digit year; uppercase three-character abbreviation, or uppercase full name, of the month; and one- or two-digit day of the month (for example, 1999 APR 25 or APRIL 25 1999).
Year-month	Year and month as described above.
Year-quarter	Year as described above, Q plus quarter number for quarter (for example, 1999 Q3).
Month	Month as described above.
Quarter	Quarter as described above.
Day of week	Three-character, uppercase abbreviation, or full, uppercase name, of the day (for example, MON or MONDAY).

The date components of a natural date literal can be specified in any order, regardless of their order in the USAGE specification of the target field. Date components are separated by one or more blanks.

For example, if a USAGE specification for a date field is YM, a natural date literal written to that field can include the year and month in any order. MAY 1999 and 1990 APR would both be valid literals.

Numeric Date Literals

Numeric date literals differ from natural date literals in that they are simple strings of digits. The order of the date components in a numeric date literal must match the order of the date components in the corresponding USAGE specification. In addition, the numeric date literal must include all of the date components included in the USAGE specification. For example, if the USAGE specification is DMY, then April 25 1999 must be represented as:

250499

Numeric date literals can be used in all date computations and all methods of data source updating.

Date Fields in Arithmetic Expressions

The general rule for manipulating date fields in arithmetic expressions is that date fields in the same expression must specify the same date components. The date components can be specified in any order, and display options are ignored. Valid date components are Y or YY, Q, M, W, and D.

Note that arithmetic expressions assigned to quarters, months, or days of the week are computed modulo 4, 12, and 7, respectively, so that anomalies like fifth quarters and thirteenth months are avoided.

For example, if NEWQUARTER and THISQUARTER both have USAGE specifications of Q, and the value of THISQUARTER is 2, then the following statement

```
NEWQUARTER = THISQUARTER + 3
```

gives NEWQUARTER a value of 1 (that is, the remainder of 5 divided by 4).

Converting a Date Field

Two types of conversion are possible: format conversion and date component conversion. In the first case, the value of a date format field can be assigned to an alphanumeric or integer field that uses date display options (see the following section); the reverse conversion is also possible.

In the second case, a field whose USAGE specifies one set of date components can be assigned to another field specifying different date components.

For example, the value of REPORTDATE (DMY) can be assigned to ORDERDATE (Y); in this case, the year is being extracted from REPORTDATE. If REPORTDATE is Apr 27 99, ORDERDATE is 99.

You can also assign the value of ORDERDATE to REPORTDATE; if the value of ORDERDATE is 99, the value of REPORTDATE would be Jan 1 99. In this case, REPORTDATE is given values for the missing date components.

Syntax

How to Convert a Date Field

```
field1/format = field2;
```

where:

field1

Is a date format field, or an alphanumeric or integer format field using date display options.

format

Is the USAGE (or FORMAT) specification of *field1* (the target field).

field2

Is a date format field, or an alphanumeric or integer format field using date display options. The format types (alphanumeric, integer, or date) and the date components (YY, Y, Q, M, W, D) of *field1* and *field2* do not need to match.

How a Date Field Is Represented Internally

Date fields are represented internally as four-byte binary integers indicating the elapsed time since the date format base date. For each field, the unit of elapsed time is that field's smallest date component.

For example, if the USAGE specification of REPORTDATE is MDY, then elapsed time is measured in days, and internally the field contains the number of days elapsed between the entered date and the base date. If you entered the numeric literal for February 13, 1964 (that is, 021364), and then printed the field in a report, 02/13/64 would be displayed. If you used it in the equation

```
NEWDATE = 'FEB 28 1964' - REPORTDATE ;  
DAYS/D = NEWDATE ;
```

then the value of DAYS would be 15. However, the internal representation of REPORTDATE would be a four-byte binary integer representing the number of days between December 31, 1900 and February 13, 1964.

Just as the unit of elapsed time is based on a field's smallest date component, so too is the base date. For example, for a YQ field, elapsed time is measured in quarters and the base date is the first quarter of 1901. For a YM field, elapsed time is measured in months and the base date is the first month of 1901.

In reports, to display blanks or the actual base date, use the SET DATEDISPLAY command described in the *Developing Reporting Applications* manual. The default value, OFF, displays blanks when a date matches the base date. ON displays the actual base date value.

You do not need to be concerned with the date format's internal representation, except to note that all dates set to the base date display as blanks, and all date fields that are entered blank or as all zeroes are accepted during validation and interpreted as the base date. They will be displayed as blanks, but will be interpreted in date computations and expressions as the base date.

Displaying a Non-Standard Date Format

By default, if a date field in a non-FOCUS data source contains an invalid date, a message displays and the entire record fails to display in a report. For example, if a date field contains '980450' with an ACTUAL of A6 and a USAGE of YMD, the record containing that field will not display. The SET ALLOWCVTERR command enables you to display the rest of the record that contains the incorrect date.

Syntax

Invoking ALLOWCVTERR

```
SET ALLOWCVTERR = {ON|OFF}
```

where:

ON

Allows the display of a field containing an incorrect date.

OFF

Generates a diagnostic message if incorrect data is encountered, and does not display the record containing the bad data. This is the default value.

When a bad date is encountered, ALLOWCVTERR sets the value of the field to either MISSING or to the base date depending on whether MISSING=ON.

The following chart shows the results of interaction between DATEDISPLAY and MISSING assuming ALLOWCVTERR=ON and the presence of a bad date.

	MISSING=OFF	MISSING=ON
DATEDISPLAY=ON	Displays Base Date 19001231 or 1901/1	.
DATEDISPLAY=OFF	Displays Blanks	.

DATEDISPLAY affects only how the base date is displayed. See the *Developing Reporting Applications* manual for a description of DATEDISPLAY.

Date Format Support

Date format fields are used in special ways with the following facilities:

- **Dialogue Manager.** Amper variables can function as date fields if they are set to natural date literals. For example:

```
-SET &NOW = 'APR 25 1960' ;
-SET &LATER = '1990 25 APR' ;
-SET &DELAY = &LATER - &NOW ;
```

In this case, the value of &DELAY is the difference between the two dates, measured in days: 10,957.

- **Extract files.** Date fields in SAVB and unformatted HOLD files are stored as four-byte binary integers representing the difference between the field's face value and the standard base date. Date fields in SAVE files and formatted HOLD files (for example, USAGE WP) are stored without any display options.
- **GRAPH.** Date fields are not supported as sort fields in ACROSS and BY phrases.
- **FML.** Date fields are not supported within the RECAP statement.

Alphanumeric and Numeric Formats With Date Display Options

In addition to the standard date format, you can also represent a date by using an alphanumeric, integer, or packed-decimal field with date display options (D, M, Y, and T). Note, however, that this does not offer the full date support that is provided by the standard date format.

Alphanumeric and integer fields used with date display options have some date functionality when used with special date functions, as described in the *Creating Reports With WebFOCUS Language* manual.

When representing dates as alphanumeric or integer fields with date display options, you can specify the year, month, and day. If all three of these elements are present, then the date has six digits (or eight if the year is presented as four digits) and the USAGE can be:

Format	Display
I6MDY	04/21/98
I6YMD	98/04/21
P6DMY	21/04/98
I8DMYY	21/04/1998

A month's number (1 to 12) can be translated to the corresponding month name by adding the letter T to the format, immediately after the M. For instance:

Format	Data	Display
I6MTDY	05/21/98	MAY 21 98
I4MTY	0698	JUN 98
I2MT	07	JUL

If the date has only the month element, a format of I2MT will display the value 4 as APR, for example. This is particularly useful in reports where columns or rows are sorted by month. They will then appear in correct calendar order; for example, JAN, FEB, MAR, because the sorting is based on the numerical, not alphabetical, values. (Note that without the T display option, I2M would be interpreted as an integer with a floating dollar sign.)

Date-Time Formats

The date-time data type supports both the date and time, similar to the timestamp data types available in many relational data sources.

Date-time fields are stored in eight or ten bytes, four digits for date and either four or six digits for time, depending on whether the format specifies a microsecond.

See the *Developing Reporting Applications* manual for information on subroutines for manipulating date-time fields.

Describing a Date-Time Field

In a Master File, The USAGE (or FORMAT) attribute determines how date-time field values are displayed in report output and forms, and how they behave in expressions and functions. For FOCUS data sources, it also determines how they are stored.

A new format type, H, describes date-time fields. The USAGE attribute for a date-time field contains the H format code and can identify either the length of the field or the relevant date-time display options.

The MISSING attribute for date-time fields can be ON or OFF. If it is OFF, and the date-time field has no value, it defaults to blank.

Syntax

How to Describe a Date-Time Field

The USAGE attribute can be one of the following:

USAGE = *Hnn*

USAGE = *Htimefmt1*

USAGE = *Hdatefmt [separator] [timefmt2]*

where:

Hnn

Is the USAGE value for a numeric date-time value without date-time display options. This format is appropriate for use in alphanumeric HOLD files or transaction files.

nn is the field length, from 1 to 20, including up to eight characters for displaying the date and up to nine or 12 characters for the time. For lengths less than 20, the date is truncated on the right.

An eight-character date includes four digits for the year, two digits for the month, and two digits for the day of the month, YYYYMMDD.

A nine-character time includes two digits for the hour, two digits for the minute, two digits for the second, and three digits for the millisecond, HHMMSSsss. The millisecond component represents the decimal portion of the second to three places.

A twelve-character time includes two digits for the hour, two digits for the minute, two digits for the second, three digits for the millisecond, and three digits for the microsecond, HHMMSSsssmmm. The millisecond component represents the decimal portion of the second value to three places. The microsecond component represents three additional decimal places beyond the millisecond value.

With this format, there are no spaces between the date and time components, no decimal points, and no spaces or separator characters within either component. The time must be entered using the 24-hour system. For example, the value 19991231225725333444 represents 1999/12/31 10:57:25.333444PM.

Htimefmt1

Is the USAGE format for displaying time only. Hour, minute, and second components are always displayed separated by colons (:), with no intervening blanks.

Unless you specify one of the AM/PM time display options, the time component is displayed using the 24-hour system.

When the format includes more than one time display option:

- The options must appear in the order hour, minute, second, millisecond, microsecond.
- The first option must be either hour, minute, or second.
- No intermediate component can be skipped. That is, if hour is specified the next option must be minute, it cannot be second.

The following table lists the valid time display options for a time-only USAGE attribute. Assume the time value is 2:05:27.123456 a.m.

Option	Meaning	Effect
H	hour (two digits) If the format includes the option a or A, the hour value is from 01 to 12. Otherwise, the hour value is from 00 to 23, with 00 representing midnight.	Prints a two-digit hour. For example: USAGE = HH prints 02
h	hour with zero suppression If the format includes the option a or A, the hour value is from 1 to 12. Otherwise, the hour is from 0 to 23.	Displays the hour with zero suppression. For example: USAGE = Hh prints 2
I	minute (two digits) The minute value is from 00 to 59.	Prints the two-digit minute. For example: USAGE = HHI prints 02:05
i	minute with zero suppression The minute value is from 0 to 59.	Prints the minute with zero suppression. Cannot be used together with an hour format (H or h). For example: USAGE = Hi prints 5
S	Second (two digits) 00 to 59	Prints the two-digit second. For example: USAGE = HHIS prints 02:05:27
s	millisecond (three digits — after the decimal point in the second) 000 to 999	Prints the second to three decimal places. For example: USAGE = HHISs prints 02:05:27.123
m	microsecond (three additional digits after millisecond) 000 through 999	Prints the second to six decimal places. For example: USAGE = HSsm prints 27.123456
A	12-hour time display with AM or PM in upper case	Prints the hour from 01 to 12 followed by AM or PM. For example: USAGE = HHISA prints 02:05:27AM
a	12-hour time display with am or pm in lower case	Prints the hour from 01 to 12 followed by am or pm. For example: USAGE = HHISa prints 02:05:27am

Hdatefmt

Is the USAGE format for displaying the date portion of the date-time field.

The date components can be in any of the following combinations and order:

- Year first combinations: Y, YY, YM, YYM, YMD, YYMD
- Month-first combinations: M, MD, MY, MYY, MDY, MDYY
- Day-first combinations: D, DM, DMY, DMY Y

The date format can include the following display options as long as they conform to the allowed combinations. In the following table, assume the date is February 5, 1999.

Option	Meaning	Example
Y	2-digit year	99
YY	4-digit year	1999
M	2-digit month (01 - 12)	02
MT	Full month name	February
Mt	Short month name	Feb
D	2-digit day	05
d	Zero-suppressed day	5
k	For formats in which month or day is followed by year, and month is translated to a short or full name, k separates the year from the day with a comma and blank. Otherwise, the separator is a blank.	USAGE = HMtDkYY prints Feb 05, 1999

separator

Is a separator between the date components. The default separator is a slash (/). Other valid separators are: period (.), hyphen (-), blank (B), or none (N). With translated months, these separators can only be specified when the k option is not used.

timefmt2

Is the format for a time that follows a date. Time is separated from the date by a blank; time components are separated from each other by colons. Unlike the format for time alone, a time format that follows a date format consists of at most two characters: a single character to represent all of the time components to be displayed and, optionally, one character for an AM/PM option.

The following table lists the valid options. Assume the date is February 5, 1999 and the time is 02:05:25.444555 a.m.

Option	Meaning	Example
H	Prints hour	USAGE = HYYMDH prints 1999/02/05 02
I	Prints hour:minute	USAGE = HYYMDI prints 1999/02/05 02:05
S	Prints hour:minute:second	USAGE = HYYMDS prints 1999/02/05 02:05:25
s	Prints hour:minute:second.millisecond	USAGE = HYYMDS prints 1999/02/05 02:05:25.444
m	Prints hour:minute:second.microsecond	USAGE = HYYMDm prints 1999/02/05 02:05:25.444555
A	Prints AM or PM. Uses the 12-hour system and causes the hour to be printed with zero suppression.	USAGE = HYYMDSA prints 1999/02/05 2:05:25AM
a	Prints am or pm. Uses the 12-hour system and causes the hour to be printed with zero suppression.	USAGE = HYYMDSA prints 1999/02/05 2:05:25am

Note: Unless you specify one of the AM/PM time display options, the time component is displayed using the 24-hour system.

Specifying a Date-Time Value

An external date-time value is a constant in character format from one of the following sources:

- A sequential data source.
- Used in an expression in a WHERE, IF, DEFINE, or a COMPUTE.

A date-time constant or a date-time value as it appears in a character file has one of the following formats:

```
time_string [date_string]  
date_string [time_string]
```

A date-time constant in a COMPUTE, DEFINE, or WHERE expression must have one of the following formats:

```
DT(time_string [date_string])  
DT(date_string [time_string])
```

A date-time constant in an IF expression has one of the following formats:

```
'time_string [date_string]'  
'date_string [time_string]'
```

If the value contains no blanks or special characters, the single quotation marks are not necessary. Note that the DT prefix is not supported in IF criteria.

where:

time_string

Cannot contain blanks. Time components are separated by colons and may be followed by AM, PM, am, or pm. For example:

```
14:30:20:99      (99 milliseconds)  
14:30  
14:30:20.99     (99/100 seconds)  
14:30:20.999999 (999999 microseconds)  
02:30:20:500pm
```

Note that the second can be expressed with a decimal point or be followed by a colon.

- If there is a colon after the second, the value following it represents the millisecond. There is no way to express the microsecond using this notation.
- A decimal point in the second value indicates the decimal fraction of a second. A microsecond can be represented using six decimal digits.

date_string

Can have one of the following three formats:

- **Numeric string format** is exactly four, six, or eight digits. Four-digit strings are considered to be a year (century must be specified); the month and day are set to January 1. Six and eight-digit strings contain two or four digits for the year, followed by two for the month, and then two for the day. Because the component order is fixed with this format, the DATEFORMAT setting described in the *Developing Reporting Applications* manual is ignored.

If a numeric-string format longer than eight digits is encountered, it is treated as a combined date-time string in the *Hnn* format described in *Date-Time Formats* on page 4-2. The following are examples of numeric string date constants:

```
99
1999
19990201
```

- **Formatted-string format** contains a one or two-digit day, a one or two-digit month, and a two or four-digit year separated by spaces, slashes, hyphens, or periods. All three parts must be present and follow the DATEFORMAT setting described in the *Developing Reporting Applications* manual. If any of the three fields is four digits, it is interpreted as the year, and the other two fields must follow the order given by the DATEFORMAT setting. The following are examples of formatted-string date constants:

```
1999/05/20
5 20 1999
99.05.20
1999-05-20
```

- **Translated-string format** contains the full or abbreviated month name. The year must also be present in four-digit or two-digit form. If the day is missing, day 1 of the month is assumed; if present, it can have one or two digits. If the string contains both a two-digit year and a two-digit day, they must be in the order given by the DATEFORMAT setting. For example:

```
January 6 2000
```

Note:

- The date and time strings must be separated by at least one blank space. Blank spaces are also permitted at the beginning and end of the date-time string.
- In each date format, two-digit years are interpreted using the [F]DEFCENT and [F]YRTHRESH settings.

Text Field Format

`FIELD = fieldname, ALIAS = aliasname, USAGE = TXnn,$`

where:

fieldname

Is the name you assign the text field.

aliasname

Is an alternate name for the field name.

nn

Is the output display length in TABLE for the text field. The display length may be between 1 and 256 characters.

All letters, digits, and special characters can be stored with this format. The following are some sample text field formats.

Format	Display
TX50	This course provides the DP professional with the skills needed to create, maintain, and report from FOCUS data sources.
TX35	This course provides the DP professional with the skills needed to create, maintain, and report from FOCUS data sources.

The standard edit options are not available for the text field format.

The Stored Data Type: ACTUAL

ACTUAL describes the type and length of data as it is actually stored in the data source. While some data types, such as alphanumeric, are universal, others differ between different types of data sources. Some data sources support unique data types. For this reason, the values you can assign to the ACTUAL attribute differ for each type of data source.

For information about using ACTUAL with sequential, VSAM, and ISAM data sources, see Chapter 5, *Describing a Sequential, VSAM, or ISAM Data Source*. For other types of data sources, see the iWay documentation for your server. Note that FOCUS data sources do not use the ACTUAL attribute, and instead rely upon the USAGE attribute to specify both how a field is stored and how it is formatted.

Null or MISSING Values: MISSING

If a segment instance exists but no data has been entered into one of its fields, that field has no value. Some types of data sources represent this absence of data as a blank space () or zero (0), but others explicitly indicate an absence of data with a null indicator or as a special null value. Null values (sometimes known as missing data) are significant in reporting applications, especially those that perform aggregating functions such as averaging.

If your type of data source supports missing data, as do FOCUS data sources and most relational data sources, then you can use the optional MISSING attribute to enable null values to be entered into and read from a field. MISSING plays a role when you:

- **Create new segment instances.** If no value is supplied for a field for which MISSING has been set to ON, then the field is assigned a missing value.
- **Generate reports.** If a field with a null value is retrieved, the field value is not used in aggregating calculations such as averaging and summing. If the report calls for the field's value to be displayed, a special character is displayed to indicate a missing value. The default character is a period (.), but you can change it to any character string you wish using the SET NODATA command, as described in the *Developing Reporting Applications* manual.

Syntax

How to Specify a Missing Value

MISSING = {ON|OFF}

where:

ON

Distinguishes a missing value from an intentionally entered blank or zero when creating new segment instances and reporting.

OFF

Does not distinguish between missing values and blank or zero values when creating new segment instances and reporting. This is the default value.

Reference

Usage Notes for MISSING

Note the following rules when using MISSING:

- **Alias.** MISSING does not have an alias.
- **Setting.** It is recommended that you set the MISSING attribute to match the field's predefined null characteristic (whether the characteristic was explicitly set when the data source was created, or set by default). For example, if a relational table column has been created with the ability to accept null data, you should describe the field with the MISSING attribute set to ON so that its null values are correctly interpreted. This is not a consideration for FOCUS data sources, for which the field declaration in the Master File both defines the field and describes it.
- **Changes.** You can change the MISSING attribute at any time. Note that changing MISSING will not affect the actual stored data values that had been entered using the old setting. However, it will affect how that data is interpreted: if null data is entered when MISSING is set to ON, and then MISSING is switched to OFF, the data originally entered as null will be interpreted as blanks (for alphanumeric fields) or zeroes (for numeric fields). The only exception is FOCUS data sources, in which the data originally entered as missing will be interpreted as the internal missing value for that data type, which is described in Chapter 6, *Describing a FOCUS Data Source*.

Using a Missing Value

Consider the field values shown in the following four records:

		1	3
--	--	---	---

If you average these values without declaring the field with the MISSING attribute, a value of zero will automatically be supplied for the two blank records. Thus, the average of these four records will be $(0+0+1+3)/4$ or 1. If you set MISSING to ON, the two blank records will not be used in the calculation, so the average will be $(1+3)/2$ or 2.

Missing values in a unique segment are also automatically supplied with a zero, a blank, or a missing value depending on the MISSING attribute. What distinguishes missing values in unique segments from others is that they are not stored. You do have to supply a missing attribute for fields in unique segments on which you want to perform counts or averages.

The *Creating Reports With WebFOCUS Language* manual contains a more thorough discussion of using null values (sometimes called missing data) in reports. Included in the discussion are alternative ways of distinguishing these values in reports, such as using the WHERE phrase with MISSING selection operators, and creating virtual fields using the DEFINE FILE command with the SOME or ALL phrase.

Defining a Dimension: WITHIN

The OLAP model organizes data structures by pre-defining dimensions in the Master File, using the field name attribute WITHIN. A dimension is a group or list of related fields called *elements*. A Master File can define up to 500 elements, including fields in joined or cross-referenced files.

The WITHIN attribute enables drill up and drill down functionality on hierarchical dimensions. You can manipulate a report by selecting an OLAP-enabled field and drilling down to view other levels of a dimension's hierarchy.

For example, a hierarchy of sales regions can be defined in the Master File as the GEOGRAPHY dimension and can include the following fields (elements): Region, State, and City in descending order. Region, the highest element in the hierarchy, would contain a list of all of the Regions within the GEOGRAPHY dimension. State, the second highest element in the hierarchy, would contain a list of all available States within Region, and so on. Dimensions can be defined in the Master File for any supported data source.

The combination, or matrix, of two or more dimensional hierarchies in an OLAP-enabled data source is called "multi-dimensional." For example, although products are sold within states they need not be grouped in the same dimension as states. Instead, the elements Product Category and Product Name likely would be grouped in a dimension called PRODUCT. State would be a member of the GEOGRAPHY dimension that also can include Region and City. These dimensions are combined in a matrix so that the intersections of their criteria provide specific values, for example, sales of coffee in the Northeast region.

You can specify a list of acceptable values for each dimension element (field) using the ACCEPT attribute. This is done using either a hard coded list in the Master File or a lookup file. For more information on the ACCEPT attribute, see *Validating Data: ACCEPT* on page 4-2.

Syntax

How to Define a Dimension

`WITHIN= '*dimensionname'`

`WITHIN=field`

where:

`'*dimensionname'`

Is the name of the dimension and can include up to 66 characters. The dimension is defined in the field declaration for the field that is at the top of the hierarchy. The name must be preceded by an asterisk and enclosed within single quotation marks. The name must start with a letter and can consist of any combination of letters, digits, underscores, or periods. Avoid using special characters and embedded blanks.

`field`

Is used to define the hierarchical relationship among additional elements to be included in a given dimension. After the dimension name is defined at the top of the hierarchy, each element (field) uses the WITHIN attribute to link to the field directly above it in the hierarchy. The WITHIN attribute can refer to a field either by its field name or its alias. Note that a given field may participate in only one dimension, and two fields cannot reference the same higher level field.

Example

Defining a Dimension

The following example shows how to define the PRODUCT dimension in the OSALES Master File.

PRODUCT Dimension

====> Product Category

====> Product Name

```
FILENAME=OSALES,  SUFFIX=FOC
SEGNAME=SALES01,  SEGTYPE=S1
FIELD=PRODCAT,   ALIAS=PCAT,   FORMAT=A11,
WITHIN= '*PRODUCT', $
FIELD=PRODNAME,  ALIAS=PNAME,   FORMAT=A16,
WITHIN=PRODCAT, $
```

Example

Defining Multiple Dimensions

The following annotated example shows how to define the dimensions, PRODUCT, GEOGRAPHY, and TIME in the OSALES Master File.

PRODUCT Dimension

- ====> Product Category
- ====> Product Name

GEOGRAPHY Dimension

- ====> Region
- ====> State
- ====> City
- ====> Store Name (from the OSTORES Master File)

TIME Dimension

- ====> Year
- ====> Quarter
- ====> Month
- ====> Date

OSALES Master File

```

FILENAME=OSALES, SUFFIX=FOC
SEGNAME=SALES01, SEGTYPE=S1
  FIELD=SEQ_NO,      ALIAS=SEQ,      FORMAT=I5,          TITLE='Sequence#',
  DESC='Sequence number in database', $
  FIELD=PRODCAT,    ALIAS=PCAT,      FORMAT=A11, INDEX=I, TITLE='Category',
  DESC='Product category',
  ACCEPT='Coffee' OR 'Food' OR 'Gifts',
1.    WITHIN='*PRODUCT', $
  FIELD=PRODCODE,   ALIAS=PCODE,     FORMAT=A4,  INDEX=I, TITLE='Product ID',
  DESC='Product Identification code (for sale)', $
  FIELD=PRODNAME,   ALIAS=PNAME,     FORMAT=A16,          TITLE='Product',
  DESC='Product name',
2.    ACCEPT='Espresso' OR 'Latte' OR 'Cappuccino' OR 'Scone' OR 'Biscotti' OR
  'Croissant' OR 'Mug' OR 'Thermos' OR 'Coffee Grinder' OR 'Coffee Pot',
      WITHIN=PRODCAT, $
  FIELD=REGION,     ALIAS=REG,      FORMAT=A11, INDEX=I, TITLE='Region',
  DESC='Region code',
  ACCEPT='Midwest' OR 'Northeast' OR 'Southwest' OR 'West',
3.    WITHIN='*GEOGRAPHY', $
  FIELD=STATE       ALIAS=ST,        FORMAT=A2,  INDEX=I, TITLE='State',
  DESC='State',
4.    ACCEPT=(OSTATE),
      WITHIN=REGION, $
  FIELD=CITY,       ALIAS=CTY,      FORMAT=A20,          TITLE='City',
  DESC='City',
      WITHIN=STATE, $
  FIELD=STORE_CODE, ALIAS=STCD,     FORMAT=A5,  INDEX=I, TITLE='Store ID',
  DESC='Store identification code (for sale)', $
  FIELD=DATE,       ALIAS=DT,      FORMAT=I8YYMD,      TITLE='Date',
  DESC='Date of sales report',
      WITHIN=MO, $
5.    FIELD=UNITS,   ALIAS=UN,        FORMAT=I8,          TITLE='Unit Sales',
  DESC='Number of units sold', $
  FIELD=DOLLARS,    ALIAS=DOL,      FORMAT=I8,          TITLE='Dollar Sales',
  DESC='Total dollar amount of reported sales', $
  FIELD=BUDUNITS,   ALIAS=BUNIITS,     FORMAT=I8,          TITLE='Budget Units',
  DESC='Number of units budgeted', $
  FIELD=BUDDOLLARS, ALIAS=BDOLLARS,     FORMAT=I8,          TITLE='Budget Dollars',
  DESC='Total sales quota in dollars', $
6.    DEFINE ADATE/A8 = EDIT (DATE); $
  DEFINE YR/I4 = EDIT (EDIT (ADATE, '9999$$$$')); WITHIN='*TIME', $
  DEFINE MO/I2 = EDIT (EDIT (ADATE, '$$$$99$$')); WITHIN=QTR, $
  DEFINE QTR/I1 = IF MO GE 1 AND MO LE 3 THEN 1 ELSE IF MO GE 4 AND MO LE 6
  THEN 2 ELSE IF MO GE 7 AND MO LE 9 THEN 3 ELSE IF MO GE 10 AND MO LE 12
  THEN 4 ELSE 0; WITHIN=YR, $
7.    SEGNAME = STORES01, SEGTYPE = KU, PARENT = SALES01,
      CRFILE = OSTORES, CRKEY = STORE_CODE, $

```

1. Declares the PRODUCT dimension. The name must be preceded by an asterisk and enclosed within single quotation marks.
2. A list of acceptable values may be defined for each dimension element (field), if you wish. You specify the ACCEPT attribute using either a hard coded list in the Master File or an external flat file. The list of acceptable values is presented to you as possible selection criteria values. In this example, the value for the product name must be Espresso, Latte, Cappuccino, Scone, Biscotti, Croissant, Mug, Thermos, Coffee Grinder, or Coffee Pot. For more information on the ACCEPT attribute, see *Specifying Acceptable Values for a Dimension* on page 4-2.
3. Declares the GEOGRAPHY dimension and defines the dimension hierarchy for GEOGRAPHY: Region within GEOGRAPHY (top of the hierarchy), State within Region, and City within State.
4. In this example, the ACCEPT attribute is using an external flat file (OSTATE) to determine all of the possible data values for state (field ST).
5. The four fields, UNITS, DOLLARS, BUDUNITS, and BUDDOLLARS, are examples of measure fields. A measure field is used for analysis that typically defines how much or how many. For example, Units, Dollars, Budget Units, and Budget Dollars are measures that specify how many units were sold, the total dollar amount of reported sales, how many units were budgeted, and total sales quota in dollars, respectively.
6. Shows the following:
 - Virtual fields may be included at any level in a dimension. In this example, the fields YR, MO, and QTR are defined within the TIME dimension. The WITHIN attribute for a virtual field must be placed on the same line as the semicolon that ends the expression.
 - How to define the dimension hierarchy for the TIME dimension: Year within Time, Quarter within Year, Month within Quarter, Date within Month.
 - Fields in a hierarchy can occur in any order in the Master File.

7. Dimensions may span a dynamic or static JOIN structure using a qualified name. In this example, the OSALES data source is statically cross-referenced to the OSTORES data source using the common field STORE_CODE. Through this linkage, the OLAP application can retrieve the value for store name from the OSTORES data source. Note that STORE_NAME in the OSTORES Master File is an element of the GEOGRAPHY dimension that was defined in the OSALES Master File. The following is the OSTORES Master File:

```
FILENAME=OSTORES, SUFFIX=FOC
SEGNAME=STORES01, SEGTYPE=S1
  FIELD=STORE_CODE, ALIAS=STCD,  FORMAT=A5, INDEX=I, TITLE='Store ID',
    DESC='Franchisee ID Code', $
  FIELD=STORE_NAME, ALIAS=SNAME, FORMAT=A23,          TITLE='Store Name',
    DESC='Store Name',
    WITHIN=SALES01.CITY, $
  FIELD=ADDRESS1,  ALIAS=ADDR1,  FORMAT=A19,          TITLE='Contact',
    DESC='Franchisee Owner', $
  FIELD=ADDRESS2,  ALIAS=ADDR2,  FORMAT=A31,          TITLE='Address',
    DESC='Street Address', $
  FIELD=CITY,      ALIAS=CTY,    FORMAT=A22,          TITLE='City',
    DESC='City', $
  FIELD=STATE,    ALIAS=ST,     FORMAT=A2,          TITLE='State',
    DESC='State', $
  FIELD=ZIP,      ALIAS=ZIP,    FORMAT=A6,          TITLE='Zip Code',
    DESC='Postal Code', $
```

Validating Data: ACCEPT

ACCEPT is an optional attribute that you can use to validate data that is entered into a field from a MODIFY procedure. The ACCEPT test is applied immediately after a FIXFORM or FREEFORM is processed after which subsequent COMPUTE statements can manipulate the value. By including ACCEPT in a field declaration you can define a list or range of acceptable field values. In relational terms, you are defining the domain.

Note: Suffix VSAM and FIX data sources may use the ACCEPT attribute to specify multiple RECTYPE values, which are discussed in Chapter 5, *Describing a Sequential, VSAM, or ISAM Data Source*.

Syntax

How to Validate Data

ACCEPT = *list*

ACCEPT = *range*

ACCEPT = FIND (*field* [AS *name*] IN *file*)

where:

list

Is a string of acceptable values. The syntax is:

value1 OR *value2* OR *value3*...

For example, ACCEPT = RED OR WHITE OR BLUE. You can also use a blank as an item separator. If the list of acceptable values runs longer than one line, continue it on the next. The list is terminated by a comma.

range

Gives the range of acceptable values. The syntax is:

value1 TO *value2*

For example, ACCEPT = 150 TO 1000.

FIND

Verifies the incoming data against the values in another indexed field. This option is available only for FOCUS data sources. See Chapter 6, *Describing a FOCUS Data Source*, for more information.

Any value in the ACCEPT that contains an embedded blank (for example, Great Britain) must be enclosed within single quotation marks. For example:

ACCEPT = SPAIN OR ITALY OR FRANCE OR 'GREAT BRITAIN'

If the ACCEPT attribute is included in a field declaration and the SET command parameter ACCBLN has a value of OFF, blank () and zero (0) values will be accepted only if they are explicitly coded into the ACCEPT. SET ACCBLN is described in the *Developing Reporting Applications* manual.

Reference**Usage Notes for ACCEPT**

Note the following rules when using ACCEPT:

- **Alias.** ACCEPT does not have an alias.
- **Changes.** You can change the information in an ACCEPT attribute at any time.
- **Virtual fields.** You cannot use the ACCEPT attribute to validate virtual fields created with the DEFINE attribute.
- **HOLD files.** If you wish the ACCEPT attribute to be propagated into the Master File of a HOLD file, use the SET HOLDATTR command. HOLD files are discussed in the *Creating Reports With WebFOCUS Language* manual.
- **ACCEPT** is used only in MODIFY procedures. It is useful for providing one central validation list to be used by several procedures. The FIND function is useful when the list of values is large or undergoes frequent change.

Specifying Acceptable Values for a Dimension

The optional ACCEPT attribute enables you to specify a list of acceptable values for use in an OLAP-enabled Master File.

ACCEPT is useful whenever data fields are to be addressed by several requests. One ACCEPT attribute in the Master File eliminates the need to provide lists in each separate procedure (FOCEXEC).

The ACCEPT attribute enables you to specify:

- A list of acceptable data values.
- A range of acceptable data values.
- Acceptable data values that match any value contained in an external flat data source (lookup).

The use of the lookup option is helpful when the list of values is large and undergoes frequent change.

Syntax**How to Specify a List of Acceptable Values for a Dimension**

```
ACCEPT=value1 OR value2 OR value3...
```

where:

```
value1, value2, value3...
```

Is a list of acceptable values. Any value in the ACCEPT list which contains an embedded blank must be enclosed within single quotation marks. For values without embedded blanks, single quotation marks are optional.

Example

Specifying a List of Acceptable Products for a Dimension

The following shows the ACCEPT syntax in the Master File, which specifies the list of acceptable products: Espresso, Latte, Cappuccino, Scone, Biscotti, Croissant, Mug, Thermos, Coffee Grinder, and Coffee Pot.

```
ACCEPT='Espresso' OR 'Latte' OR 'Cappuccino' OR 'Scone' OR 'Biscotti' OR  
      'Croissant' OR 'Mug' OR 'Thermos' OR 'Coffee Grinder' OR 'Coffee  
      Pot'
```

Syntax

How to Specify a Range of Acceptable Values for a Dimension

```
ACCEPT=value1 TO value2
```

where:

value1

Is the value for the beginning of the range.

value2

Is the value for the end of the range.

Example

Specifying a Range of Acceptable Values for a Dimension

The following shows the ACCEPT syntax in the Master File, which specifies a range of acceptable data values between 150 and 1000.

```
ACCEPT=150 TO 1000
```

Syntax

How to Verify Data Against Values in an External Flat Data Source

```
ACCEPT=(ddname)
```

where:

ddname

Is the ddname that points to the fully qualified name of an existing data source that contains the list of acceptable values. The ddname can consist of a maximum of 8 characters. Note that you must FILEDEF or ALLOCATE the ddname to the external data source. You can issue the FILEDEF or ALLOCATE statement in any valid profile.

Example

Verifying Data Against Acceptable Values for the State Field

The OSTATE data source contains the list of all acceptable values for the state field. The following shows the ACCEPT syntax in the Master File which verifies state values against the external OSTATE data source:

```
ACCEPT=(OSTATE)
```

Alternative Report Column Titles: TITLE

When you generate a report, each column title in the report defaults to the name of the field displayed in that column. However, you can change the default column title by specifying the optional TITLE attribute for that field.

Of course, you can always specify a different column title within an individual report by using the AS phrase in that report request, as described in the *Creating Reports With WebFOCUS Language* manual.

Note that the TITLE attribute has no effect in a report if the field is used with a prefix operator such as AVE. You can supply an alternative column title for fields used with prefix operators by using the AS phrase.

Syntax

How to Specify an Alternative Title

```
TITLE = 'text'
```

where:

text

Is any string of up to 64 characters. You can split the text across as many as five separate title lines by separating the lines with a comma (.). You can include blanks at the end of a column title by including a slash (/) in the final blank position. You must enclose the string within single quotation marks if it includes commas or leading blanks. For example,

```
FIELD = LNAME, ALIAS = LN, USAGE = A15, TITLE = 'Client,Name', $
```

replaces the default column heading, LNAME, with the following:

```
Client
Name
-----
```

Reference

Usage Notes for TITLE

Note the following rules when using TITLE:

- **Alias.** TITLE does not have an alias.
- **Changes.** You can change the information in TITLE at any time. You can also override the TITLE with an AS name in a request or turn it off with the SET TITLE=OFF command.
- **Virtual fields.** If you use the TITLE attribute for a virtual field created with the DEFINE attribute, the semicolon (;) terminating the DEFINE expression must be on the same line as the TITLE keyword.
- **HOLD files.** If you wish the TITLE attribute to be propagated into the Master File of a HOLD file, use the SET HOLDATTR command. HOLD files are discussed in the *Creating Reports With WebFOCUS Language* manual.

Documenting the Field: DESCRIPTION

DESCRIPTION is an optional attribute that enables you to provide comments and other documentation for a field within the Master File. You can include any comment up to 78 characters in length.

Note that you can also add documentation to a field declaration, or to a segment or file declaration, by typing a comment in the columns following the terminating dollar sign. You can even create an entire comment line by inserting a new line following a declaration and placing a dollar sign at the beginning of the line. The syntax and rules for creating a Master File are described in Chapter 1, *Understanding a Data Source Description*.

The DESCRIPTION attribute for a FOCUS data source can be changed at any time without rebuilding the data source.

Syntax

How to Supply Field Documentation

```
DESC[RIPTION] = text
```

where:

DESCRIPTION

Can be shortened to DESC. Abbreviating the keyword has no effect on its function.

text

Is any string of up to 78 characters. If the string contains a comma, the string must be enclosed within single quotation marks.

For example:

```
FIELD=UNITS,ALIAS=QTY,USAGE=I6, DESC='QUANTITY SOLD, NOT RETURNED', $
```

Reference

Usage Notes for DESCRIPTION

Note the following rules when using the DESCRIPTION attribute:

- **Alias.** The DESCRIPTION attribute has an alias of DEFINITION.
- **Changes.** You can change DESCRIPTION at any time.
- **Virtual fields.** If you use the DESCRIPTION attribute for a virtual field created with the DEFINE attribute, the DESCRIPTION attribute must be on the same line as the semicolon (;) terminating the DEFINE expression or, if there are other attributes in the declaration (such as TITLE), on the last line of the declaration.

Describing a Virtual Field: DEFINE

DEFINE is an optional attribute used to create a virtual field for reporting. You can derive the virtual field's value from information already in the data source—that is, from permanent fields. Some common uses of virtual data fields include:

- Computing new numerical values that are not on the data record.
- Computing a new string of alphanumeric characters from other strings.
- Classifying data values into ranges or groups.
- Invoking subroutines in calculations.

Virtual fields are available whenever the data source is used for reporting.

Syntax

How to Define a Virtual Field

```
DEFINE fieldname/format = expression; [, attribute2, ... ] $
```

where:

fieldname

Is the name of the virtual field. You can assign any name up to 66 characters long. The name is subject to the same conventions as names assigned using the FIELDNAME attribute. FIELDNAME is described in *The Field's Name: FIELDNAME* on page 4-2.

format

Is the field's format. The format is specified in the same way as formats assigned using the USAGE attribute, which is described in *The Displayed Data Type: USAGE* on page 4-2. If you do not specify a format, it defaults to D12.2.

expression

Is a valid expression. Expressions are fully described in the *Creating Reports With WebFOCUS Language* manual. The expression must end with a semicolon (;).

Note that when an IF-THEN phrase is used in the expression of a virtual field, it must include the ELSE phrase.

attribute2

The declaration for a virtual field can include additional optional attributes, such as TITLE and DESCRIPTION. Any additional attributes must be on the same line as the semicolon that ends the DEFINE expression. An exception is made for the DESCRIPTION attribute. However, if you put it on the final line of the declaration, it does not need to be on the same line as the semicolon.

You can devote an entire line to these additional attributes by placing the semicolon on the line following the DEFINE expression.

Place each DEFINE attribute after all of the field descriptions for that segment. For example, the following shows how to define a field called PROFIT in the segment CARS:

```
SEGMENT = CARS ,SEGTYPE = S1 ,PARENT = CARREC, $
  FIELDNAME = DEALER_COST ,ALIAS = DCOST ,USAGE = D7, $
  FIELDNAME = RETAIL_COST ,ALIAS = RCOST ,USAGE = D7, $
  DEFINE PROFIT/D7 = RETAIL_COST - DEALER_COST; $
```

Reference Usage Notes for Virtual Fields in a Master File

Note the following rules when using DEFINE:

- **Alias.** DEFINE does not have an alias.
- **Changes.** You can change the virtual field's declaration at any time.

Using a Virtual Field

A DEFINE attribute cannot contain qualified field names on the left-hand side of the expression. You can use the WITH phrase on the left-hand side to place the defined field in the same segment as any real field you choose.

When FIELDNAME is set to OLD, a DEFINE attribute in a Master File can only refer to fields in its own segment. In this case, if you want to create a virtual field that uses information from several different segments, you will have to create it with a DEFINE FILE command request prior to a report request as discussed in the *Creating Reports With WebFOCUS Language* manual.

When FIELDNAME is set to NEW, expressions on the right-hand side of the DEFINE can refer to fields from any segment in the same path. The expression on the right-hand side of a DEFINE or REDEFINES statement in a Master File can contain qualified field names.

A DEFINE attribute in a Master File can refer to only fields in its own path. If you want to create a virtual field that derives its value from fields in several different paths, you will have to create it with a DEFINE FILE command using an alternate view prior to a report request, as discussed in the *Creating Reports With WebFOCUS Language* manual. The DEFINE FILE command is also helpful when you wish to create a virtual field that will be used only once, and you do not want to add a declaration for it to the Master File. Virtual fields defined in the Master File are available whenever the data source is used and are treated like other stored fields. Thus, a field defined in the Master File cannot be cleared in your report request. A virtual field cannot be used for cross-referencing in a join.

Note: Maintain does not support DEFINE attributes that have a constant value. Using such a field in a Maintain procedure generates the following message:

```
(FOC03605) name is not recognized.
```

CHAPTER 5

Describing a Sequential, VSAM, or ISAM Data Source

Topics:

- Sequential Data Source Formats
- Standard Master File Attributes for a Sequential Data Source
- Standard Master File Attributes for a VSAM or ISAM Data Source
- Describing a Multiply Occurring Field in a Free-Format Data Source
- Describing A Multiply Occurring Field in a Fixed-Format, VSAM, or ISAM Data Source
- Redefining a Field in a Non-FOCUS Data Source
- Extra-Large Record Length Support
- Describing Multiple Record Types
- Combining Multiply Occurring Fields and Multiple Record Types
- VSAM Data and Index Buffers
- A VSAM Alternate Index

You can describe and report from sequential, VSAM, and ISAM data sources.

In a sequential data source, records are stored and retrieved in the same order as they were entered.

With VSAM and ISAM data sources a new element is introduced, the key or group key. A group key consists of one or more fields and can be used to identify the various record types in the data source. In the Master File representation of a data source with different record types, each record type is assigned its own segment.

Note: For VSAM and ISAM data sources, you must allocate the Master File name to the CLUSTER component of the data source.

Only ESDS and KSDS data sources are supported. If you wish to retrieve data from RRDS VSAM data sources, you may code your own access routine using SUFFIX=PRIVATE. For information, see the *WebFOCUS Security and Administration* manual.

Sequential Data Source Formats

Sequential data sources formatted in the following ways are recognized:

- Fixed-format, in which each field occupies a pre-defined position in the record.
- Free-format, also known as comma-delimited, in which fields can occupy any position in a record and a record can span multiple lines.

You can describe two types of sequential data sources:

- **Simple.** This is the most basic type, consisting of only one segment. It is supported in all formats.
- **Complex.** This is a multi-segment data source. The descendant segments exist in the data source as multiply occurring fields (which are supported in both fixed- and free-format) or multiple record types (which are supported only in fixed-format).

What Is a Fixed-Format Data Source?

Fixed-format data sources are sequential data sources in which each field occupies a pre-defined position in the record. You describe the record format in the Master File.

For example, a fixed-format record might look like this:

```
1352334556George Eliot The Mill on the Floss H
```

The simplest form of a fixed-record data source can be described by providing just field declarations. For example, suppose you have a data source for a library that consists of the following components:

- A number, like an ISBN number, that identifies the book by publisher, author, and title.
- The name of the author.
- The title of the book.
- A single letter that indicates whether the book is hard- or soft-bound.
- The book's price.
- A serial number that actually identifies the individual copies of the book in the library (a call number).
- A synopsis of the book.

This data source can be described with the seven field declarations shown here:

```
FIELDNAME = PUBNO ,ALIAS = PN ,USAGE = A10 ,ACTUAL = A10 ,  
FIELDNAME = AUTHOR ,ALIAS = AT ,USAGE = A25 ,ACTUAL = A25 ,  
FIELDNAME = TITLE ,ALIAS = TL ,USAGE = A50 ,ACTUAL = A50 ,  
FIELDNAME = BINDING ,ALIAS = BI ,USAGE = A1 ,ACTUAL = A1 ,  
FIELDNAME = PRICE ,ALIAS = PR ,USAGE = D8.2N ,ACTUAL = D8 ,  
FIELDNAME = SERIAL ,ALIAS = SN ,USAGE = A15 ,ACTUAL = A15 ,  
FIELDNAME = SYNOPSIS ,ALIAS = SYN ,USAGE = A150 ,ACTUAL = A150 ,
```

Note:

- Each declaration begins with the word FIELDNAME, and normally contains four elements (a FIELDNAME, an ALIAS, a USAGE attribute, and an ACTUAL attribute).
- ALIAS=, USAGE=, and ACTUAL= may be omitted as identifiers since they are positional attributes following FIELDNAME.
- If you omit the optional ALIAS, its absence must be signaled by a second comma between FIELDNAME and USAGE (FIELDNAME=PUBNO,,A10,A10,\$).
- Both the USAGE and the ACTUAL attributes must be included. Failure to specify both is a common cause of errors in describing non-FOCUS data sources (FOCUS data sources do not have ACTUAL attributes).
- Each declaration can span multiple lines and must be terminated with a comma followed by a dollar sign (,\$). The LRECL for a Master File is 80. The RECFM for a Master File is F.
- When using Maintain to read a fixed-format data source, the record length as described in the Master File may not exceed the actual length of the data record (the LRECL value).

You need to describe only those fields to which you intend to refer. This is very significant when using existing data sources, because they frequently contain information that you do not need for your requests. You describe only the fields you wish to include in your reports or calculations and use filler fields to represent the rest of the logical record length (LRECL) of the data source.

In the above example, the book synopsis is hardly necessary for most reports. The synopsis can, therefore, be replaced with a filler field, as follows:

```
FIELDNAME = FILLER, ALIAS = FILL1, USAGE = A150, ACTUAL = A150,$
```

Fillers of this form may contain up to 256 characters. If you need to describe larger areas, use several filler fields together:

```
FIELDNAME = FILLER, ,A256,A256,$
FIELDNAME = FILLER, ,A200,A200,$
```

The field name FILLER is no different than any other field name. To prevent access to the data in the field, you can use a blank field name. For example,

```
FIELDNAME = , ,A200,A200,$
```

We recommend including file and segment attributes, even for simple data sources, to complete your documentation. The example below shows the Master File for the library data source with file and segment declarations added.

```
FILENAME = LIBRARY1, SUFFIX = FIX,$
SEGNAME = BOOKS, SEGTYPE = S0,$
FIELDNAME = PUBNO ,ALIAS = PN ,USAGE = A10 ,ACTUAL = A10 ,,$
FIELDNAME = AUTHOR ,ALIAS = AT ,USAGE = A25 ,ACTUAL = A25 ,,$
FIELDNAME = TITLE ,ALIAS = TL ,USAGE = A50 ,ACTUAL = A50 ,,$
FIELDNAME = BINDING,ALIAS = BI ,USAGE = A1 ,ACTUAL = A1 ,,$
FIELDNAME = PRICE ,ALIAS = PR ,USAGE = D8.2N ,ACTUAL = D8 ,,$
FIELDNAME = SERIAL ,ALIAS = SN ,USAGE = A15 ,ACTUAL = A15 ,,$
FIELDNAME = FILLER ,ALIAS = FILL1 ,USAGE = A150 ,ACTUAL = A150 ,,$
```

What Is a Free-Format Data Source?

A common type of external structure is a comma-delimited sequential data source. These data sources are a convenient way to maintain low volumes of data, since the fields in a record are separated from one another by commas rather than being padded with blanks or zeroes to fixed field lengths. Comma-delimited data sources must be stored as physical sequential data sources.

The report request language processes comma-delimited data sources in the same way it processes fixed-format data sources. The same procedure is used to describe these data sources in a comma-delimited Master File. The only difference is that the file suffix is changed to COM, as shown:

```
FILENAME = filename, SUFFIX = COM,$
```

You can use the system editor to change values, add new records, and delete records. Since the number of data fields on a line is variable, depending on the presence or absence of fields and the actual length of the data values, a logical record may be one or several lines. Hence, you need to use a terminator character to signal the end of the logical record. This is a dollar sign following the last comma (,\$). A section of comma-delimited data might look like this:

```
PUBNO=1352334556, AUTHOR='Eliot, George',  
TITLE='The Mill on the Floss', BINDING=H,$
```

The order in which the data values are described in the Master File plays an important role in comma-delimited data sources. If the data values are typed in their natural order, then only commas between the values are necessary. If a value is out of its natural order, then it is identified by its name or alias and an equal sign preceding it, for example, AUTHOR= 'Eliot, George'.

Rules for Maintaining a Free-Format Data Source

If a logical record contains every data field, it will contain the same number of commas used as delimiters as there are data fields. It will also have a dollar sign following the last comma, signaling the end of the logical record. Thus, a logical record containing ten data fields will contain ten commas as delimiters, plus a dollar sign record terminator.

A logical record may occupy as many lines in the data source as is necessary to contain the data. A record terminator (,\$) must follow the last physical field.

Each record need not contain every data field, however. The identity of a data field that might be out of sequence can be provided in one of the following ways:

- You can use the field name, followed by an equal sign and the data value.
- You can use the field's alias, followed by an equal sign and the data value.
- You can use the shortest unique truncation of the field's name or alias, followed by an equal sign and the data value.
- If a field name is not mentioned, it inherits its value from the prior record.

Thus, the following statements are all equivalent.

```
BI=H, PRICE=17.95,$  
BI=H, PR=17.95,$  
BI=H, P=17.95,$
```

Standard Master File Attributes for a Sequential Data Source

Most standard Master File attributes—those described in Chapter 2, *Identifying a Data Source*, Chapter 3, *Describing a Group of Fields*, and Chapter 4, *Describing an Individual Field*—are used with sequential data sources in the standard way.

- **SEGTYPE.** The SEGTYPE attribute is ignored with free-format data sources. The SEGTYPE value for fixed-format data sources defaults to S0. However, if you use keyed retrieval, the SEGTYPE value depends on the number of keys and sort sequence. See Chapter 6, *Describing a FOCUS Data Source*, for a description of the SEGTYPE attribute. For a description of keyed retrieval from fixed format data sources, see the *Creating Reports With WebFOCUS Language* manual.
- **ACTUAL.** The ACTUAL values for sequential data sources are described in Chapter 4, *Describing an Individual Field*.

Note that file and segment declarations are optional for simple sequential data sources that you will not join. However, they are recommended to make the data source description self-documenting, and to give you the option of joining the data source in the future.

Standard Master File Attributes for a VSAM or ISAM Data Source

Most standard Master File attributes—those described in Chapter 2, *Identifying a Data Source*, Chapter 3, *Describing a Group of Fields*, and Chapter 4, *Describing an Individual Field*—are used with VSAM and ISAM data sources in the standard way.

- **SUFFIX.** The SUFFIX attribute in the file declaration for these data sources has the value VSAM or ISAM.
- **SEGNAME.** The SEGNAME attribute of the first or root segment in a Master File for a VSAM or ISAM data source must be ROOT. The remaining segments can have any valid segment name.

The only exception involves unrelated RECTYPES, where the root SEGNAME must be DUMMY.

All non-repeating data goes in the root segment. The remaining segments may have any valid name from one to eight characters.

Any segment except the root is the descendant, or child, of another segment. The PARENT attribute supplies the name of the segment that is the hierarchical parent or owner of the current segment. If no PARENT attribute appears, the default is the immediately preceding segment. The PARENT name may be one to eight characters.

- **SEGTYPE.** The SEGTYPE attribute should be S0 for VSAM data sources. (For a general description of the SEGTYPE attribute, see Chapter 3, *Describing a Group of Fields*.)
- **GROUP.** The keys of a VSAM or ISAM data source are defined in the segment declarations as GROUPs consisting of one or more fields.

Group Fields

A single segment data source may have only one key field, but it must still be described with a GROUP declaration. The group must have ALIAS=KEY.

Groups can also be assigned simply to provide convenient reference names for groups of fields. Suppose, for instance, that you have a series of three fields for an employee: last name; first name; and middle initial. You use these three fields consistently to identify the employee. You can identify the three fields in your Master File as a GROUP named EMPINFO. Then, you can refer to these three linked fields as a single unit, called EMPINFO. When using the GROUP feature for non-keys, the parameter ALIAS= must still be used, but should not equal KEY.

For group fields you must supply both the USAGE and ACTUAL formats in alphanumeric format. The length must be exactly the sum of the subordinate field lengths.

The GROUP declaration USAGE attribute specifies how many positions to use to describe the key in a VSAM KSDS data source. If a Master File does not completely describe the full key at least once, the following warning message is returned:

(FOC1016) INVALID KEY DESCRIPTION IN MASTER FILE

The cluster key definition is compared to the Master File for length and displacement. When you expand on the key in a RECTYPE data source, describe the key length in full on the last non-OCCURS segment on each data path.

Do not describe a group with ALIAS=KEY for OCCURS segments.

If the fields that make up a group key are not alphanumeric fields, the format of the group key is still alphanumeric, but its length is determined differently. The ACTUAL length is still the sum of the subordinate field lengths. The USAGE format, however, is the sum of the internal storage lengths of the subordinate fields. You determine these internal storage lengths as follows:

- Fields of type I have a value of 4.
- Fields of type F have a value of 4.
- Fields of type P that are 8 bytes can have a USAGE of P15 or P16 (sign and decimal for a total of 15 digits). Fields that are 16 bytes will have a USAGE of P17 or larger.
- Fields of type D have a value of 8.
- Alphanumeric fields have a value equal to the number of characters they contain as their field length.

Note: Since all group fields must be defined in alphanumeric format, those that include numeric component fields should not be used as verb objects in a report request.

Syntax

How to Describe a VSAM Group Field

`GROUP = keyname , ALIAS = KEY , USAGE = Ann , ACTUAL = Ann , $`

where:

keyname

Can have up to 66 characters.

Example

Describing a VSAM Group Field

In the library data source, the first field, PUBNO, could be described as a group key. The publisher's number consists of three elements: a number that identifies the publisher, one that identifies the author, and one that identifies the title. They can be described as a group key, consisting of a separate field for each element if the data source were a VSAM data structure. The Master File would look as follows:

```
FILE = LIBRARY5 , SUFFIX = VSAM , $
SEGMENT = ROOT , SEGTYPE = S0 , $
GROUP = BOOKKEY , ALIAS = KEY , USAGE = A10 , ACTUAL = A10 , $
FIELDNAME = PUBNO , ALIAS = PN , USAGE = A3 , ACTUAL = A3 , $
FIELDNAME = AUTHNO , ALIAS = AN , USAGE = A3 , ACTUAL = A3 , $
FIELDNAME = TITLNO , ALIAS = TN , USAGE = A4 , ACTUAL = A4 , $
FIELDNAME = AUTHOR , ALIAS = AT , USAGE = A25 , ACTUAL = A25 , $
FIELDNAME = TITLE , ALIAS = TL , USAGE = A50 , ACTUAL = A50 , $
FIELDNAME = BINDING , ALIAS = BI , USAGE = A1 , ACTUAL = A1 , $
FIELDNAME = PRICE , ALIAS = PR , USAGE = D8.2N , ACTUAL = D8 , $
FIELDNAME = SERIAL , ALIAS = SN , USAGE = A15 , ACTUAL = A15 , $
FIELDNAME = SYNOPSIS , ALIAS = SY , USAGE = A150 , ACTUAL = A150 , $
FIELDNAME = RECTYPE , ALIAS = B , USAGE = A1 , ACTUAL = A1 , $
```

Describing a Multiply Occurring Field in a Free-Format Data Source

Since any data field not explicitly referred to in a logical record continues to have the same value it had the last time a value was assigned, up until the point a new data value is entered, a free-format sequential data source can resemble a hierarchical structure. The parent information need be entered only once, and it will carry over for each descendant segment.

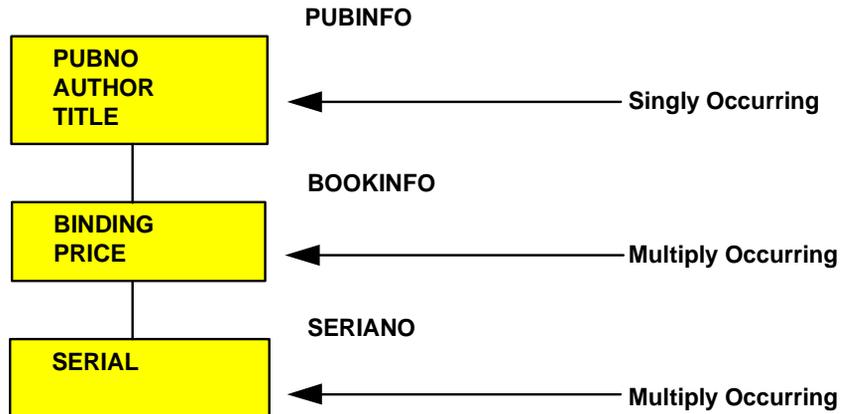
Example

A Multiply Occurring Field in a Free-Format Data Source

Consider our example of a library data source. The information for two copies of *The Sun Also Rises*, one hardcover and one paperback, can be entered as follows:

```
PUBNO=1234567890, AUTHOR='Hemingway, Ernest',  
TITLE='The Sun Also Rises',  
BI=H,PR=17.95, $\br/>BI=S,PR=5.25, $
```

There are two values for binding and price, which both correspond to the same publisher's number, author, and title. In the Master File, the information that occurs only once—the publisher's number, author, and title—is placed in one segment and the information that occurs several times in relation to this information is placed in a descendant segment. Similarly, information that occurs several times in relation to the descendant segment, such as an individual serial number for each copy of the book, is placed in a segment that is a descendant of the first descendant segment, as shown in the following diagram:



You describe this data source with the following data source description:

```
FILENAME = LIBRARY4, SUFFIX = COM, $  
  
SEGNAME = PUBINFO, SEGTYPE=S0, $  
  FIELDNAME = PUBNO, ALIAS = PN, USAGE = A10, ACTUAL = A10, $  
  FIELDNAME = AUTHOR, ALIAS = AT, USAGE = A25, ACTUAL = A25, $  
  FIELDNAME = TITLE, ALIAS = TL, USAGE = A50, ACTUAL = A50, $  
  
SEGNAME = BOOKINFO, PARENT = PUBINFO, SEGTYPE=S0, $  
  FIELDNAME = BINDING, ALIAS = BI, USAGE = A1, ACTUAL = A1, $  
  FIELDNAME = PRICE, ALIAS = PR, USAGE = D8.2N, ACTUAL = D8, $  
  
SEGNAME = SERIANO, PARENT = BOOKINFO, SEGTYPE=S0, $  
  FIELDNAME = SERIAL, ALIAS = SN, USAGE = A15, ACTUAL = A15, $
```

Note that each segment other than the first has a PARENT attribute. You use the PARENT attribute to signal that you are describing a hierarchical structure.

Describing A Multiply Occurring Field in a Fixed-Format, VSAM, or ISAM Data Source

Fixed-format sequential, VSAM, or ISAM data sources can have repeating fields. Consider the following data structure:

A	B	C1	C2	C1	C2
---	---	----	----	----	----

Fields C1 and C2 repeat within this data record. C1 has an initial value, as does C2. C1 then provides a second value for that field, as does C2. Thus, there are two values for fields C1 and C2 for every one value for fields A and B.

The number of times C1 and C2 occur does not have to be fixed. It can depend on the value of a counter field. Suppose field B is this counter field. In the case shown above, the value of field B is 2, since C1 and C2 occur twice. The value of field B in the next record can be 7, 1, 0, or any other number you choose and fields C1 and C2 will occur that number of times.

The number of times fields C1 and C2 occur can also be variable. In this case, everything after fields A and B is assumed to be a series of C1s and C2s, alternating to the end of the record.

You describe these multiply occurring fields by placing them in a separate segment. Fields A and B are placed in the root segment. Fields C1 and C2, which occur multiply in relation to A and B, are placed in a descendant segment. You use an additional segment attribute, the OCCURS attribute, to specify that these segments represent multiply occurring fields. In certain cases, you may also need a second attribute, called the POSITION attribute.

The OCCURS Attribute

The OCCURS attribute is an optional segment attribute used to describe records containing repeating fields or groups of fields. You define such records by describing the singly occurring fields in one segment and the multiply occurring fields in a descendant segment. The OCCURS attribute appears in the declaration for the descendant segment.

You can have several sets of repeating fields in your data structure. You describe each of these sets of fields as a separate segment in your data source description. Sets of repeating fields can be divided into two basic types: parallel and nested.

Syntax

How to Specify a Repeating Field

`OCCURS = occurstype`

Possible values are:

n

Is an integer value showing the number of occurrences (from 1 to 4095).

fieldname

Names a field in the parent segment whose integer value contains the number of occurrences of the descendant segment.

`VARIABLE`

Indicates that the number of occurrences varies from record to record. The number of occurrences is computed from the record length (that is, if the field lengths for the segment add up to 40, and 120 characters are read in, it means there are three occurrences).

You place the OCCURS attribute in your segment declaration after the PARENT attribute.

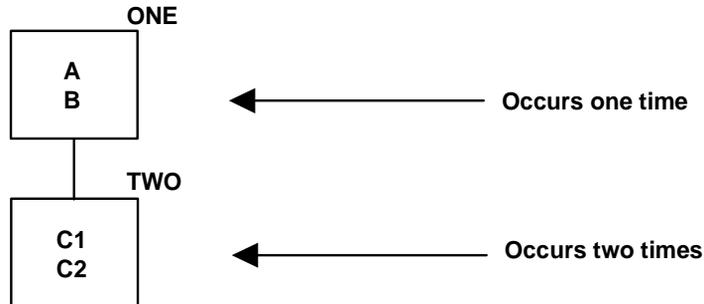
When different types of records are combined in one data source, each record type can contain only one segment defined as OCCURS=VARIABLE. It may have OCCURS descendants (if it contains a nested group), but it may not be followed by any other segment with the same parent—that is, there can be no other segments to its right in the hierarchical data structure. This restriction is necessary to ensure that data in the record is interpreted unambiguously.

Example Using the OCCURS Attribute

Consider the following simple data structure:

A	B	C1	C2	C1	C2
---	---	----	----	----	----

You have two occurrences of fields C1 and C2 for every one occurrence of fields A and B. Thus, to describe this data source, you place fields A and B in the root segment, and fields C1 and C2 in a descendant segment, as shown here:



You describe this data source with the following data source description:

```
FILENAME = EXAMPLE1, SUFFIX = FIX, $
```

```
SEGNAME = ONE, SEGTYPE=S0, $
FIELDNAME = A, ALIAS= , USAGE = A2, ACTUAL = A2, $
FIELDNAME = B, ALIAS= , USAGE = A1, ACTUAL = A1, $
```

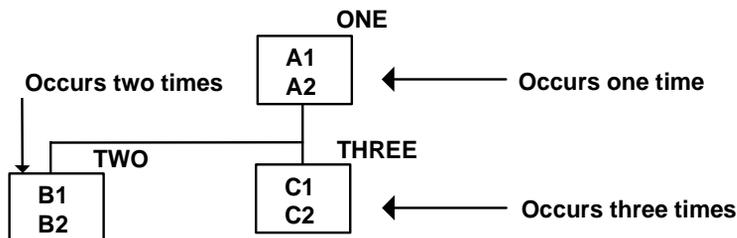
```
SEGNAME = TWO, PARENT = ONE, OCCURS = 2, SEGTYPE=S0, $
FIELDNAME = C1, ALIAS= , USAGE = I4, ACTUAL = I2, $
FIELDNAME = C2, ALIAS= , USAGE = I4, ACTUAL = I2, $
```

A Parallel Set of Repeating Fields

Parallel sets of repeating fields are those that have nothing to do with one another (that is, they have no parent-child or logical relationship). Consider the following data structure:

A1	A2	B1	B2	B1	B2	C1	C2	C1	C2	C1	C2
----	----	----	----	----	----	----	----	----	----	----	----

In this example, fields B1 and B2 and fields C1 and C2 repeat within the record. The number of times that fields B1 and B2 occur has nothing to do with the number of times fields C1 and C2 occur. Fields B1 and B2 and fields C1 and C2 are parallel sets of repeating fields. They should be described in the data source description as children of the same parent, the segment that contains fields A1 and A2. The following data structure reflects their relationship:

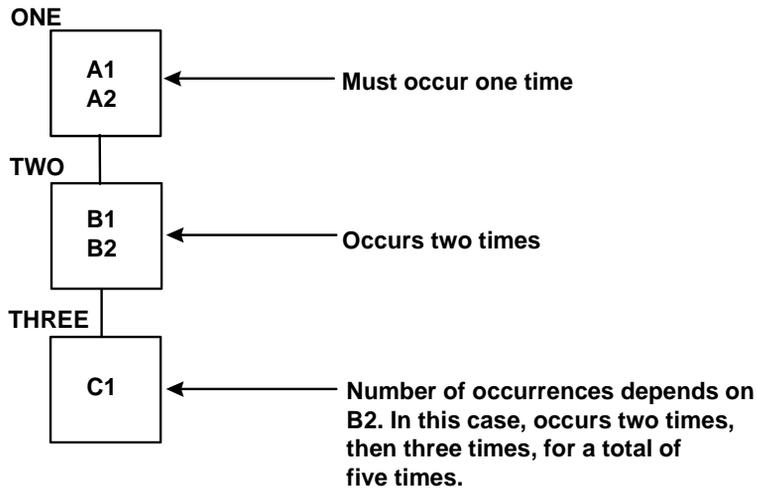


A Nested Set of Repeating Fields

Nested sets of repeating fields are those whose occurrence in some way depends on one another. Consider the following data structure:

A1	A2	B1	B2	C1	C1	B1	B2	C1	C1	C1
----	----	----	----	----	----	----	----	----	----	----

In this example, field C1 only occurs after fields B1 and B2 occur once. It occurs varying numbers of times recorded by a counter field, B2. The one thing we know about this field is you will not have a set of occurrences of C1 without it being preceded by an occurrence of fields B1 and B2. Fields B1, B2, and C1 are a nested set of repeating fields. They can be represented by the following data structure:



Since field C1 repeats with relation to fields B1 and B2, which repeat in relation to fields A1 and A2, field C1 is described as a separate, descendant segment of Segment Two, which is in turn a descendant of Segment One.

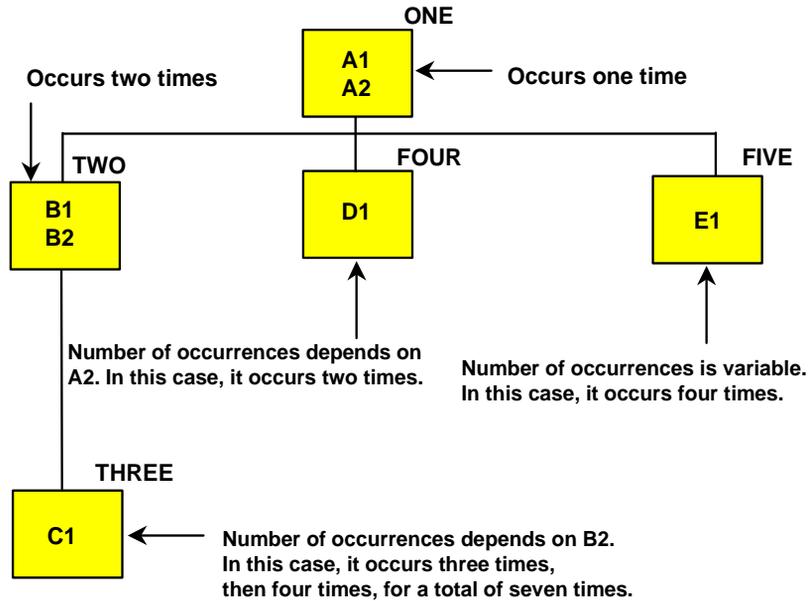
Example

Describing Parallel and Nested Repeating Fields

The following data structure contains both nested and parallel sets of repeating fields.

A1	A2	B1	B2	C1	C1	C1	B1	B2	C1	C1	C1	C1	D1	D1	E1	E1	E1	E1
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

It produces the following data structure:



You describe this data source with the following data source description. Notice that the assignment of the PARENT attributes shows you how the occurrences are nested.

```

FILENAME = EXAMPLE3, SUFFIX = FIX,$

SEGNAME = ONE, SEGTYPE=S0,$
FIELDNAME = A1 ,ALIAS= ,ACTUAL = A1 ,USAGE = A1 ,,$
FIELDNAME = A2 ,ALIAS= ,ACTUAL = I1 ,USAGE = I1 ,,$

SEGNAME = TWO, SEGTYPE=S0, PARENT = ONE, OCCURS = 2 ,,$
FIELDNAME = B1 ,ALIAS= ,ACTUAL = A15 ,USAGE = A15 ,,$
FIELDNAME = B2 ,ALIAS= ,ACTUAL = I1 ,USAGE = I1 ,,$

SEGNAME = THREE, SEGTYPE=S0, PARENT = TWO, OCCURS = B2 ,,$
FIELDNAME = C1 ,ALIAS= ,ACTUAL = A25 ,USAGE = A25 ,,$

SEGNAME = FOUR, SEGTYPE=S0, PARENT = ONE, OCCURS = A2 ,,$
FIELDNAME = D1 ,ALIAS= ,ACTUAL = A15 ,USAGE = A15 ,,$

SEGNAME = FIVE, SEGTYPE=S0, PARENT = ONE, OCCURS = VARIABLE,$
FIELDNAME = E1 ,ALIAS= ,ACTUAL = A5 ,USAGE = A5 ,,$
    
```

Note:

- Segments ONE, TWO, and THREE represent a nested group of repeating segments. Fields B1 and B2 occur a fixed number of times, so OCCURS equals 2. Field C1 occurs a certain number of times within each occurrence of fields B1 and B2. The number of times C1 occurs is determined by the value of field B2, which is a counter. In this case, its value is 3 for the first occurrence of Segment TWO and 4 for the second occurrence.
- Segments FOUR and FIVE consist of fields that repeat independently within the parent segment. They have no relationship to each other or to Segment TWO except their common parent, so they represent a parallel group of repeating segments.
- As in the case of Segment THREE, the number of times Segment FOUR occurs is determined by a counter in its parent, A2. In this case, the value of A2 is two.
- The number of times Segment FIVE occurs is variable. This means that all the rest of the fields in the record will be read (all those to the right of the first occurrence of E1) as recurrences of field E1. To ensure that data in the record is interpreted unambiguously, a segment defined as OCCURS=VARIABLE must be at the end of the record. In a data structure diagram, it will be the right-most segment. Note that there can be only one segment defined as OCCURS=VARIABLE for each record type.

The POSITION Attribute

The POSITION attribute is an optional attribute used to describe a structure in which multiply occurring fields with an established number of occurrences are located in the middle of the record. You describe the data source as a hierarchical structure, made up of a parent segment and at least one child segment that contains the multiply occurring fields. The parent segment is made up of whatever singly occurring fields are in the record, as well as one or more alphanumeric fields that appear where the multiply occurring fields appear in the record. The alphanumeric field may be a dummy field that is the exact length of the combined multiply occurring fields. For example, if you have four occurrences of an eight-character field, the length of the field in the parent segment will be 32 characters.

Syntax

How to Specify the Position of a Repeating Field

The POSITION attribute is described in the child segment. It gives the name of the field in the parent segment that specifies the starting position and overall length of the multiply occurring fields. The syntax of the POSITION attribute is

`POSITION = fieldname`

where:

`fieldname`

Is the name of the field in the parent segment that defines the starting position and overall length of the multiple field occurrences.

Example

Specifying the Position of a Repeating Field

Consider the following data structure:

A1	Q1	Q1	Q1	Q1	A2	A3	A4
----	----	----	----	----	----	----	----

In this example, field Q1 repeats four times in the middle of the record. When you describe this structure, you specify a field or fields that occupy the position of the four Q1 fields in the record. You then assign the actual Q1 fields to a descendant, multiply occurring segment. The POSITION attribute, specified in the descendant segment, gives the name of the field in the parent segment that identifies the starting position and overall length of the Q fields.

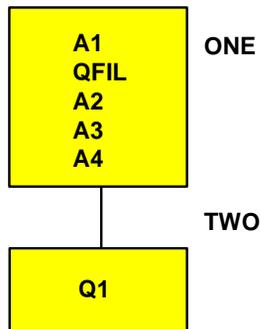
You would use the following Master File to describe this structure:

```
FILENAME = EXAMPLE3, SUFFIX = FIX,$

SEGNAME = ONE, SEGTYPE=S0,$
  FIELDNAME = A1 ,ALIAS= ,USAGE = A14 ,ACTUAL = A14 ,,$
  FIELDNAME = QFIL ,ALIAS= ,USAGE = A32 ,ACTUAL = A32 ,,$
  FIELDNAME = A2 ,ALIAS= ,USAGE = I2 ,ACTUAL = I2 ,,$
  FIELDNAME = A3 ,ALIAS= ,USAGE = A10 ,ACTUAL = A10 ,,$
  FIELDNAME = A4 ,ALIAS= ,USAGE = A15 ,ACTUAL = A15 ,,$

SEGNAME = TWO, SEGTYPE=S0, PARENT = ONE, POSITION = QFIL, OCCURS = 4 ,,$
  FIELDNAME = Q1 ,ALIAS= ,USAGE = D8 ,ACTUAL = D8 ,,$
```

This produces the following structure:



If the total length of the multiply occurring fields is longer than 256, you can use a filler field after the dummy field to make up the remaining length. This is required because the format of an alphanumeric field cannot exceed 256 bytes.

Notice that this structure will work only if you have a fixed number of occurrences of the repeating field. This means the OCCURS attribute of the descendant segment must be of the type OCCURS=*n*. OCCURS=*fieldname* or OCCURS=VARIABLE will not work.

Specifying the ORDER Field

In an OCCURS segment, the order of the data may be significant. For example, the values may represent monthly or quarterly data, but the record itself may not explicitly specify the month or quarter to which the data applies.

If you wish to associate a sequence number with each occurrence of the field, you may define an internal counter field in any OCCURS segment. A value is automatically supplied that defines the sequence number of each repeating group.

Syntax

How to Specify the Sequence of a Repeating Field

The syntax rules for an ORDER field are:

- It must be the last field described in an OCCURS segment.
- The field name is arbitrary.
- The ALIAS is ORDER.
- The USAGE is In, with any appropriate edit options.
- The ACTUAL is I4.

For example:

```
FIELD = ACT_MONTH, ALIAS = ORDER, USAGE = I2MT, ACTUAL = I4, $
```

Order values are 1, 2, 3, and so on, within each occurrence of the segment. The value is assigned prior to any selection tests that might accept or reject the record, and so it can be used in a selection test.

For example, to obtain data for only the month of June, type:

```
SUM AMOUNT...  
WHERE ACT_MONTH IS 6
```

The ORDER field is a virtual field used internally. It does not alter the logical record length (LRECL) of the data source being accessed.

Redefining a Field in a Non-FOCUS Data Source

Support is provided for redefining record fields in non-FOCUS data sources. This allows a field to be described with an alternate layout.

Within the Master File, the redefined fields must be described in a separate unique segment (SEGTYPE=U) using the POSITION=fieldname and OCCURS=1 attributes.

The redefined fields can have any user-defined name. ALIAS names for redefined fields are not required.

Syntax

How to Redefine a Field

```
SEGNAME = segname, SEGTYPE = U, PARENT = parentseg,  
OCCURS = 1, POSITION = fieldname, $
```

where:

segname

Is the name of the segment.

parentseg

Is the name of the parent segment.

fieldname

Is the name of the first field being redefined. Use of the unique segment with redefined fields helps avoid problems with multipath reporting.

A one-to-one relationship is established between the parent record and the redefined segment.

Example

Redefining a VSAM Structure

The following example illustrates redefinition of the VSAM structure described in the COBOL file description where the COBOL FD is:

```
01 ALLFIELDS  
  02 FLD1  PIC X(4)           - this describes alpha/numeric data  
  02 FLD2  PIC X(4)           - this describes numeric data  
  02 RFLD1 PIC 9(5)V99 COMP-3 REDEFINES FLD2  
  02 FLD3  PIC X(8)           - this describes alpha/numeric data  
  
FILE = REDEF, SUFFIX = VSAM, $  
SEGNAME = ONE, SEGTYPE = S0, $  
  GROUP = RKEY, ALIAS = KEY, USAGE = A4      , ACTUAL = A4    , $  
  FIELDNAME = FLD1, ,      USAGE = A4      , ACTUAL = A4    , $  
  FIELDNAME = FLD2, ,      USAGE = A4      , ACTUAL = A4    , $  
  FIELDNAME = FLD3, ,      USAGE = A8      , ACTUAL = A8    , $  
SEGNAME = TWO, SEGTYPE = U, POSITION = FLD2, OCCURS = 1, PARENT = ONE , $  
  FIELDNAME = RFLD1, ,      USAGE = P8.2   , ACTUAL = Z4    , $
```

Reference

Special Considerations for Redefining a Field

- Redefinition is a read-only feature and is used only for presenting an alternate view of the data. It is not used for changing the format of the data.
- A field that is being redefined must be equal in length to the field that it is redefining (same actual length).
- For non-alphanumeric fields, you must know your data. Attempts to print numeric fields that contain alphanumeric data will produce data exceptions or errors converting values. It is recommended that the first definition always be alphanumeric to avoid conversion errors.
- More than one field can be redefined in a segment.

Redefines are supported only for IDMS, IMS, VSAM, DB2, and FIX data sources.

Extra-Large Record Length Support

If the Master File describes a data source with OCCURS segments, and if the longest single record in the data source is larger than 16K bytes, it is necessary to specify a larger record size in advance.

To define the maximum record length, type:

```
SET MAXLRECL = nnnn
```

where *nnnn* is up to 32768.

For example, SET MAXLRECL=12000 allows handling of records that are 12000 bytes long. Once you have entered the SET MAXLRECL command, you can obtain the current value of the MAXLRECL parameter by using the ? SET command.

If the actual record length is longer than specified, retrieval is halted and the actual record length is displayed in hexadecimal notation.

Describing Multiple Record Types

Fixed-format sequential, VSAM, and ISAM data sources can contain more than one type of record. When they do, they can be structured in one of two ways.

- A positional relationship may exist between the various record types, with a record of one type being followed by one or more records containing detailed information about the first record.

If a positional relationship exists between the various record types, with a parent record of one type followed by one or more child records containing detail information about the parent, you describe the structure by defining the parent as the root and the detail segments as descendants.

Some VSAM and ISAM data sources are structured so that descendant records relate to each other through concatenating key fields. That is, the key fields of a parent record serve as the first part of the key of a child record. In such cases, the segment's key fields must be described using a GROUP declaration. Each segment's GROUP key fields will consist of the renamed key fields from the parent segment plus the unique key field from the child record.

- The records have no meaningful positional relationship, and records of varying record types exist independently of each other in the data source.

If the records have no meaningful positional relationship, you have to provide some means for interpreting the type of record that has been read. You do this by creating a dummy root segment for the records.

In order to describe sequential data sources with several types of records, regardless of whether they are logically related, use the PARENT segment attribute and the RECTYPE field attribute. Any complex sequential data source is described as a multi-segment structure.

Key-sequenced VSAM and complex ISAM data sources also use the RECTYPE attribute to distinguish various record types within the data source.

A parent does not always share its RECTYPE with its descendants. It shares some other identifying piece of information, such as the PUBNO in our example. This is the field that should be included in the parent key, as well as all of its descendants' keys, to relate them.

When using the RECTYPE attribute in VSAM or ISAM data sources with group keys, the RECTYPE field can be part of the segment's group key only when it belongs to a segment that has no descendants, or to a segment whose descendants are described with an OCCURS attribute. In the *Describing VSAM Positionally Related Records* on page 5-24, the RECTYPE field is added to the group key in the SERIANO segment, the lowest descendant segment in the chain.

Describing a RECTYPE Field

When a data source contains multiple record types, there must be a field in the records themselves that can be used to differentiate between the record types. You can find information on this field in your existing description of the data source (for example, a COBOL FD statement). This field must appear in the same physical location of each record. For example, columns 79 and 80 could contain a different two-digit code for each unique record type. You describe this identifying field with the field name RECTYPE.

Another technique for redefining parts of records is to use the MAPFIELD and MAPVALUE attributes described in *Describing a Repeating Group Using MAPFIELD* on page 5-31.

Syntax

How to Specify a Record Type Field

The RECTYPE field must fall in the same physical location of each record in the data source or the record will be ignored. The syntax to describe the RECTYPE field is

```
FIELDNAME = RECTYPE, ALIAS = value, USAGE = format, ACTUAL = format , $
```

where:

value

Is the record type in alphanumeric format.

format

Is the data type of the field. In addition to RECTYPE fields in alphanumeric format, RECTYPE fields in packed and integer formats (formats P and I) are supported.

Possible values are:

An (where *n* is 1-256) indicates character data, including letters, digits, and other characters.

In indicates ACTUAL (internal) format binary integers:

I1 = single-byte binary integer.

I2 = half-word binary integer (2 bytes).

I4 = full-word binary integer (4 bytes).

USAGE format can be I1 through I9, depending on the magnitude of the ACTUAL format.

Pn (where *n* is 1-16) indicates packed decimal ACTUAL (internal) format. *n* is the number of bytes, each of which contains two digits, except for the last byte which contains a digit and the sign (+ or -). For example, P6 means 11 digits plus a sign.

If the field contains an assumed decimal point, represent the field with a USAGE format of *Pm.n*, where *m* is the total number of digits, and *n* is the number of decimal places. Thus P11.1 means an eleven-digit number with one decimal place.

Example Specifying the RECTYPE Field

The following field description describes a one-byte packed RECTYPE field containing the value 1:

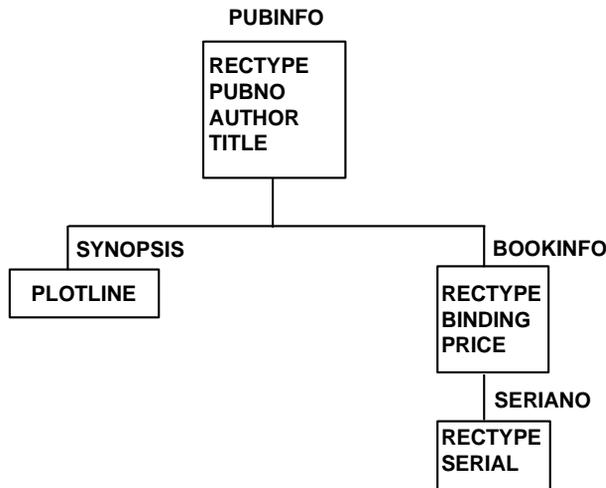
`FIELD = RECTYPE, ALIAS = 1, USAGE = P1, ACTUAL = P1, $`

This field description describes a three-byte alphanumeric RECTYPE field containing the value A34:

`FIELD = RECTYPE, ALIAS = A34, USAGE = A3, ACTUAL = A3,$`

Describing Positionally Related Records

The following diagram shows a more complex version of the library data source discussed previously.



Information that is common to all copies of a given book (the identifying number, the author's name, and its title) has the same record type. They are all assigned to the root segment in the Master File. The synopsis is common to all copies of a given book, but, in this data source, it has been described as a series of repeating fields of ten characters each in order to save space.

The synopsis is assigned to its own subordinate segment with an attribute of OCCURS=VARIABLE in the Master File. Although there are segments that are shown in the diagram to the right of the OCCURS=VARIABLE segment, the OCCURS=VARIABLE segment is the right-most segment within its own record type. Only segments with a RECTYPE that is different from the OCCURS=VARIABLE segment can appear to its right in the structure. Note also that the OCCURS=VARIABLE segment does not have a RECTYPE. This is because the OCCURS=VARIABLE segment is part of the same record as its parent segment.

Binding and price can vary among copies of a given title. For instance, the library may have two different versions of *Pamela*, one a paperback costing \$7.95, the other a hardcover costing \$15.50. These two fields are of a second record type, and are assigned to a descendant segment in the Master File.

Finally, every copy of the book in the library will have its own identifying serial number, which will be described in a field of record type S. In the Master File, this information is assigned to a segment that is a child of the segment containing the binding and price information.

You would use the following Master File to describe this data source:

```
FILENAME = LIBRARY2, SUFFIX = FIX,$

SEGNAME = PUBINFO, SEGTYPE = S0,$
  FIELDNAME = RECTYPE ,ALIAS = P ,USAGE = A1 ,ACTUAL = A1 ,,$
  FIELDNAME = PUBNO ,ALIAS = PN ,USAGE = A10 ,ACTUAL = A10 ,,$
  FIELDNAME = AUTHOR ,ALIAS = AT ,USAGE = A25 ,ACTUAL = A25 ,,$
  FIELDNAME = TITLE ,ALIAS = TL ,USAGE = A50 ,ACTUAL = A50 ,,$
SEGNAME = SYNOPSIS , PARENT = PUBINFO, OCCURS = VARIABLE, SEGTYPE = S0,$
  FIELDNAME = PLOTLINE ,ALIAS = PLOTL ,USAGE = A10 ,ACTUAL = A10 ,,$
SEGNAME = BOOKINFO, PARENT = PUBINFO, SEGTYPE = S0,$
  FIELDNAME = RECTYPE ,ALIAS = B ,USAGE = A1 ,ACTUAL = A1 ,,$
  FIELDNAME = BINDING ,ALIAS = BI ,USAGE = A1 ,ACTUAL = A1 ,,$
  FIELDNAME = PRICE ,ALIAS = PR ,USAGE = D8.2N ,ACTUAL = D8 ,,$
SEGNAME = SERIANO, PARENT = BOOKINFO, SEGTYPE = S0,$
  FIELDNAME = RECTYPE ,ALIAS = S ,USAGE = A1 ,ACTUAL = A1 ,,$
  FIELDNAME = SERIAL ,ALIAS = SN ,USAGE = A15 ,ACTUAL = A15 ,,$
```

Note that each segment, except the OCCURS segment, contains a field named RECTYPE and that the ALIAS for the field contains a unique value for each segment (P, B, and S). If a record in this data source is encountered with a RECTYPE other than P, B, or S, the record will be ignored. The RECTYPE field must fall in the same physical location in each record.

Order of Records in the Data Source

With sequential records, physical order determines parent/child relationships. Every parent record need not have descendants. You can specify how you want data in missing segment instances handled in your reports by using the SET command to change the ALL parameter. The SET command is described in the *Developing Reporting Applications* manual.

In the structure shown in the example in *Describing Positionally Related Records* on page 5-22, if the first record in the data source is not a PUBINFO record, the record is considered to be a child without a parent. Any information allotted to the SYNOPSIS segment will appear in the PUBINFO record. The next record may be a BOOKINFO or even another PUBINFO (in which case the first PUBINFO is assumed to have no descendants). Any SERIANO records are assumed to be descendants of the previous BOOKINFO record. If a SERIANO record follows a PUBINFO record with no intervening BOOKINFO, it is treated as if it has no parent.

Example

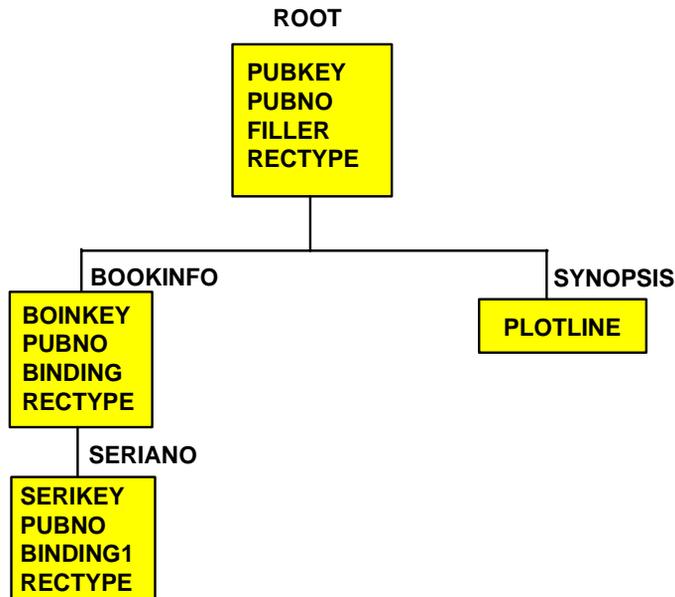
Describing VSAM Positionally Related Records

Consider the following VSAM data source that contains three types of records. The ROOT records have a key that consists of the publisher's number, PUBNO. The BOOKINFO segment has a key that consists of that same publisher's number, plus a hard- or soft-cover indicator, BINDING. The SERIANO segment key consists of the first two elements, plus a record type field, RECTYPE.

```
FILENAME = LIBRARY6, SUFFIX = VSAM,$
SEGNAME = ROOT, SEGTYPE = S0,$
  GROUP=PUBKEY      ,ALIAS=KEY      ,USAGE=A10    ,ACTUAL=A10  ,$
  FIELDNAME=PUBNO  ,ALIAS=PN      ,USAGE=A10    ,ACTUAL=A10  ,$
  FIELDNAME=FILLER ,ALIAS=        ,USAGE=A1     ,ACTUAL=A1   ,$
  FIELDNAME=RECTYPE,ALIAS=1        ,USAGE=A1     ,ACTUAL=A1   ,$
  FIELDNAME=AUTHOR ,ALIAS=AT      ,USAGE=A25    ,ACTUAL=A25  ,$
  FIELDNAME=TITLE  ,ALIAS=TL      ,USAGE=A50    ,ACTUAL=A50  ,$
SEGNAME=BOOKINFO, PARENT=ROOT, SEGTYPE=S0,$
  GROUP=BOINKEY    ,ALIAS=KEY      ,USAGE=A11    ,ACTUAL=A11  ,$
  FIELDNAME=PUBNO1 ,ALIAS=P1       ,USAGE=A10    ,ACTUAL=A10  ,$
  FIELDNAME=BINDING,ALIAS=BI       ,USAGE=A1     ,ACTUAL=A1   ,$
  FIELDNAME=RECTYPE,ALIAS=2        ,USAGE=A1     ,ACTUAL=A1   ,$
  FIELDNAME=PRICE  ,ALIAS=PR       ,USAGE=D8.2N  ,ACTUAL=D8   ,$
SEGNAME=SERIANO, PARENT=BOOKINFO, SEGTYPE=S0,$
  GROUP=SERIKEY    ,ALIAS=KEY      ,USAGE=A12    ,ACTUAL=A12  ,$
  FIELDNAME=PUBNO2 ,ALIAS=P2       ,USAGE=A10    ,ACTUAL=A10  ,$
  FIELDNAME=BINDING1,ALIAS=B1       ,USAGE=A1     ,ACTUAL=A1   ,$
  FIELDNAME=RECTYPE,ALIAS=3        ,USAGE=A1     ,ACTUAL=A1   ,$
  FIELDNAME=SERIAL ,ALIAS=SN       ,USAGE=A15    ,ACTUAL=A15  ,$
SEGNAME=SYNOPSIS, PARENT=ROOT, SEGTYPE=S0, OCCURS=VARIABLE,$
  FIELDNAME=PLOTLINE,ALIAS=PLOTL  ,USAGE=A10    ,ACTUAL=A10  ,
```

Notice that the length of the key fields specified in the USAGE and ACTUAL attributes of a GROUP declaration is the length of the key fields from the parent segments plus the length of the added field of the child segment (RECTYPE field). In the example above, the length of the GROUP key SERIKEY equals the length of PUBNO2 and BINDING1, the group key from the parent segment, plus the length of RECTYPE, the field added to the group key in the child segment. The length of the key increases as you traverse the structure.

In the sample data source, the repetition of the publisher's number as PUBNO1 and PUBNO2 in the descendant segments interrelates the three types of records. The data source can be diagrammed as the following structure:



A typical query might request information on price and call numbers for a specific publisher's number:

```
PRINT PRICE AND SERIAL BY PUBNO
IF PUBNO EQ 1234567890 OR 9876054321
```

Since PUBNO is part of the key, the retrieval can be made quickly and the processing continues. To further speed retrieval you could add search criteria based on the BINDING field, which is also part of the key.

Describing Unrelated Records

Some VSAM and ISAM data sources do not have records that are related to one another. That is, the VSAM or ISAM key of one record type is independent of the keys of other record types. To describe data sources with unrelated records, you define a dummy root segment. You make the record types descendants of a dummy root segment. The following rules apply to the dummy root segment:

- The name of the root segment must be DUMMY.
- It must have only one field with an empty name and alias.
- The USAGE and ACTUAL attributes must both be A1.

All other non-repeating segments must point to the dummy root as their parent. Except for the root, all non-repeating segments must have a RECTYPE and a PARENT attribute and describe the full VSAM/ISAM key. If the data source does not have a key, the group should not be described. RECTYPES may be anywhere in the record.

Example

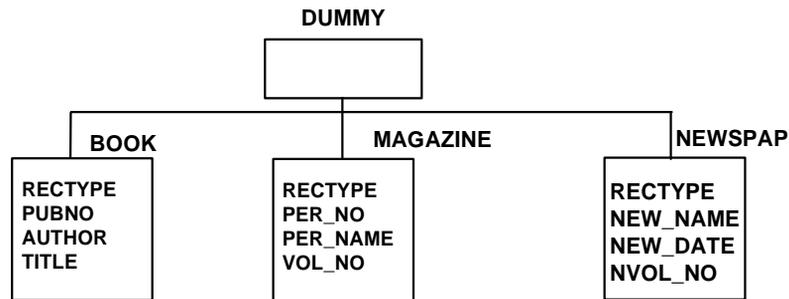
Describing Unrelated Records Using a Dummy Root Segment

The library data source has three types of records: book information, magazine information, and newspaper information. Since book information, magazine information, and newspaper information have nothing in common, these three record types cannot be described as parent records followed by detail records.

The data source might look like this:



A structure such as the following could also describe this data source:



The Master File for the structure in this example is:

```

FILENAME = LIBRARY3, SUFFIX = FIX,$
SEGMENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME=          ,ALIAS=          ,USAGE = A1      ,ACTUAL = A1  ,$
SEGMENT = BOOK, PARENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME = RECTYPE ,ALIAS = B ,USAGE = A1      ,ACTUAL = A1  ,$
  FIELDNAME = PUBNO   ,ALIAS = PN ,USAGE = A10     ,ACTUAL = A10 ,$
  FIELDNAME = AUTHOR  ,ALIAS = AT ,USAGE = A25    ,ACTUAL = A25 ,$
  FIELDNAME = TITLE   ,ALIAS = TL ,USAGE = A50    ,ACTUAL = A50 ,$
  FIELDNAME = BINDING ,ALIAS = BI ,USAGE = A1     ,ACTUAL = A1  ,$
  FIELDNAME = PRICE   ,ALIAS = PR ,USAGE = D8.2N ,ACTUAL = D8  ,$
  FIELDNAME = SERIAL  ,ALIAS = SN ,USAGE = A15    ,ACTUAL = A15 ,$
  FIELDNAME = SYNOPSIS,ALIAS = SY ,USAGE = A150   ,ACTUAL = A150,$
SEGMENT = MAGAZINE, PARENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME = RECTYPE ,ALIAS = M ,USAGE = A1      ,ACTUAL = A1  ,$
  FIELDNAME = PER_NO  ,ALIAS = PN ,USAGE = A10     ,ACTUAL = A10 ,$
  FIELDNAME = PER_NAME,ALIAS = NA ,USAGE = A50     ,ACTUAL = A50 ,$
  FIELDNAME = VOL_NO  ,ALIAS = VN ,USAGE = I2      ,ACTUAL = I2  ,$
  FIELDNAME = ISSUE_NO,ALIAS = IN ,USAGE = I2      ,ACTUAL = I2  ,$
  FIELDNAME = PER_DATE,ALIAS = DT ,USAGE = I6MDY   ,ACTUAL = I6  ,$
SEGMENT = NEWSPAP, PARENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME = RECTYPE ,ALIAS = N ,USAGE = A1      ,ACTUAL = A1  ,$
  FIELDNAME = NEW_NAME,ALIAS = NN ,USAGE = A50     ,ACTUAL = A50 ,$
  FIELDNAME = NEW_DATE,ALIAS = ND ,USAGE = I6MDY   ,ACTUAL = I6  ,$
  FIELDNAME = NVOL_NO ,ALIAS = NV ,USAGE = I2      ,ACTUAL = I2  ,$
  FIELDNAME = ISSUE   ,ALIAS = NI ,USAGE = I2      ,ACTUAL = I2  ,$
  
```

Example

Describing a VSAM Data Source With Unrelated Records

Consider another VSAM data source containing information on our library. This data source has three types of records: book information, magazine information, and newspaper information.

There are two possible structures:

- The RECTYPE is the beginning of the key. The key structure is:

RECTYPE B	Book Code
RECTYPE M	Magazine Code
RECTYPE N	Newspaper Code

The sequence of records is:

Book
Book
Magazine
Magazine
Newspaper
Newspaper

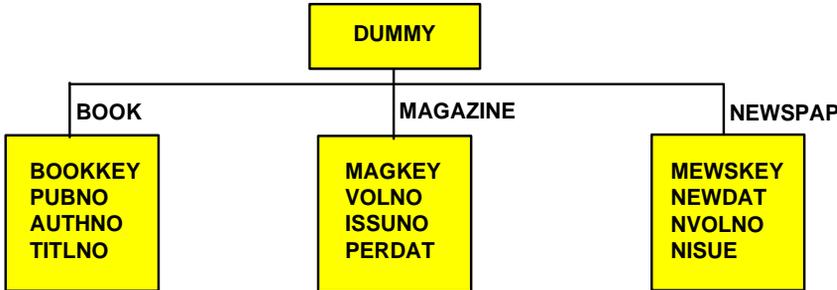
Note the difference between the use of the RECTYPE here and its use when the records are positionally related. In this case, the codes are unrelated and the database designer has chosen to accumulate the records by type first (all the book information together, all the magazine information together, and all the newspaper information together), so the RECTYPE may be the initial part of the key.

- The RECTYPE is not in the beginning of the key or is outside of the key. The key structure is:

Book Code
Magazine Code
Newspaper Code

The sequence of record types in the data source can be arbitrary.

Both types of file structure can be represented by the following structure:



Example

Describing a Key and a Record Type for a VSAM Data Source With Unrelated Records

```
FILE=LIBRARY7, SUFFIX=VSAM,$
SEGMENT=DUMMY,$
  FIELDNAME=, ALIAS=, USAGE=A1, ACTUAL=A1,$
SEGMENT=BOOK, PARENT=DUMMY, SEGTYPE=S0,$
GROUP=BOOKKEY, ALIAS=KEY, USAGE=A11, ACTUAL=A11,$
  FIELDNAME=PUBNO, ALIAS=PN, USAGE=A3, ACTUAL=A3,$
  FIELDNAME=AUTHNO, ALIAS=AN, USAGE=A3, ACTUAL=A3,$
  FIELDNAME=TITLNO, ALIAS=TN, USAGE=A4, ACTUAL=A4,$
  FIELDNAME=RECTYPE, ALIAS=B, USAGE=A1, ACTUAL=A1,$
  FIELDNAME=AUTHOR, ALIAS=AT, USAGE=A25, ACTUAL=A25,$
  FIELDNAME=TITLE, ALIAS=TL, USAGE=A50, ACTUAL=A50,$
  FIELDNAME=BINDING, ALIAS=BI, USAGE=A1, ACTUAL=A1,$
  FIELDNAME=PRICE, ALIAS=PR, USAGE=D8.2N, ACTUAL=D8,$
  FIELDNAME=SERIAL, ALIAS=SN, USAGE=A15, ACTUAL=A15,$
  FIELDNAME=SYNOPSIS, ALIAS=SY, USAGE=A150, ACTUAL=A150,$
SEGMENT=MAGAZINE, PARENT=DUMMY, SEGTYPE=S0,$
GROUP=MAGKEY, ALIAS=KEY, USAGE=A11, ACTUAL=A11,$
  FIELDNAME=VOLNO, ALIAS=VN, USAGE=A2, ACTUAL=A2,$
  FIELDNAME=ISSUNO, ALIAS=IN, USAGE=A2, ACTUAL=A2,$
  FIELDNAME=PERDAT, ALIAS=DT, USAGE=A6, ACTUAL=A6,$
  FIELDNAME=RECTYPE, ALIAS=M, USAGE=A1, ACTUAL=A1,$
  FIELDNAME=PER_NAME, ALIAS=PRN, USAGE=A50, ACTUAL=A50,$
SEGMENT=NEWSPAP, PARENT=DUMMY, SEGTYPE=S0,$
GROUP=NEWSKEY, ALIAS=KEY, USAGE=A11, ACTUAL=A11,$
  FIELDNAME=NEWDAT, ALIAS=ND, USAGE=A6, ACTUAL=A6,$
  FIELDNAME=NVOLNO, ALIAS=NV, USAGE=A2, ACTUAL=A2,$
  FIELDNAME=NISSUE, ALIAS=NI, USAGE=A2, ACTUAL=A2,$
  FIELDNAME=RECTYPE, ALIAS=N, USAGE=A1, ACTUAL=A1,$
  FIELDNAME=NEWNAME, ALIAS=NN, USAGE=A50, ACTUAL=A50,$
```

Using a Generalized Record Type

If your fixed-format sequential, VSAM, or ISAM data source has multiple record types that share the same layout, you can specify a single generalized segment that describes all record types having the common layout. By using a generalized segment—also known as a generalized RECTYPE—instead of one segment per record type, you reduce the number of segments you need to describe in the Master File.

When you use a generalized segment, you identify RECTYPE values using the ACCEPT attribute. You can assign any value you wish to the ALIAS attribute.

Syntax

How to Specify a Generalized Record Type

```
FIELDNAME = RECTYPE, ALIAS = alias, USAGE = format, ACTUAL = format,
ACCEPT = {list|range} , $
```

where:

RECTYPE

Is the required field name.

Note: Since the field name, RECTYPE, may not be unique across segments, you should not use it in this way unless you qualify it. An alias is not required; you may leave it blank.

alias

Is any valid alias specification. You can specify a unique name as the alias value for the RECTYPE field only if you use the ACCEPT attribute. The alias can then be used in a TABLE request as a display field, a sort field, or in selection tests using either WHERE or IF.

list

Is a list of one or more lines of specific RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Each item in the list must be separated by either a blank or the keyword OR. If the list contains embedded blanks or commas, it must be enclosed within single quotation marks. The list may contain a single RECTYPE value.

For example:

```
FIELDNAME = RECTYPE, ALIAS = TYPEABC, USAGE = A1,
ACTUAL = A1, ACCEPT = A OR B OR C, $
```

range

Is a range of one or more lines of RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If the range contains embedded blanks or commas, it must be enclosed within single quotation marks.

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order. For example:

```
FIELDNAME = RECTYPE, ALIAS = ACCTREC, USAGE = P3,
ACTUAL = P2, ACCEPT = 100 TO 200, $
```

Example Using a Generalized Record Type

To illustrate the use of the generalized record type in VSAM Master Files, consider the following record layouts in the DOC data source. Record type DN is the root segment and contains the document number and title. Record types M, I, and C contain information about manuals, installation guides, and course guides, respectively. Notice that record types M and I have the same layout.

Record Type DN:

```
---KEY---
+-----+
+
DOCID  FILLER          RECTYPE  TITLE
+-----+
+
```

Record Type M:

```
-----KEY-----
+-----+
+
MDOCID  MDATE          RECTYPE  MRELEASE      MPAGES  FILLER
+-----+
-
```

Record Type I:

```
-----KEY-----
+-----+
+
IDOCID  IDATE          RECTYPE  IRELEASE      IPAGES  FILLER
+-----+
+
```

Record Type C:

```
-----KEY-----
+-----+
+
CRSEDOC  CDATE          RECTYPE  COURSENUM  LEVEL  CPAGES  FILLER
+-----+
+
```

Without the ACCEPT attribute, each of the four record types must be described as separate segments in the Master File. In particular, a unique set of field names must be provided for record type M and for record type I, although they have the same layout. The generalized RECTYPE capability enables you to code just one set of field names that will apply to the record layout for both record type M and record type I. The ACCEPT attribute can be used for any RECTYPE specification, even when there is only one acceptable value.

```

FILENAME=DOC2, SUFFIX=VSAM,$
SEGNAME=ROOT, SEGTYPE=SO,$
GROUP=DOCNUM, ALIAS=KEY, A5, A5,$
  FIELD=DOCID, ALIAS=SEQNUM, A5, A5, $
  FIELD=FILLER, ALIAS=, A5, A5, $
  FIELD=RECTYPE, ALIAS=DOCRECORD, A3, A3, ACCEPT =DN,$
  FIELD=TITLE, ALIAS=, A18, A18, $
SEGNAME=MANUALS, PARENT=ROOT, SEGTYPE=SO,$
GROUP=MDOCNUM,ALIAS=KEY, A10, A10,$
  FIELD=MDOCID, ALIAS=MSEQNUM, A5, A5, $
  FIELD=MDATE, ALIAS=MPUBDATE, A5, A5, $
  FIELD=RECTYPE, ALIAS=MANUAL, A3, A3, ACCEPT = M OR I,$
  FIELD=MRELEASE, ALIAS=, A7, A7, $
  FIELD=MPAGES, ALIAS=, I5, A5, $
  FIELD=FILLER, ALIAS=, A6, A6, $
SEGNAME=COURSES, PARENT=ROOT, SEGTYPE=SO,$
GROUP=CRSEDOC, ALIAS=KEY, A10, A10,$
  FIELD=CDOCID, ALIAS=CSEQNUM, A5, A5, $
  FIELD=CDATE, ALIAS=CPUBDATE, A5, A5, $
  FIELD=RECTYPE, ALIAS=COURSE, A3, A3, ACCEPT = C,$
  FIELD=COURSENUM, ALIAS=CNUM, A4, A4, $
  FIELD=LEVEL, ALIAS=, A2, A2, $
  FIELD=CPAGES, ALIAS=, I5, A5, $
  FIELD=FILLER, ALIAS=, A7, A7, $

```

Using an ALIAS in a Report Request

You can include an alias for the RECTYPE field if you use the ACCEPT attribute to specify one or more RECTYPE values in the Master File. This enables you to use the alias in a report request as a display field, as a sort field, or in selection tests using either WHERE or IF.

Example

Using a RECTYPE Value in a Display Command

You can display the RECTYPE values by including the alias as a display field. In this example, the alias MANUAL displays the RECTYPE values M and I:

```
TABLE FILE DOC
PRINT MANUAL MRELEASE MPAGES
BY DOCID BY TITLE BY MDATE
END

PAGE 1
```

DOCID	TITLE	MDATE	RECTYPE	MRELEASE	MPAGES
----	-----	-----	-----	-----	-----
40001	FOCUS USERS MANUAL	8601	M	5.0	1800
		8708	M	5.5	2000
40057	MVS INSTALL GUIDE	8806	I	5.5.3	66
		8808	I	5.5.4	66
40114	CMS INSTALL GUIDE	8806	I	5.5.3	58
		8808	I	5.5.4	58

Example

Using a RECTYPE Value in a WHERE Test

You can use the alias in a WHERE test to display a subset of records.

```
TABLE FILE DOC
PRINT MANUAL MRELEASE MPAGES
BY DOCID BY TITLE BY MDATE
WHERE MANUAL EQ 'I'
END

PAGE 1
```

DOCID	TITLE	MDATE	RECTYPE	MRELEASE	MPAGES
----	-----	-----	-----	-----	-----
40057	MVS INSTALL GUIDE	8806	I	5.5.3	66
		8808	I	5.5.4	66
40114	CMS INSTALL GUIDE	8806	I	5.5.3	58
		8808	I	5.5.4	58

Combining Multiply Occurring Fields and Multiple Record Types

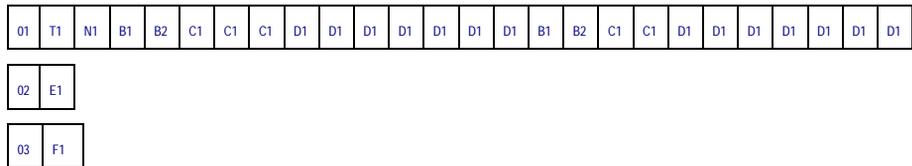
You can have two types of descendant segments in a single fixed-format sequential, VSAM, or ISAM data source:

- Descendant segments consisting of multiply occurring fields,
- Additional descendant segments consisting of multiple record types.

Describing a Multiply Occurring Field and Multiple Record Types

In the data structure shown below, the first record—of type 01—contains several different sequences of repeating fields, all of which must be described as descendant segments with an OCCURS attribute. The data source also contains two separate records, of types 02 and 03, which contain information that is related to that in record type 01.

The relationship between the records of various types is expressed as parent-child relationships. The children that contain record types 02 and 03 do not have an OCCURS attribute. They are distinguished from their parent by the field declaration where field=RECTYPE.



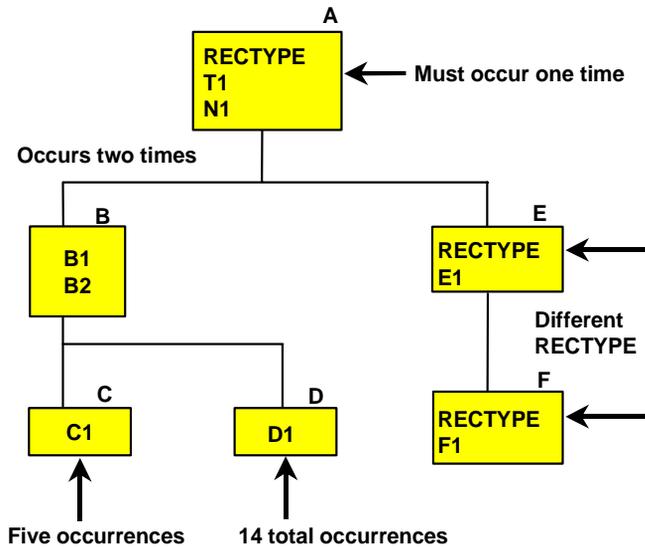
The data source description for this data source is:

```

FILENAME = EXAMPLE1, SUFFIX = FIX,$
SEGNAME = A, SEGTYPE=S0,$
  FIELDNAME = RECTYPE ,ALIAS = 01 ,USAGE = A2 ,ACTUAL = A2 ,$
  FIELDNAME = T1 ,ALIAS = ,USAGE = A2 ,ACTUAL = A1 ,$
  FIELDNAME = N1 ,ALIAS = ,USAGE = A1 ,ACTUAL = A1 ,$
SEGNAME = B, PARENT = A, OCCURS = VARIABLE, SEGTYPE=S0,$
  FIELDNAME = B1 ,ALIAS = ,USAGE = I2 ,ACTUAL = I2 ,$
  FIELDNAME = B2 ,ALIAS = ,USAGE = I2 ,ACTUAL = I2 ,$
SEGNAME = C, PARENT = B, OCCURS = B1, SEGTYPE=S0,$
  FIELDNAME = C1 ,ALIAS = ,USAGE = A1 ,ACTUAL = A1 ,$
SEGNAME = D, PARENT = B, OCCURS = 7, SEGTYPE=S0,$
  FIELDNAME = D1 ,ALIAS = ,USAGE = A1 ,ACTUAL = A1 ,$

SEGNAME = E, PARENT = A, SEGTYPE=S0,$
  FIELDNAME = RECTYPE ,ALIAS = 02 ,USAGE = A2 ,ACTUAL = A2 ,$
  FIELDNAME = E1 ,ALIAS = ,USAGE = A1 ,ACTUAL = A1 ,$
SEGNAME = F, PARENT = E, SEGTYPE=S0,$
  FIELDNAME = RECTYPE ,ALIAS = 03 ,USAGE = A2 ,ACTUAL = A2 ,$
  FIELDNAME = F1 ,ALIAS = ,USAGE = A1 ,ACTUAL = A1 ,$
  
```

It produces the following data structure:



Segments A, B, C, and D all belong to the same record type. Segments E and F each are stored as separate record types.

Note:

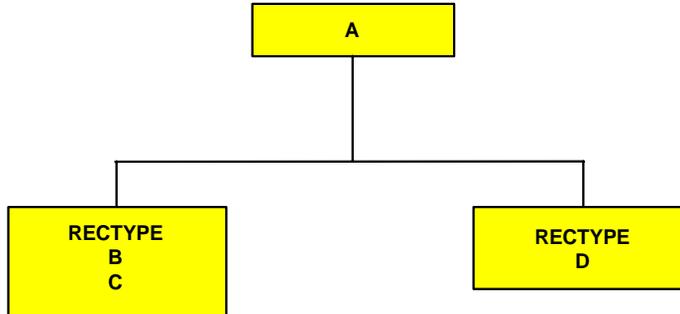
- Segments A, E, and F are different records, related through their record types. The record type attribute consists of certain prescribed values and is stored in a fixed location in the records. The records are expected to be retrieved in a given order. If the first record does not have a RECTYPE of 01, the record is considered to be a child without a parent. The next record can have a RECTYPE of either 01 (in which case the first record is considered to have no descendants except the OCCURS descendants) or 02. A record with a RECTYPE of 03 can follow only a record with a RECTYPE of 02 (its parent).
- The OCCURS descendants all belong to the record whose RECTYPE is 01. (This is not a necessary condition; records of any type can have OCCURS descendants.) Note that the OCCURS=VARIABLE segment, Segment B, is the right-most segment within its own record type. If you look at the data structure, the pattern that makes up Segment B and its descendants (the repetition of fields B1, B2, C1, and D1) extends from the first mention of fields B1 and B2 to the end of the record.
- Although fields C1 and D1 appear in separate segments, they are actually part of the repeating pattern that makes up the OCCURS=VARIABLE segment. Since they occur multiple times within Segment B, they are each assigned to their own descendant segment. The number of times field C1 occurs depends on the value of field B2. In the example, the first value of field B2 is 3, the second, 2. Field D1 occurs a fixed number of times, 7.

A VSAM Repeating Group With RECTYPES

Suppose you want to describe a data source that, schematically, looks like this:

A	RECTYPE B C	RECTYPE B C
A	RECTYPE D	RECTYPE D

You need to describe three segments in your Master File, with A as the root segment, and segments for B, C, and D as two descendant OCCURS segments for A:



Each of the two descendant OCCURS segments in this example depends on the RECTYPE indicator that appears for each occurrence.

All the rules of syntax for using RECTYPE fields and OCCURS segments also apply to RECTYPES within OCCURS segments.

Since each OCCURS segment depends on the RECTYPE indicator for its evaluation, the RECTYPE must appear at the start of each OCCURS segment. This allows very complex data sources to be described, including those with nested and parallel repeating groups that depend on RECTYPES.

Example

Describing a VSAM Repeating Group With RECTYPES

In this example, B/C, and D represent a nested repeating group, and E represents a parallel repeating group.

A	RECTYPE B C	RECTYPE D	RECTYPE E	RECTYPE E
---	-------------	-----------	-----------	-----------

```

FILENAME=SAMPLE , SUFFIX=VSAM , $
SEGNAME=ROOT , SEGTYPE=S0 , $
  GROUP=GRPKEY , ALIAS=KEY , USAGE=A8 , ACTUAL=A8 , $
  FIELD=FLD000 , E00 , A08 , A08 , $
  FIELD=A_DATA , E01 , A02 , A02 , $
SEGNAME=SEG001 , PARENT=ROOT , OCCURS=VARIABLE , SEGTYPE=S0 , $
  FIELD=RECTYPE , A01 , A01 , ACCEPT=B OR C , $
  FIELD=B_OR_C_DATA , E02 , A08 , A08 , $
SEGNAME=SEG002 , PARENT=SEG001 , OCCURS=VARIABLE , SEGTYPE=S0 , $
  FIELD=RECTYPE , D , A01 , A01 , $
  FIELD=D_DATA , E03 , A07 , A07 , $
SEGNAME=SEG003 , PARENT=ROOT , OCCURS=VARIABLE , SEGTYPE=S0 , $
  FIELD=RECTYPE , E , A01 , A01 , $
  FIELD=E_DATA , E04 , A06 , A06 , $
  
```

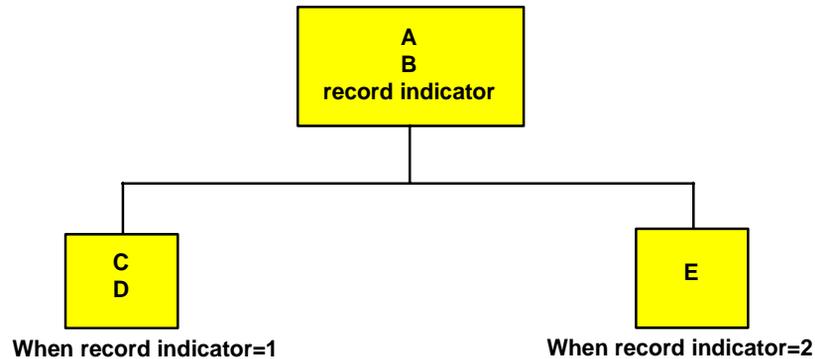
Describing a Repeating Group Using MAPFIELD

In another possible combination of record indicator and OCCURS, a record contains a record indicator that is followed by a repeating group. In this case, the record indicator is in the fixed portion of the record, not in each occurrence. Schematically, the record would appear like this:

A B	record indicator (1)	C D	C D	C D
A B	record indicator (2)	E E		

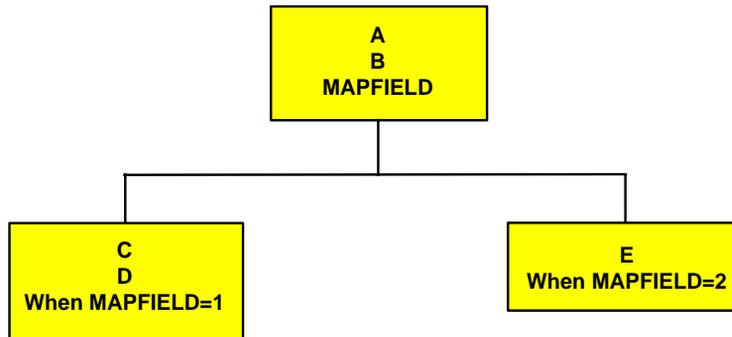
The first record contains header information, values for A and B, followed by an OCCURS segment of C and D that was identified by its preceding record indicator. The second record has a different record indicator and contains a different repeating group, this time for E.

The following diagram illustrates this relationship.



Since the OCCURS segments are identified by the record indicator rather than the parent A/B segment, you must use the keyword MAPFIELD. MAPFIELD identifies a field in the same way RECTYPE does, but since the OCCURS segments will each have their own values for MAPFIELD, the value of MAPFIELD is associated with each OCCURS segment by means of a complementary field named MAPVALUE.

The following diagram illustrates this relationship:



MAPFIELD is assigned as the ALIAS of the field that will be the record indicator. You can give this field any name.

Syntax

How to Describe a Repeating Group With MAPFIELD

`FIELD = name, ALIAS = MAPFIELD, USAGE = format, ACTUAL = format,$`

where:

`name`

The name you choose to provide for this field.

`ALIAS`

MAPFIELD is assigned as the alias of the field that will be the RECTYPE indicator.

`USAGE`

Follows the usual field format.

`ACTUAL`

Follows the usual field format.

The descendant segment values depend on the value of the MAPFIELD. They are described as separate segments, one for each possible value of MAPFIELD, and all descending from the segment that has the MAPFIELD. A special field, MAPVALUE, is described as the last field in these descendant segments after the ORDER field, if one has been used. The actual MAPFIELD value is supplied as the ALIAS of MAPVALUE.

Syntax

How to Use MAPFIELD for a Descendant Repeating Segment in a Repeating Group

```
FIELD = MAPVALUE, ALIAS = alias, USAGE = format, ACTUAL = format,  
ACCEPT = {list|range} , $
```

where:

MAPVALUE

Indicates that the segment depends on a MAPFIELD in its parent segment.

alias

Is the primary MAPFIELD identifier. If there is an ACCEPT list, this value is any value in the ACCEPT list or range.

USAGE

Is the same format as the MAPFIELD format in the parent segment.

ACTUAL

Is the same format as the MAPFIELD format in the parent segment.

list

Is the list of one or more lines of specified MAPFIELD values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Each item in the list must be separated by either a blank or the keyword OR. If the list contains embedded blanks or commas, it must be enclosed within single quotation marks ('). The list may contain a single MAPFIELD value.

For example:

```
FIELDNAME = MAPFIELD, ALIAS = A, USAGE = A1, ACTUAL = A1,  
ACCEPT = A OR B OR C, $
```

range

Is a range of one or more lines of MAPFIELD values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If the range contains embedded blanks or commas, it must be enclosed in single quotation marks (').

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order.

Example Using MAPFIELD and MAPVALUE

Using the sample data source at the beginning of this section, the Master File for this data source looks like this:

```

FILENAME=EXAMPLE , SUFFIX=FIX , $
SEGNAME=ROOT , SEGTYPE=S0 , $
FIELD =A , ,A14 ,A14 , $
FIELD =B , ,A10 ,A10 , $
FIELD =FLAG ,MAPFIELD ,A01 ,A01 , $
SEGNAME=SEG001 , PARENT=ROOT , OCCURS=VARIABLE , SEGTYPE=S0 , $
FIELD =C , ,A05 ,A05 , $
FIELD =D , ,A07 ,A07 , $
FIELD =MAPVALUE ,1 ,A01 ,A01 , $
SEGNAME=SEG002 , PARENT=ROOT , OCCURS=VARIABLE , SEGTYPE=S0 , $
FIELD =E , ,D12.2 ,D8 , $
FIELD =MAPVALUE ,2 ,A01 ,A01 , $

```

Note: MAPFIELD can only exist on an OCCURS segment that has not been re-mapped. This means that this segment definition cannot contain POSITION=*fieldname*.

MAPFIELD and MAPVALUE may be used with suffix=FIX and suffix=VSAM data sources.

VSAM Data and Index Buffers

Two SET commands make it possible to establish DATA and INDEX buffers for processing VSAM data sources online.

The AMP sub-parameters BUFND and BUFNI allow MVS BATCH users to enhance the I/O efficiency of TABLE, TABLEF, MODIFY, and JOIN against VSAM data sources by holding frequently used VSAM Control Intervals in memory, rather than on physical DASD. By reducing the number of physical Input/Output operations, job throughput is improved. The new SET commands allow users (in CMS, MVS/TSO, and MSO) to realize similar performance gains in interactive sessions. In general, BUFND (data buffers) increase the efficiency of physical sequential reads, whereas BUFNI (index buffers) are most beneficial in JOIN or KEYED access operations.

Syntax

How to Establish VSAM Data and Index Buffers

```

{MVS|CMS} VSAM SET BUFND {n|8}
{MVS|CMS} VSAM SET BUFNI {n|1}

```

where:

n

Is the number of data or index buffers. The default values are BUFND=8, BUFNI=1 (eight data buffers and one index buffer).

To determine how many buffers are in effect at any time, issue the query:

```
{MVS|CMS} VSAM SET ?
```

A VSAM Alternate Index

The use of alternate indexes (keys) with VSAM key-sequenced data sources is supported. A key-sequenced VSAM data source consists of two components: an index component and a data component. The data component contains the actual data records, while the index component is the key used to locate the data records in the data source. Together, these two components are referred to as the base cluster.

An alternate index is a separate, additional index structure that allows you to access records in a KSDS VSAM data source based on a key other than the data source's primary key. For instance, you may usually use a personnel data source sequenced by Social Security number, but have an occasional need to have the records retrieved sorted by job description. The job description field might be described as an alternate index. An alternate index must be related to the base cluster it describes by a path, which is stored in a separate data source.

The alternate index is a VSAM structure and is, consequently, created and maintained in the VSAM environment. It can, however, be described in your Master File, so that you can take advantage of the benefits of an alternate index.

The primary benefit of these indexes is improved efficiency. You can use it as an alternate, more efficient, retrieval sequence or you can take advantage of its potential indirectly, with screening tests (IF...LT, IF...LE, IF...GT, IF...GE, IF...EQ, IF...FROM...TO, IF...IS), which are translated into direct reads against the alternate index. You can also join data sources with the JOIN command through this alternate index.

It is not necessary to explicitly identify the indexed view in order to take advantage of the alternate index. An alternate index is automatically used when described in the Master File.

To take advantage of a specific alternate index during a TABLE request, provide an IF or WHERE test on the alternative index field that meets the above criteria. For example:

```
TABLE FILE CUST
PRINT SSN
WHERE LNAME EQ 'SMITH'
END
```

As you will see in the Master File in *Describing a VSAM Alternate Index* on page 5-31, the LNAME field is defined as an alternate index field. The records in the data source will be retrieved according to their last names, and certain IF screens on the field LNAME will result in direct reads. Note that if the alternate index field name is omitted, the primary key (if there is any) will be used for a sequential or a direct read, and the alternate indexes will be treated as regular fields.

Alternate indexes must be described in the Master File as fields with FIELDTYPE=I. The ALIAS of the alternate index field must be the file name allocated to the corresponding path name. Alternate indexes can be described as GROUPS if they consist of portions with dissimilar formats. Remember that the ALIAS=KEY must be used to describe the primary key.

Only one record type can be referred to in the request when alternate indexes are used, but the number of OCCURS segments is unrestricted.

Note that the path name in the allocation will be different from both the cluster name and the alternate index name.

If you are not sure of the path names and alternate indexes associated with a given base cluster, you can use the IDCAMS utility. (See the IBM manual entitled *Using VSAM Commands and Macros* for details.)

Example

Describing a VSAM Alternate Index

Consider the following example:

```
FILENAME = CUST, SUFFIX = VSAM,$
SEGNAME = ROOT, SEGTYPE = S0,$
GROUP = G, ALIAS = KEY, A10, A10,$
FIELD = SSN, SSN, A10, A10,$
FIELD = FNAME, DD1, A10, A10, FIELDTYPE=I,$
FIELD = LNAME, DD2, A10, A10, FIELDTYPE=I,$
```

In this example, SSN is a primary key and FNAME and LNAME are alternate indexes. The path data set must be allocated to the ddname specified in ALIAS= of your alternate index field. In this Master File, ALIAS=DD1 and ALIAS=DD2 would each have an allocation pointing to the path data set. FNAME and LNAME must have INDEX=I or FIELDTYPE=I coded in the Master File. CUST must be allocated to the base cluster.

Example Using IDCAMS

The following example demonstrates how to use IDCAMS to find the alternate index and path names associated with a base cluster named CUST.DATA:

First, find the alternate index names (AIX) associated with the given cluster.

```
IDCAMS input:
LISTCAT CLUSTER ENTRIES(CUST.DATA) ALL
```

```
IDCAMS output (fragments):
CLUSTER ----- CUST.DATA
ASSOCIATIONS
AIX ----- CUST.INDEX1
AIX ----- CUST.INDEX2
```

This gives you the names of the alternate indexes (AIX): CUST.INDEX1 and CUST.INDEX2.

Next, find the path names associated with the given AIX name.

```
IDCAMS input:
LISTCAT AIX ENTRIES (CUST.INDEX1 CUST.INDEX2) ALL
```

```
IDCAMS output (fragments):
AIX -----CUST.INDEX1
ASSOCIATIONS
CLUSTER -- CUST.DATA
PATH -----CUST.PATH1
AIX -----CUST.INDEX2
ASSOCIATIONS
CLUSTER -- CUST.DATA
PATH -----CUST.PATH2
```

This gives you the path names: CUST.PATH1 and CUST.PATH2.

This information along with the TSO DDNAME command may be used to ensure the proper allocation of your alternate index.

CHAPTER 6

Describing a FOCUS Data Source

Topics:

- Designing a FOCUS Data Source
- Describing a Single Segment
- The ACCEPT Attribute
- The INDEX Attribute

The following topics cover data description topics unique to FOCUS data sources:

- **Design tips.** Provides some suggestions for people who are designing a new FOCUS data source or changing the design of an existing data source.
- **Describing segments.** Contains information about Master File segment declarations for FOCUS data sources, including defining segment relationships, keys, and sort order using the SEGTYPE attribute, and storing segments in different locations using the LOCATION attribute. Chapter 7, *Defining a Join in a Master File*, explains how to define static and dynamic joins in a Master File.
- **Describing fields.** Contains information about Master File field declarations for FOCUS data sources, including the FIND option of the ACCEPT attribute; indexing fields using the INDEX attribute; and the internal storage requirements of each data type defined by the FORMAT attribute, and of null values described by the MISSING attribute.

Designing a FOCUS Data Source

The database management system enables you to create sophisticated hierarchical data structures. The following sections provide information to help you design an effective and efficient FOCUS data source and tell you how you can change the design after the data source has been created.

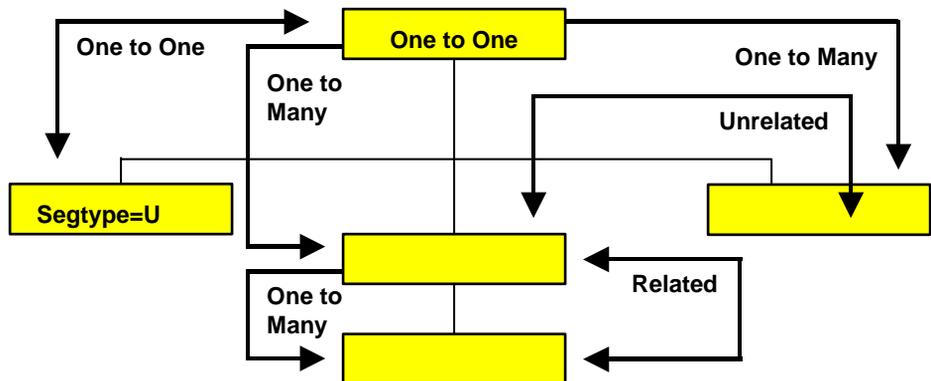
Data Relationships

The primary consideration when designing a data source is the set of relationships among the various fields. Before you create the Master File, you may wish to draw a diagram of these relationships. Is a field related to any other fields? If so, is it a one-to-one or a one-to-many relationship? If any of the data already exists in another data source, can that data source be joined to this one?

In general, you can use the following guidelines:

- All information that occurs once for a given record should be placed in the root segment or a unique child segment.
- Any information that can be retrieved from a joined data source should, in most cases, be retrieved in this way, and not redundantly maintained in two different data sources.
- Any information that has a many-to-one relationship with the information in a given segment should be stored in a descendant of that segment.
- Related data in child segments should be stored in the same path; unrelated data should be placed in different paths.

The following illustration summarizes the rules for data relationship considerations:



Join Considerations

If you plan to join one segment to another, remember that both the host and cross-referenced fields must have the same format, and the cross-referenced field must be indexed using the INDEX attribute. In addition, for a cross-reference in a Master File, the host and cross-referenced fields must share the same name—that is, the name of both fields, or the alias of both fields must be identical, or else the name of one field must be identical to the alias of the other.

General Efficiency Considerations

A FOCUS data source is processing by reading the root segment first, then traversing the hierarchy to satisfy your query. The smaller you make the root segment, the more root segment instances can be read at one time, and the faster records can be selected to process a query.

You can also improve record substitution efficiency by setting AUTOPATH. AUTOPATH is the automation of TABLE FILE *ddname.fieldname* syntax, where field name is not indexed and physical retrieval starts at the field name segment. AUTOPATH is described in the *Developing Reporting Applications* manual.

As with most information processing issues, there is a trade-off when designing an efficient FOCUS data source: you need to balance the desire to speed up record *retrieval*, by reducing the size of the root segment, against the need to speed up record selection, by placing fields used in record selection tests as high in the data structure as possible. The segment location of fields used in WHERE or IF tests has significant implications to the processing efficiency of a request. When a field fails a record selection test, there is no additional processing to that segment instance or its descendants. The higher the selection fields are in a data structure, the fewer the number of segments that need to be read to determine a record's status, and the greater the number of segments that will be ignored—consuming no processing time—when a record fails a selection test.

After you have designed and created a data source, if you want to select records based on fields that are low in the data structure, you can rotate the data structure to place those fields temporarily higher by using an alternate view. Alternate views are discussed in Chapter 3, *Describing a Group of Fields*. Using alternate views in report requests is discussed in the *Creating Reports With WebFOCUS Language* manual.

The following guidelines will help you to design an efficient data structure:

- Limit the information in the root segment to what is necessary to identify the record and to what is used often in screening conditions.
- Avoid unnecessary key fields. Segments with a SEGTYPE of S1 will be processed much more efficiently than those with, for example, a SEGTYPE of S9.
- Use segments with a SEGTYPE of SH1 when adding and maintaining data in date sequence. A SEGTYPE of SH1 puts each new record at the beginning of the data source, not at the end.
- If a segment contains fields frequently used in record selection tests, keep the segment small by limiting it to key fields, selection fields, and other fields frequently used in reports.
- Index fields on which you perform frequent searches of unique instances. When you specify that a field is indexed, a table of data values and their corresponding physical locations in the data source is constructed and maintained. Thus, indexing a field speeds retrieval.

You can index any field you want, although it is advisable to limit the number of indexes in a data source since each index requires additional storage space and maintenance overhead. You will need to weigh the increase in speed against the increase in space.

Changing a FOCUS Data Source

Once you have designed and created a FOCUS data source, you can change some of its characteristics simply by editing the corresponding attribute in the Master File. The documentation for each attribute specifies whether it can be edited after the data source has been created.

Some characteristics whose attributes cannot be edited *can* be changed if you rebuild the data source using the REBUILD facility, as described in Chapter 10, *Creating and Rebuilding a Data Source*. You can also use REBUILD to add new fields to a data source.

Describing a Single Segment

You can code LOCATION segments in a Master File to expand the file size by pointing to another physical file location. LOCATION is discussed in Chapter 3, *Describing a Group of Fields*.

Three additional segment attributes that describe joins between FOCUS segments, CRFILE, CRKEY, and CRSEGNAME, are described in Chapter 7, *Defining a Join in a Master File*.

Maximum Number of Segments

The number of segments cannot exceed 64.

Describing Keys, Sort Order, and Segment Relationships: SEGTYPE

FOCUS data sources use the SEGTYPE attribute to describe a segment's key fields and sort order, as well as the relationship of the segment to its parent.

The SEGTYPE attribute is also used with SUFFIX=FIX data sources to indicate a logical key sequence for that data source. SEGTYPE is discussed in Chapter 3, *Describing a Group of Fields*.

Syntax

How to Describe a Segment

The syntax of the SEGTYPE attribute when used for a FOCUS data source is:

`SEGTYPE = segtype`

Valid values are:

`SH[n]`

Indicates that the segment's instances are sorted from the highest value to the lowest value, based on the value of the first *n* fields in the segment. *n* can be any number from 1 to 99; if you do not specify it, it defaults to 1.

`S[n]`

Indicates that the segment's instances are sorted from the lowest value to the highest value, based on the value of the first *n* fields in the segment. *n* can be any number from 1 to 255; if you do not specify it, it defaults to 1.

`S0`

Indicates that the segment has no key field and is therefore not sorted. New instances are added to the end of the segment chain. Any search starts at the current position.

S0 segments are often used to store text for applications where the text will need to be retrieved in the order entered and the application does not need to search for particular instances.

b (blank)

Indicates that the segment has no key field and is therefore not sorted. New instances are added to the end of the segment chain. Any search starts at the beginning of the segment chain.

SEGTYPE = **b** segments are often used in situations where there are very few segment instances and the information stored in the segment does not include a field that can serve as a key.

Note that a root segment cannot be a **b** segment.

U

Indicates that the segment is unique—that is, it has a one-to-one relationship to its parent. Note that a unique segment described with a SEGTYPE of **U** cannot have any children.

KM

Indicates that this is a cross-referenced segment joined to the data source using a static join defined in the Master File and has a one-to-many relationship to the host segment. Joins defined in the Master File are described in Chapter 7, *Defining a Join in a Master File*. The parent-child pointer is stored in the data source.

KU

Indicates that this is a cross-referenced segment joined to the data source using a static join defined in the Master File and has a one-to-one relationship to the host segment (that is, it is a unique segment). Joins defined in the Master File are described in Chapter 7, *Defining a Join in a Master File*. The parent-child pointer is stored in the data source.

DKM

Indicates that this is a cross-referenced segment joined to the data source using a dynamic join defined in the Master File and has a one-to-many relationship to the host segment. Joins defined in the Master File are described in Chapter 7, *Defining a Join in a Master File*. The parent-child pointer is resolved at run-time and, therefore, new instances can be added without rebuilding.

DKU

Indicates that this is a cross-referenced segment joined to the data source using a dynamic join defined in the Master File and has a one-to-one relationship to the host segment (that is, it is a unique segment). Joins defined in the Master File are described in Chapter 7, *Defining a Join in a Master File*. The parent-child pointer is resolved at run-time and, therefore, new instances can be added without rebuilding.

KL

Indicates that this segment is described in a Master File-defined join as descending from a **KM**, **KU**, **DKM**, or **DKU** segment in a cross-referenced data source and has a one-to-many relationship to its parent.

KLU

Indicates that this segment is described in a Master File-defined join as descending from a **KM**, **KU**, **DKM**, or **DKU** segment in a cross-referenced data source and has a one-to-one relationship to its parent (that is, it is a unique segment).

Reference

Usage Notes for SEGTYPE

Note the following rules when using the SEGTYPE attribute with a FOCUS data source:

- **Alias.** SEGTYPE does not have an alias.
- **Changes.** You can change a SEGTYPE of S[n] or SH[n] to S0 or b or increase the number of key fields. To make any other change to SEGTYPE you need to use the REBUILD facility.

Describing a Key Field

You use the SEGTYPE attribute to describe which fields in a segment are key fields. The values of these fields determine how the segment instances are sequenced. The keys must be the first fields in a segment. You can specify up to 255 keys in a segment that is sorted from low to high (SEGTYPE = Sn), and up to 99 keys in a segment sorted from high to low (SEGTYPE = SHn). To maximize efficiency, it is recommended that you specify only as many keys as you need to make each record unique. You can also choose not to have any keys (SEGTYPE = S0 and SEGTYPE = b (blank)).

Note: Text fields cannot be used as key fields.

Describing Sort Order

For segments that have key fields, you use the SEGTYPE attribute to describe the segment's sort order. You can sort a segment's instances in two ways:

- **Low to high.** By specifying a SEGTYPE of Sn (where *n* is the number of keys), the instances are sorted using the concatenated values of the first *n* fields, beginning with the lowest value and continuing to the highest value.
- **High to low.** By specifying a SEGTYPE of SHn (where *n* is the number of keys), the instances are sorted using the concatenated values of the first *n* fields, beginning with the highest value and continuing to the lowest value.

Segments whose key is a date field often use a high-to-low sort order, since it ensures that the segment instances with the most recent dates will be the first ones encountered in a segment chain.

Understanding Sort Order

Suppose the following fields in a segment represent a department code and the employee's last name:

06345	19887	19887	23455	21334
Jones	Smith	Frank	Walsh	Brown

If you set SEGTYPE to S1, the department code becomes the key. (Note that two records have duplicate key values in order to illustrate a point about S2 segments later in this example; duplicate key values are not recommended for S1 and SH1 segments.) The segment instances will be sorted as follows:

06345	19887	19887	21334	23455
Jones	Smith	Frank	Brown	Walsh

If you change the field order to put the last name field before the department code and leave SEGTYPE as S1, the last name becomes the key. The segment instances will be sorted as follows:

Brown	Frank	Jones	Smith	Walsh
21334	19887	06345	19887	23455

Alternately, if you leave the department code as the first field, but set SEGTYPE to S2, the segment will be sorted first by the department code and then by last name, as follows:

06345	19887	19887	21334	23455
Jones	Frank	Smith	Brown	Walsh

Describing Segment Relationships

The SEGTYPE attribute describes the relationship of a segment to its parent segment:

- Physical one-to-one relationships are usually specified by setting SEGTYPE to U. If a segment is described in a Master File defined join as descending from the cross-referenced segment, then in the join description, SEGTYPE is set to KLU.
- Physical one-to-many relationships are specified by setting SEGTYPE to any valid value beginning with S (such as S0, SHn, and Sn), to blank, or, if a segment is described in a Master File defined join as descending from the cross-referenced segment, to KL.
- One-to-one joins defined in a Master File are specified by setting SEGTYPE to KU or DKU, as described in Chapter 7, *Defining a Join in a Master File*.
- One-to-many joins defined in a Master File are specified by setting SEGTYPE to KM or DKM, as described in Chapter 7, *Defining a Join in a Master File*.

Storing a Segment in a Different Location: LOCATION

By default, all of the segments in a FOCUS data source are stored in one physical file. For example, all of the EMPLOYEE data source's segments are stored in the data source named EMPLOYEE. If you wish, you can use the LOCATION attribute to specify that one or more segments be stored in a physical file separate from the main data source file. You can use a total of 64 LOCATION files per Master File (one LOCATION attribute per segment, except for the root). This can be helpful if you want to create a data source larger than the FOCUS limit for a single data source file, or if you want to store parts of the data source in separate locations for security or other reasons.

There are at least two cases in which you may want to use the LOCATION attribute:

- Each physical file is individually subject to a maximum file size of 64K 4K pages. You can use the LOCATION attribute to increase the size of your data source by splitting it into several physical files, each one subject to the maximum size limit. (You may also want to read Chapter 7, *Defining a Join in a Master File*, to see if it would be more efficient to structure your data as several joined data sources.)
- You can also store your data in separate physical files to take advantage of the fact that only the segments needed for a report must be present. Unreferenced segments stored in separate data sources can be kept on separate storage media to save space or implement separate security mechanisms. In some situations, the separation of the segments into different data sources allows different disk drives to be used.

Divided data sources do require more careful file maintenance. You have to be especially careful about procedures that are done separately to separate data sources, such as backups. If you do backups on Tuesday and Thursday for two related data sources, and you restore the FOCUS structure using the Tuesday backup for one half and the Thursday backup for the other, there is no way of detecting this discrepancy.

Syntax

How to Store a Segment in a Different Location

`LOCATION = filename`

where:

`filename`

Is the ddname of the file in which the segment is to be stored.

For example:

`SEGNAME = HISTREC, SEGTYPE = S1, PARENT = SSSREC, LOCATION = HISTFILE, $`

Example Specifying Location for a Segment

The following example illustrates the use of the LOCATION attribute:

```

FILENAME = PEOPLE, SUFFIX = FOC, $

SEGNAME = SSNREC,      SEGTYPE = S1, $
FIELD = SSN,          ALIAS = SOCSEG,  USAGE = I9, $

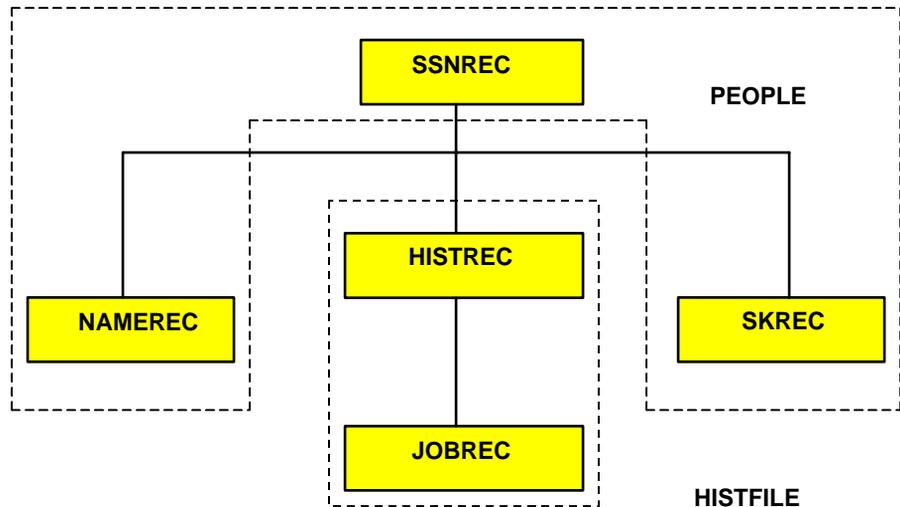
SEGNAME = NAMEREC,    SEGTYPE = U,      PARENT = SSNREC, $
FIELD = LNAME,       ALIAS = LN,      USAGE = A25, $

SEGNAME = HISTREC,   SEGTYPE = S1,    PARENT = SSNREC,
LOCATION = HISTFILE, $
FIELD = DATE,       ALIAS = DT,      USAGE = YMD, $

SEGNAME = JOBREC,    SEGTYPE = S1,    PARENT = HISTREC, $
FIELD = JOBCODE,    ALIAS = JC,      USAGE = A3, $

SEGNAME = SKREC,     SEGTYPE = S1,    PARENT = SSNREC, $
FIELD = SCODE,     ALIAS = SC,      USAGE = A3, $
    
```

This description groups the five segments into two physical files, as shown in the following diagram:



Note that the segment named SKREC, which contains no LOCATION attribute, is stored in the PEOPLE data source. This occurs because if no LOCATION attribute is specified for a segment, it is placed by default in the same file as its parent. In our example, you could assign the SKREC segment to a different file by specifying the LOCATION attribute in its declaration. However, it is strongly recommended that the LOCATION attribute be specified and not allowed to default.

Separating Large Text Fields

Text fields, by default, are stored in one physical file with non-text fields. However, as with segments, a text field can be located in its own physical file or any combination of text fields can share one or several physical files. You specify that you want a text field stored in a separate file by using the `LOCATION` attribute in the field definition.

For example, the text for `DESCRIPTION` will be stored in a separate physical file named `CRSEDESC`:

```
FIELD = DESCRIPTION, ALIAS = CDESC, USAGE = TX50, LOCATION = CRSEDESC , $
```

If you have more than one text field, each field can be stored in its own file, or several text fields can be stored in one file.

In this example, the text fields `DESCRIPTION` and `TOPICS` are stored in the `LOCATION` file `CRSEDESC`. The text field `PREREQUISITE` is stored in another file, `PREREQS`.

```
FIELD = DESCRIPTION , ALIAS = CDESC, USAGE = TX50, LOCATION = CRSEDESC, $
FIELD = PREREQUISITE, ALIAS = PREEQ, USAGE = TX50, LOCATION = PREREQS , $
FIELD = TOPICS, ALIAS = , USAGE = TX50, LOCATION = CRSEDESC, $
```

As with segments, you might want to use the `LOCATION` attribute on a text field if it is very long. However, unlike `LOCATION` segments, `LOCATION` files for text fields must be present during a request, whether or not the text field is referenced.

The `LOCATION` attribute can be used independently for segments and for text fields. That is, you can use the `LOCATION` attribute for a text field without using it for a segment. You can also use the `LOCATION` attribute for both the segment and the text field in the same Master File.

Note: Field names for text fields in FOCUS Master Files are limited to 12 characters; however, alias names for these fields can be up to 66 characters.

Limits on the Number of LOCATION Files

There is a limit on the number of different location segments and text location files you can specify. This limit is based on the number of entries allowed in the file directory table (FDT) for FOCUS data sources. The FDT contains the names of the segments in the data source, the names of indexed fields, and the names of location files for text fields. The FDT can contain 189 entries of which up to 64 can represent segments and location files. Each unique location file counts as one entry in the FDT.

For a given FOCUS data source, the maximum number of location files can be determined by the following formula

`Available FDT entries = 189 - (Segments + Indexes)`

`Location files = min (64, Available FDT entries)`

where:

Location files

Is the maximum number of location segments and text location files (up to a maximum of 64).

Segments

Is the number of segments in the Master File.

Indexes

Is the number of indexed fields.

For example, a ten-segment data source with 2 indexed fields would enable you to specify up to 52 location segments and/or location files for text fields (189 - (10 + 2)). Using the formula, the result would equal 177; however, the maximum number of text location files must always be *no more than 64*.

Note: If you specify a text field with a LOCATION attribute, the main file will be included in the text location file count.

The ACCEPT Attribute

ACCEPT is an optional attribute that you can use to validate data that is entered into a field using a MODIFY procedure. Its use with all types of data sources is described in Chapter 4, *Describing an Individual Field*. However, ACCEPT has a special option, FIND, that you can use only with FOCUS data sources. FIND enables you to verify incoming data against values stored in another field.

Syntax

How to Specify Data Validation

The syntax is

`ACCEPT = list`

`ACCEPT = range`

`ACCEPT = FIND (sourcefield [AS targetfield] IN file)`

where:

list

Is a list of acceptable values. This is described in Chapter 4, *Describing an Individual Field*.

range

Gives the range of acceptable values. This is described in Chapter 4, *Describing an Individual Field*.

FIND

Verifies the incoming data against the values in an index in a FOCUS data source.

sourcefield

Is the name of the field to which the ACCEPT attribute is being applied or any other field in the same segment or path to the segment. This must be the actual field name, not the alias or a truncation of the name.

AS targetfield

Is the name of the field that contains the acceptable data values. This field must be indexed.

IN file

Is the name of the Master File describing the data source that contains the indexed field of acceptable values.

The INDEX Attribute

You can index the values of a field by including the INDEX attribute, or its alias of FIELDTYPE, in the field's declaration. An index is an internally stored and maintained table of data values and locations that speeds retrieval. You need to create an index if you want to:

- Join two segments. The cross-referenced field in a joined FOCUS data source must be indexed, as described in *Describing a Single Segment* on page 6-5 (for joins defined in a Master File), and the *Creating Reports With WebFOCUS Language* manual (for joins defined using the JOIN command).
- Create an alternate view and make it faster, as described in Chapter 3, *Describing a Group of Fields*.
- Use a LOOKUP function in MODIFY.
- Use a FIND function in MODIFY.
- Speed segment selection and retrieval based on the values of a given field, as described for reporting in the *Creating Reports With WebFOCUS Language* manual.

Syntax

How to Specify Field Indexing

The syntax of the INDEX attribute in the Master File is:

```
INDEX = I or FIELDTYPE = I
```

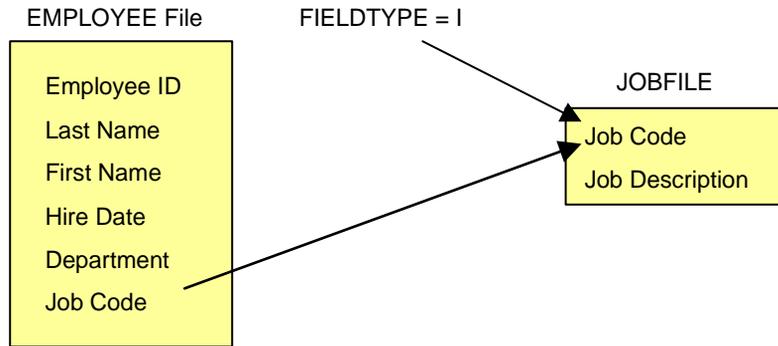
Text fields cannot be indexed. The maximum field name length for indexed fields is 12 characters.

For example:

```
FIELDNAME = JOBCODE, ALIAS = CJC, FORMAT = A3, INDEX = I, $
```

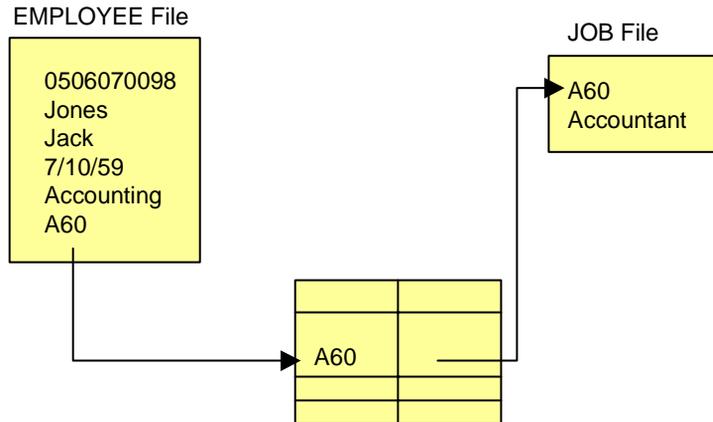
Joins and the INDEX Attribute

In order for a segment to be cross-referenced with a static cross-reference, a dynamic cross-reference, or a JOIN, at least one field in the cross-referenced segment must be indexed. This field, called the cross-referenced field, shares values with a field in the host data source. Only the cross-referenced segment needs to have an indexed field, shown as follows:



Other data sources locate and use segments through these indexes. Any number of fields may be indexed on a segment, although it is advisable to limit the number of fields you index in a data source.

The value for the field named JOBCODE in the EMPLOYEE data source is matched to the field named JOBCODE in the JOBFILe data source by using the index for the JOBCODE field in the JOBFILe data source, as follows:



Indexes are stored and maintained as part of the FOCUS data source. The presence of the index is crucial to the operation of the cross-referencing facilities. Any number of external sources may locate and thereby share a segment because of it. New data sources which have data items in common with indexed fields in existing data sources can be added at any time.

Reference

Usage Notes for INDEX

Note the following rules when using the INDEX attribute:

- **Alias.** INDEX has an alias of FIELDTYPE.
- **Changes.** If the INDEX attribute is removed from a field, or assigned a value of blank, which is equivalent, the index will no longer be maintained. If you no longer need the index, after you remove the INDEX attribute use the REORG option of the REBUILD facility to recover space occupied by the index. REBUILD is described in Chapter 10, *Creating and Rebuilding a Data Source*.

If you wish to turn off indexing temporarily—for example, to load a large amount of data into the data source quickly—you can remove the INDEX attribute before loading the data, restore the index attribute, and then use the REBUILD command with the INDEX option to create the index. This is known as post-indexing the data source.

You can index the field after the data source has already been created and populated with records by using the REBUILD facility with the INDEX option. A total of seven indexes may be added to the data source after the file is created using REBUILD INDEX. After seven indexes have been added to a data source in this way, you must use the REORG option of the REBUILD facility before adding an eighth. The following diagnostic message is issued if you attempt this:

```
(FOC720) THE NUMBER OF INDEXES ADDED AFTER FILE CREATION EXCEEDS 7
```

- **Maximum number.** The combined total of indexes, text fields, and segments cannot exceed 189 (of which a maximum of 64 can be segments and text location files).

FORMAT and MISSING: Internal Storage Requirements

Some application developers find it useful to know how different data types and values are represented and stored internally.

- Integer fields are stored as full-word (four byte) binary integers.
- Floating-point double-precision fields are stored as double-precision (eight byte) floating-point numbers.
- Floating-point single-precision fields are stored as single-precision (four byte) floating-point numbers.
- Packed-decimal fields are stored as 8 or 16 bytes and represent decimal numbers with up to 31 digits.
- Date fields are stored as full-word (four byte) binary integers representing the difference between the specified date and the date format's base date of December 31, 1900 (or JAN 1901, depending on the date format).
- Alphanumeric fields are stored as characters in the specified number of bytes.
- Missing values are represented internally by a dot (.) for alphanumeric fields, and as the value -9998998 for numeric fields.

CHAPTER 7

Defining a Join in a Master File

Topics:

- Join Types
- Static Joins Defined in the Master File: SEGTYPE = KU and KM
- Using Cross-Referenced Descendant Segments: SEGTYPE = KL and KLU
- Dynamic Joins Defined in the Master File: SEGTYPE =
- Comparing Static and Dynamic Joins
- Joining to One Cross-Referenced Segment From Several Host Segments

You can describe a new relationship between any two segments that have at least one field in common by joining them. The underlying data structures remain physically separate, but they are treated as if they were part of a single structure from which you can report. This chapter describes how to define a join in a Master File for FOCUS and fixed-format sequential data sources. For information about whether you can define a join in a Master File to be used with other types of data sources see the appropriate data adapter manual.

Join Types

You can join two data sources in the following ways:

- **Dynamically using the JOIN command.** The join lasts for the duration of the session (or until you clear it during the session) and creates a temporary view of the data that includes all of the segments in both data sources. You can also use the JOIN command to join two data sources of any type, including a FOCUS data source to a non-FOCUS data source. The JOIN command is described in detail in the *Creating Reports With WebFOCUS Language* manual.
- **Statically within a Master File.** This method is helpful if you want to access the joined structure frequently; the link (pointer) information needed to implement the join is permanently stored and does not need to be retrieved for each record during each request, saving you time. Like a dynamic Master File defined join, it is always available and retrieves only the segments that you specify. See *Static Joins Defined in the Master File: SEGTYPE = KU and KM* on page 7-3. This is supported for FOCUS data sources only.
- **Dynamically within a Master File.** This method saves you the trouble of issuing the JOIN command every time you need to join the data sources and gives you flexibility in choosing the segments that will be available within the joined structure. See *Dynamic Joins Defined in the Master File: SEGTYPE =* on page 7-14.

Development Tip:

Some users find it helpful to prototype a database design first using dynamic joins—implemented by issuing the JOIN command or within the Master File—and, once the design is stable, to change the frequently used joins to static joins defined in the Master File, accelerating data source access. Static joins should be used when the target or cross-referenced data source contents do not change. You can change dynamic joins to static joins by using the REBUILD facility.

Note: Master File defined joins are sometimes referred to as cross-references.

Static Joins Defined in the Master File: SEGTYPE = KU and KM

Static joins allow you to relate segments in different FOCUS data sources permanently. You specify static joins in the Master File of the host data source.

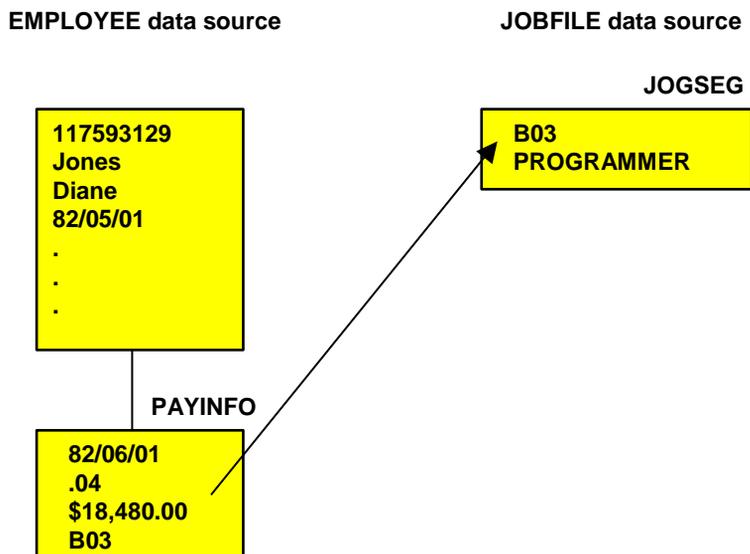
There are two types of static joins: one-to-one (SEGTYPE KU) and one-to-many (SEGTYPE KM).

- You specify a one-to-one join, also known as a unique join, when you want to retrieve at most one record instance from the cross-referenced data source for each record instance in the host data source.
- You specify a one-to-many join when you want to retrieve any number of record instances from the cross-referenced data source.

Describing a Unique Join: SEGTYPE = KU

In the EMPLOYEE data source, there is a field named JOBCODE in the PAYINFO segment. The JOBCODE field contains a code that specifies the employee's job.

The complete description of the job and other related information is stored in a separate data source named JOBFIL. You can retrieve the job description from JOBFIL by locating the record whose JOBCODE corresponds to the JOBCODE value in the EMPLOYEE data source, as shown in the following diagram:

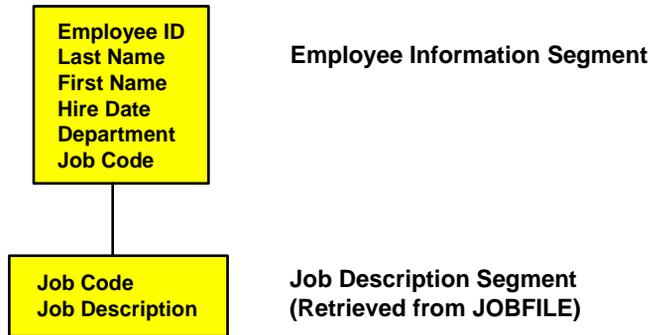


Using a join in this situation saves you the trouble of entering and revising the job description for every record in the EMPLOYEE data source. Instead, you can maintain a single list of valid job descriptions in the JOBFIL data source. Changes need be made only once, in JOBFIL, and are reflected in all of the corresponding joined EMPLOYEE data source records.

Implementing the join as a static join is most efficient because the relationship between job codes and job descriptions is not likely to change.

Although the Employee Information and Job Description segments are stored in separate data sources, for reporting purposes the EMPLOYEE data source is treated as though it also contains the Job Description segment from the JOBFIL data source. The actual structure of the JOBFIL data source is not affected. The EMPLOYEE data source is viewed as follows:

EMPLOYEE File



The Master File of the cross-referenced data source, as well as the data source itself, must be accessible whenever the host data source is used. There does not need to be any data in the cross-referenced data source.

Syntax

How to Specify a Static Unique Join

```
SEGNAME = segname, SEGTYPE = KU, PARENT = parent,  
CRFILE = db_name, CRKEY = field, [CRSEGNAME = crsegname,] $
```

where:

segname

Is the name by which the cross-referenced segment will be known in the host data source. You can assign any valid segment name, including the segment's original name in the cross-referenced data source.

parent

Is the name of the host segment.

db_name

Is the name of the cross-referenced data source. You can change the name without rebuilding the data source.

field

Is the name (field name or alias) of the field that the host file and cross-referenced file have in common. The field name or alias of the host field must be identical to the field name of the cross-referenced field. You can change the field name without rebuilding the data source as long as the SEGTYPE remains the same.

Both fields must have the same format type and length.

The cross-referenced field must be indexed (FIELDTYPE=I or INDEX=I).

crsegname

Is the name of the cross-referenced segment. If you do not specify this it defaults to the value assigned to SEGNAME. After data has been entered into the cross-referenced data source, you cannot change the crsegname without rebuilding the data source.

The SEGTYPE value KU stands for keyed unique.

Example

Creating a Static Unique Join

```
SEGNAME = JOBSEG, SEGTYPE = KU, PARENT = PAYINFO,  
CRFILE = JOBFILE, CRKEY = JOBCODE, $
```

The relevant sections of the EMPLOYEE Master File follow (for simplicity, fields and segments not essential to the example are not shown):

```
FILENAME = EMPLOYEE, SUFFIX = FOC, $  
  
SEGNAME = EMPINFO, SEGTYPE = S1, $  
.  
.  
.  
SEGNAME = PAYINFO, SEGTYPE = SH1, PARENT = EMPINFO, $  
FIELDNAME = JOBCODE, ALIAS = JBC, FORMAT = A3, $  
.  
.  
.  
SEGNAME = JOBSEG, SEGTYPE = KU, PARENT = PAYINFO, CRFILE = JOBFILE,  
CRKEY = JOBCODE, $
```

Note that you only have to give the name of the cross-referenced segment; the fields in that segment are already known from the cross-referenced data source's Master File (JOBFILE in this example). Note that the CRSEGNAME attribute is omitted, since in this example it is identical to the name assigned to the SEGNAME attribute.

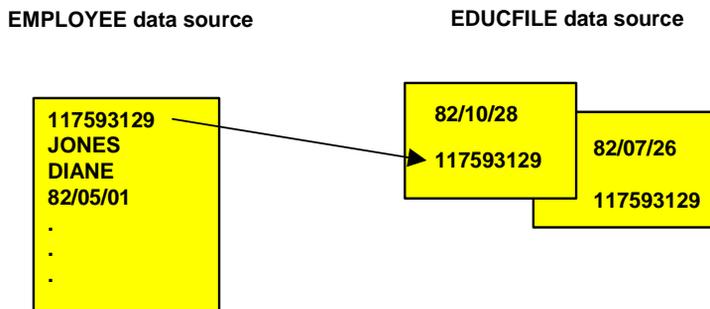
Using a Unique Join for Decoding

Decoding is the process of matching a code (such as the job code in our example) to the information it represents (such as the job description). Because every code has only one set of information associated with it, the join between the code and the information should be one-to-one, that is, unique. You can decode using a join, as in our example, or using the DECODE function with the DEFINE command, as described in the *Creating Reports With WebFOCUS Language* manual. The join method is recommended when there are a large number of codes.

Describing a Non-Unique Join: SEGTYPE = KM

You use a one-to-many join (that is, a non-unique join) when you have several instances of data in the cross-referenced segment associated with a single instance in the host segment. Using our EMPLOYEE example, suppose that you kept an educational data source named EDUCFILE to track the course work employees were doing. One segment in that data source, ATTNDSEG, contains the dates on which each employee attended a given class. The segment is keyed by attendance date. The EMP_ID field, which identifies the attendees, contains the same ID numbers as the EMP_ID field in the EMPINFO segment of the EMPLOYEE data source.

If you want to see an employee's educational record, you can join the EMP_ID field in the EMPINFO segment to the EMP_ID field in the ATTNDSEG segment. You should make this a one-to-many join, since you want to retrieve all instances of class attendance associated with a given employee ID:



Syntax

How to Specify a Static Multiple Join

The syntax for describing one-to-many joins is similar to that for one-to-one joins described in *How to Specify a Static Unique Join*, on page 7-5, except that you supply a different value, KM (which stands for keyed multiple), for the SEGTYPE attribute, as follows:

```
SEGTYPE = KM
```

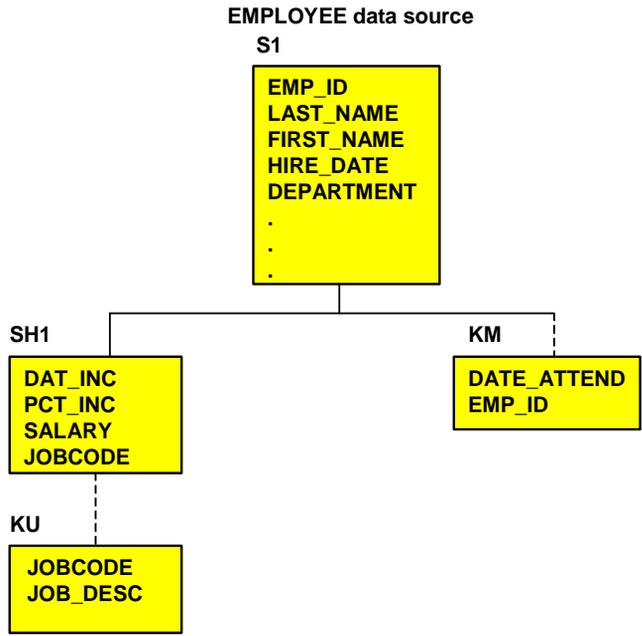
Example Specifying a Static Multiple Join

```
SEGNAME = ATTNDESEG, SEGTYPE = KM, PARENT = EMPINFO,  
CRFILE = EDUCFILE, CRKEY = EMP_ID, $
```

The relevant sections of the EMPLOYEE Master File follow (nonessential fields and segments are not shown):

```
FILENAME = EMPLOYEE, SUFFIX = FOC, $  
  
SEGNAME = EMPINFO, SEGTYPE = S1, $  
FIELDNAME = EMP_ID, ALIAS = EID, FORMAT = A9, $  
. . .  
SEGNAME = PAYINFO, SEGTYPE = SH1, PARENT = EMPINFO, $  
FIELDNAME = JOBCODE, ALIAS = JBC, FORMAT = A3, $  
. . .  
SEGNAME = JOBSEG, SEGTYPE = KU, PARENT = PAYINFO, CRFILE = JOBFILE,  
CRKEY = JOBCODE, $  
. . .  
SEGNAME = ATTNDESEG, SEGTYPE = KM, PARENT = EMPINFO, CRFILE = EDUCFILE,  
CRKEY = EMP_ID, $
```

Within a report request, both cross-referenced data sources, JOBFILE and EDUCFILE, are treated as though they are part of the EMPLOYEE data source. The data structure resembles the following:



Using Cross-Referenced Descendant Segments: SEGTYPE = KL and KLU

When you join two data sources, you can access any or all of the segments in the cross-referenced data source, not just the cross-referenced segment itself. These other segments are sometimes called linked segments. From the perspective of the host data source, all of the linked segments are descendants of the cross-referenced segment; it is as though an alternate view had been taken on the cross-referenced data source to make the cross-referenced segment the root. To access a linked segment, you only need to declare it in the Master File of the host data source.

Syntax

How to Identify Cross-Referenced Descendant Segments

```
SEGNAME = segname, SEGTYPE = {KL|KLU}, PARENT = parentname,  
CRFILE = db_name, [CRSEGNAME = crsegname,] $
```

where:

segname

Is the name assigned to the cross-referenced segment in the host data source.

KL

Indicates that this segment is a descendant segment in a cross-referenced data source (as viewed from the perspective of the host data source), and has a one-to-many relationship to its parent. KL stands for *keyed through linkage*.

KLU

Indicates that this segment is a descendant segment in a cross-referenced data source (as viewed from the perspective of the host data source), and has a one-to-one relationship to its parent. KLU stands for *keyed through linkage, unique*.

parentname

Is the name of the segment's parent in the cross-referenced data source, as viewed from the perspective of the host data source.

db_name

Is the name of the cross-referenced data source. You can change the name without rebuilding the data source.

crsegname

Is the name of the cross-referenced segment. If you do not specify this, it defaults to the value assigned to SEGNAME.

Example

Identifying a Cross-Referenced Descendant Segment

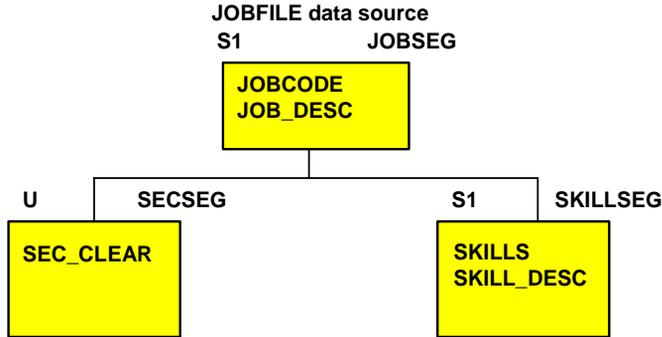
```
SEGNAME = SECSEG, SEGTYPE = KLU, PARENT = JOBSEG, CRFILE = JOBFILE, $  
SEGNAME = SKILLSEG, SEGTYPE = KL, PARENT = JOBSEG, CRFILE = JOBFILE, $
```

Note that you do not use the CRKEY attribute in a declaration for a linked segment, since the common join field (which is identified by CRKEY) needs to be specified only for the cross-referenced segment.

Example

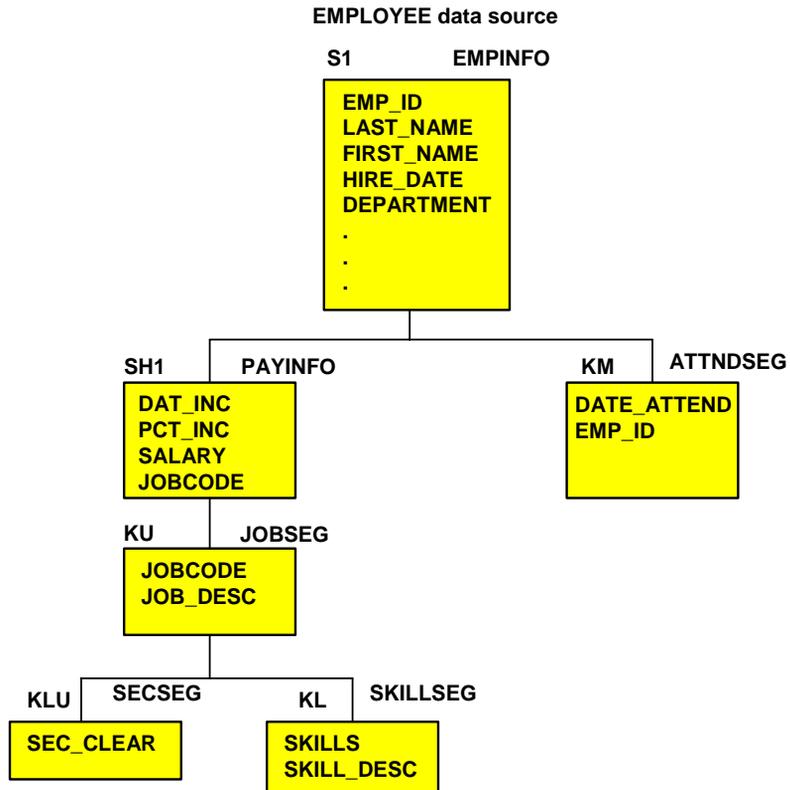
Using a Cross-Referenced Descendant Segment

Consider our EMPLOYEE example. JOBFIL is a multi-segment data source:



In your EMPLOYEE data source application, you may need the security information stored in the SECSEG segment and the job skill information stored in the SKILLSEG segment. Once you have created a join, you can access any or all of the other segments in the cross-referenced data source using the SEGTYPE value KL for a one-to-many relationship (as seen from the host data source), and KLU for a one-to-one relationship (as seen from the host data source). KL and KLU are used to access descendant segments in a cross-referenced data source for both static (KM) and dynamic (DKM) joins.

When the JOBSEG segment is retrieved from JOBFILE, it also retrieves all of JOBSEG's children that were declared with KL or KLU SEGTYPEs in the EMPLOYEE Master File:



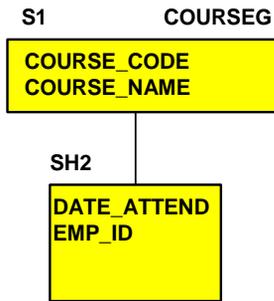
Example

Using a Cross-Referenced Ancestral Segment

Remember that you can retrieve all of the segments in a cross-referenced data source, including both descendants and ancestors of the cross-referenced segment. Ancestor segments should be declared in the host Master File with a SEGTYPE of KLU, as a segment can have only one parent and so, from the perspective of the host data source, this is a one-to-one relationship.

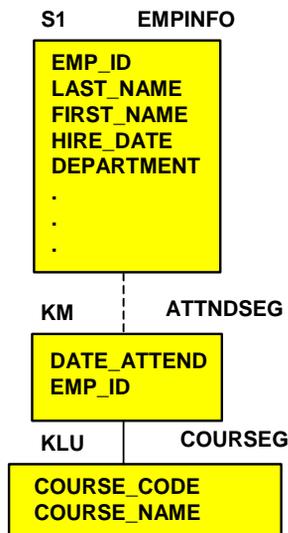
Consider the EDUCFILE data source used in our example. The COURSESEG segment is the root and describes each course; ATTNDSEG is a child and includes employee attendance information:

EDUCFILE data source



When you join EMPINFO in EMPLOYEE to ATTNDSEG in EDUCFILE, you can access course descriptions in COURSESEG by declaring it as a linked segment. From this perspective, COURSESEG is a child of ATTNDSEG:

EMPLOYEE data source

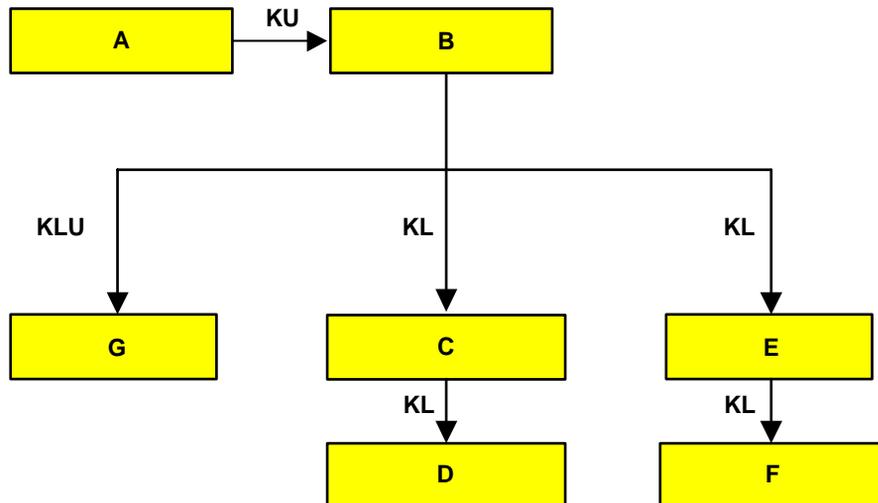


The sections of the EMPLOYEE Master File used in our examples follow (nonessential fields and segments are not shown):

```
FILENAME = EMPLOYEE, SUFFIX = FOC, $  
  
SEGNAME = EMPINFO, SEGTYPE = S1, $  
  FIELDNAME = EMP_ID, ALIAS = EID, FORMAT = A9, $  
  .  
  .  
  .  
SEGNAME = PAYINFO, SEGTYPE = SH1, PARENT = EMPINFO, $  
  FIELDNAME = JOBCODE, ALIAS = JBC, FORMAT = A3, $  
  .  
  .  
  .  
SEGNAME = JOBSEG,   SEGTYPE = KU,   PARENT = PAYINFO,  CRFILE = JOBFILE,  
  CRKEY = JOBCODE, $  
SEGNAME = SECSEG,   SEGTYPE = KLU,  PARENT = JOBSEG,   CRFILE = JOBFILE, $  
SEGNAME = SKILLSEG, SEGTYPE = KL,   PARENT = JOBSEG,   CRFILE = JOBFILE, $  
SEGNAME = ATTNDSEG, SEGTYPE = KM,   PARENT = EMPINFO,  CRFILE = EDUCFILE,  
  CRKEY = EMP_ID, $  
SEGNAME = COURSEG,  SEGTYPE = KLU,  PARENT = ATTNDSEG, CRFILE = EDUCFILE, $
```

Hierarchy of Linked Segments

A KL segment may lead to other KL segments. Graphically, this can be illustrated as:



The letters on the arrows are the SEGTYPEs.

Note that segment G may either be a unique descendant of B or B's parent.

Dynamic Joins Defined in the Master File: SEGTYPE = DKU and DKM

You can define a dynamic join in a Master File using the SEGTYPE attribute. There are two types of dynamic Master File defined joins: one-to-one (SEGTYPE DKU) and one-to-many (SEGTYPE DKM).

- As with a static join, you specify a one-to-one join, also known as a unique join, when you want to retrieve at most one record instance from the cross-referenced data source for each record instance in the host data source.
- You specify a one-to-many join when you want to retrieve any number of record instances from the cross-referenced data source.

The difference between static and dynamic joins has to do with storage, speed, and flexibility:

- The links (pointers) for a static join are retrieved once and then permanently stored in the host data source (and automatically updated as needed).
- The links for a dynamic join are not saved and need to be retrieved for each record in each report request.

This makes static joins much faster than dynamic ones, but harder to change. You can redefine or remove a static join only using the REBUILD facility. You can redefine or remove a dynamic join at any time by editing the Master File.

Syntax

How to Specify a Dynamic Join in a Master File

You specify a dynamic Master File defined join the same way that you specify a static join (as described in *How to Specify a Static Unique Join*, on page 7-5), except that the value of the SEGTYPE attribute for the cross-referenced segment is DKU (standing for dynamic keyed unique) for a one-to-one join, and DKM (standing for dynamic keyed multiple) for a one-to-many join.

For example:

```
SEGNAME = JOBSEG, SEGTYPE = DKU, PARENT = PAYINFO,  
CRFILE = JOBFIL, CRKEY = JOBCODE, $
```

You declare linked segments in a dynamic join the same way that you do in a static join. In both cases, SEGTYPE has a value of KLU for unique linked segments, and KL for non-unique linked segments.

Example**Specifying a Dynamic Join in a Master File**

The following Master File includes the relevant sections of EMPLOYEE and the segments joined to it, but with the static joins replaced by dynamic joins (nonessential fields and segments are not shown):

```
FILENAME = EMPLOYEE, SUFFIX = FOC, $

SEGNAME = EMPINFO, SEGTYPE = S1, $
    FIELDNAME = EMP_ID, ALIAS = EID, FORMAT = A9, $
.
.
.
SEGNAME = PAYINFO, SEGTYPE = SH1, PARENT = EMPINFO, $
    FIELDNAME = JOBCODE, ALIAS = JBC, FORMAT = A3, $
.
.
.
SEGNAME = JOBSEG, SEGTYPE = DKU, PARENT = PAYINFO, CRFILE = JOBFILE,
    CRKEY = JOBCODE, $
SEGNAME = SECSEG, SEGTYPE = KLU, PARENT = JOBSEG, CRFILE = JOBFILE, $
SEGNAME = SKILLSEG, SEGTYPE = KL, PARENT = JOBSEG, CRFILE = JOBFILE, $
SEGNAME = ATTNDSEG, SEGTYPE = DKM, PARENT = EMPINFO, CRFILE = EDUCFILE,
    CRKEY = EMP_ID, $
SEGNAME = COURSEG, SEGTYPE = KLU, PARENT = ATTNDSEG, CRFILE = EDUCFILE, $
```

Comparing Static and Dynamic Joins

If you wish to join two FOCUS data sources, you can choose between two types of joins (static and dynamic) and two methods of defining the join (defined in the Master File and defined by issuing the JOIN command).

- For a static join, the links, which point from a host segment instance to the corresponding cross-referenced segment instance, are created once and then permanently stored and automatically maintained in the host data source.
- For a dynamic join, the links are retrieved each time they are needed. This makes static joins faster than dynamic ones, since the links only need to be established once, but less flexible, as you can redefine or remove a static join only by using the REBUILD facility.

Among dynamic joins, the JOIN command is easier to use in that you do not need to edit the Master File each time you want to change the join specification, and you do not need to describe each linked segment as it appears from the perspective of the host data source. On the other hand, Master File defined dynamic joins enable you to omit unnecessary cross-referenced segments.

You may find it efficient to implement frequently used joins as static joins. You can change static joins to dynamic, and dynamic to static, using the REBUILD facility.

The following chart compares implementing a static join defined in a Master File, a dynamic join defined in a Master File, and a dynamic join defined by issuing the JOIN command.

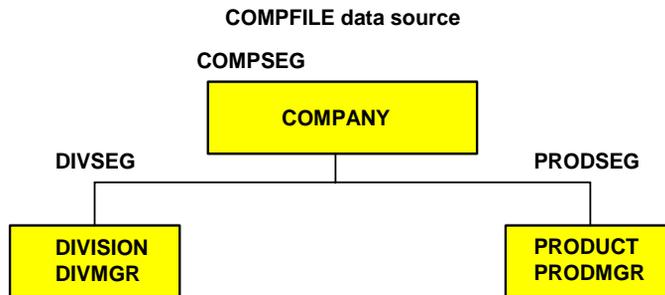
	Advantages	Disadvantages
Static Join in Master File (SEGTYPE = KU or KM)	<p>Faster after first use: links are created only once.</p> <p>Always in effect.</p> <p>Can select some linked segments and omit others.</p>	<p>Must be specified before data source is created or reloaded using REBUILD.</p> <p>Requires REBUILD utility to change.</p> <p>Requires four bytes of file space per instance.</p> <p>User needs to know how to specify relationships for linked segments (KL, KLU).</p>
Dynamic Join in Master File (SEGTYPE = DKU or DKM)	<p>Can be specified at any time.</p> <p>Always in effect. Does not use any space in the data source.</p> <p>Can be changed or removed as needed, without using the REBUILD facility.</p> <p>Can select some linked segments and omit others.</p>	<p>Slower: links are retrieved for each record in each report request.</p> <p>User needs to know how to specify relationships for linked segments (KL, KLU).</p>
Dynamic Join (using the JOIN Command)	<p>Can be specified at any time.</p> <p>Does not use any space in the data source. Can be changed or removed as needed, without using the REBUILD facility.</p> <p>User never needs to describe relationships of linked segments.</p>	<p>Slower: links are retrieved for each record in each report request.</p> <p>JOIN command must be issued in each session in which you want the join to be in effect.</p> <p>All linked segments are always included, whether or not you need them.</p>

Joining to One Cross-Referenced Segment From Several Host Segments

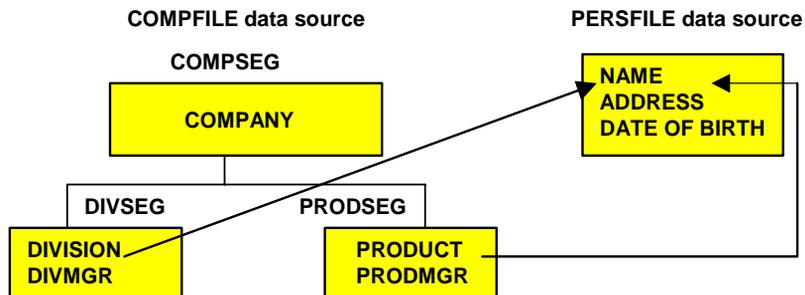
You may come upon situations where you need to join to one cross-referenced segment from several different segments in the host data source. You may also find a need to join to one cross-referenced segment from two different host data sources at the same time. You can handle these data structures using Master File defined joins.

Joining From Several Segments in One Host Data Source

In an application, you may want to use the same cross-referenced segment in several places in the same data source. Suppose, for example, that you have a data source named COMPFILE that maintains data on companies you own:



The **DIVSEG** segment contains an instance for each division and includes fields for the name of the division and its manager. Similarly, the **PRODSEG** segment contains an instance for each product and the name of the product manager. You might want to retrieve personal information for both the product managers and the division managers from a single personnel data source, as shown below:



You cannot retrieve this information with a standard Master File defined join because there are two cross-reference keys in the host data source (**PRODMGR** and **DIVMGR**) and in your reports you will want to distinguish addresses and dates of birth retrieved for the **PRODSEG** segment from those retrieved for the **DIVSEG** segment.

A way is provided for you to implement a join to the same cross-referenced segment from several segments in the one host data source; you can match the cross-referenced and host fields from alias to field name and uniquely rename the fields.

The Master File of the PERSFILE might look like this:

```
FILENAME = PERSFILE, SUFFIX = FOC, $
SEGNAME = IDSEG, SEGTYPE = S1, $
    FIELD = NAME, ALIAS = FNAME, FORMAT = A12, INDEX=I, $
    FIELD = ADDRESS, ALIAS = DAS, FORMAT = A24, $
    FIELD = DOB, ALIAS = IDOB, FORMAT = YMD, $
```

You use the following Master File to join PERSFILE to COMPFIL. Note that there is no record terminator (\$) following the cross-referenced segment declaration (preceding the cross-referenced field declarations).

```
FILENAME = COMPFIL, SUFFIX = FOC, $
SEGNAME = COMPSEG, SEGTYPE = S1, $
    FIELD = COMPANY, ALIAS = CPY, FORMAT = A40, $
SEGNAME = DIVSEG, PARENT = COMPSEG, SEGTYPE = S1, $
    FIELD = DIVISION, ALIAS = DV, FORMAT = A20, $
    FIELD = DIVMGR, ALIAS = NAME, FORMAT = A12, $
SEGNAME = ADSEG, PARENT = DIVSEG, SEGTYPE = KU,
    CRSEGNAME = IDSEG, CRKEY = DIVMGR, CRFILE = PERSFILE,
    FIELD = NAME, ALIAS = FNAME, FORMAT = A12, INDEX = I, $
    FIELD = DADDRESS, ALIAS = ADDRESS, FORMAT = A24, $
    FIELD = DDOB, ALIAS = DOB, FORMAT = YMD, $
SEGNAME = PRODSEG, PARENT = COMPSEG, SEGTYPE = S1, $
    FIELD = PRODUCT, ALIAS = PDT, FORMAT = A8, $
    FIELD = PRODMGR, ALIAS = NAME, FORMAT = A12, $
SEGNAME = BDSEG, PARENT = PRODSEG, SEGTYPE = KU,
    CRSEGNAME = IDSEG, CRKEY = PRODMGR, CRFILE = PERSFILE,
    FIELD = NAME, ALIAS = FNAME, FORMAT = A12, INDEX = I, $
    FIELD = PADDRESS, ALIAS = ADDRESS, FORMAT = A24, $
    FIELD = PDOB, ALIAS = DOB, FORMAT = YMD, $
```

DIVMGR and PRODMGR are described as CRKEYs. Their common alias, NAME, is automatically matched to the field name NAME in the PERSFILE data source. In addition, the field declarations that follow the join information rename the ADDRESS and DOB fields so that they can be referred to separately in reports. Their actual field names in the PERSFILE are supplied as aliases.

Note that the NAME field cannot be renamed, since it is the common join field. It must be included in the declaration along with the fields being renamed, as it is described in the cross-referenced data source. That it cannot be renamed is not a problem, since its ALIAS can be renamed, and, in any event, the field does not need to be used in reports; because it is the join field, it contains exactly the same information as the DIVMGR and PRODMGR fields.

The following conventions must be observed:

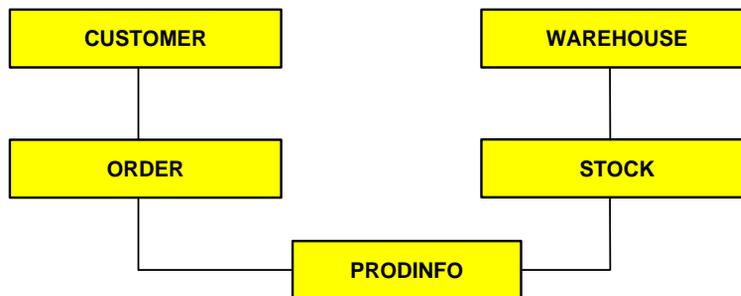
- The common join field's FIELDNAME or ALIAS in the host data source must be identical to its FIELDNAME in the cross-referenced data source.
- The common join field should not be renamed, but the alias can be changed. The other fields in the cross-referenced segment can be renamed.
- Place field declarations for the cross-referenced segment after the cross-referencing information in the Master File of the host data source, in the order in which they actually occur in the cross-referenced segment. Omit the record terminator (\$) at the end of the cross-referenced segment declaration in the host Master File, as shown:

```
SEGNAME = BDSEG, PARENT = PRODSEG, SEGTYPE = KU,  
CRSEGNAME = IDSEG, CRKEY = PRODMGR, CRFILE = PERSFILE,  
FIELD = NAME, ALIAS = FNAME, FORMAT = A12, INDEX=I, $  
FIELD = PADDRESS, ALIAS = ADDRESS, FORMAT = A24, $  
FIELD = PDOB, ALIAS = DOB, FORMAT = YMD, $
```

Joining From Several Segments in Several Host Data Sources: Multiple Parents

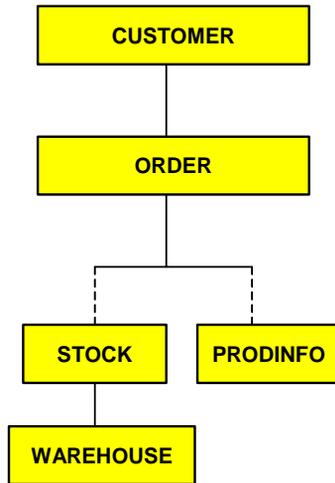
At some point you may need to join to a cross-referenced segment from two different host data sources at the same time. If you were to describe a structure like this as a single data source, you would have to have two parents for the same segment, which is invalid. You can, however, describe the information in separate data sources, using joins to achieve a similar effect.

Consider an application that keeps track of customer orders for parts, warehouse inventory of parts, and general part information. If this were described as a single data source, it would be structured as follows:

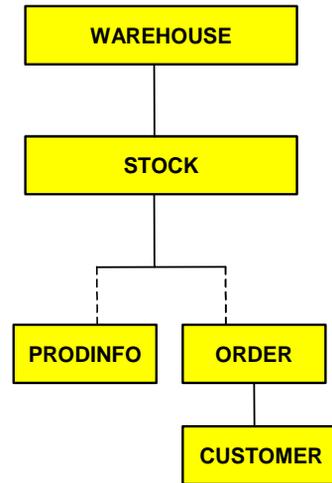


You can join several data sources to create this structure. For example:

view from ORDERS data source:



view from INVENTORY data source:



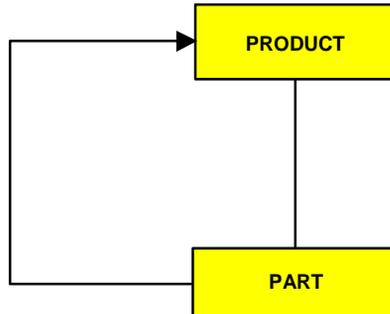
The CUSTOMER and ORDER segments are in the ORDERS data source, the WAREHOUSE and STOCK segments are in the INVENTORY data source, and the PRODINFO segment is stored in the PRODUCTS data source. Both the INVENTORY and ORDERS data sources have one-to-one joins to the PRODUCTS data source. In the INVENTORY data source, STOCK is the host segment; in the ORDERS data source, ORDER is the host segment.

In addition, there is a one-to-many join from the STOCK segment in the INVENTORY data source to the ORDER segment in the ORDERS data source, and a reciprocal one-to-many join from the ORDER segment in the ORDERS data source to the STOCK segment in the INVENTORY data source.

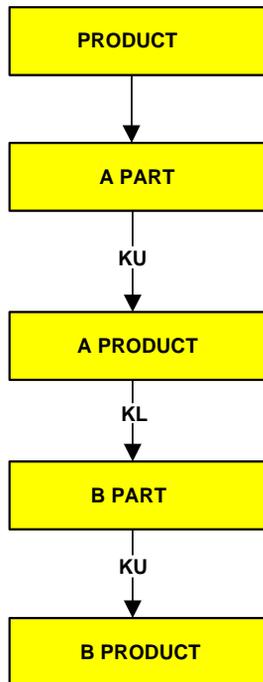
The joins among these three data sources can be viewed from the perspectives of both host data sources, approximating the multiple-parent structure described earlier.

Recursive Reuse of a Segment

In rare cases, a data source may cross-reference itself. Consider the case of a data source of products, each with a list of parts that compose the product, where a part may itself be a product and have sub-parts. Schematically, this would appear as:



A description for this case, shown for two levels of sub-parts, is:



See the *Creating Reports With WebFOCUS Language* manual for more information on recursive joins.

CHAPTER 8

Checking and Changing a Master File: CHECK

Topics:

- CHECK Command Display
- The PICTURE Option
- The HOLD Option

Use the CHECK command to validate your Master Files. You must always do this after writing the Master File. If you do not issue the CHECK command, your Master File may not be updated with the changes that you just made. The CHECK output highlights any errors in your Master File and allows you to correct them before reading the data source. After making any necessary corrections, use CHECK again to confirm that the Master File is valid.

Syntax

How to Check a Data Source Description

```
CHECK FILE filename[.field] [PICTURE [RETRIEVE]] [DUPLICATE]  
[HOLD [AS name] [ALL]]
```

where:

filename

Is the name under which you created the Master File.

.field

Is used for an alternate view of the Master File.

PICTURE

Is an option that displays a diagram showing the complete data source structure. The keyword PICTURE can be abbreviated to PICT. This option is explained in *The PICTURE Option* on page 8-5.

RETRIEVE

Alters the picture to reflect the order in which segments are retrieved when TABLE or TABLEF commands are issued. Note that unique segments are viewed as logical extensions of their parent segment. The keyword RETRIEVE can be abbreviated to RETR.

DUPLICATE

Lists duplicate field names for the specified data source. The keyword DUPLICATE can be abbreviated to DUPL.

HOLD

Generates a temporary HOLD file and HOLD Master File containing information about fields in the data source. You can use this HOLD file to write reports. The AS option specifies a field name for your data sources. The option is described and illustrated in *The HOLD Option* on page 8-8.

name

Is a name for the HOLD file and HOLD Master File.

ALL

Adds the values of FDFCENT and FYRTHRESH at the file level and the values of DFECENT and YRTHRESH at the field level to the HOLD file.

CHECK Command Display

If your Master File contains syntactical errors, the CHECK command displays appropriate messages.

If the data source description has no syntactical errors, the CHECK command displays the following message:

```
NUMBER OF ERRORS=          0
NUMBER OF SEGMENTS=       n ( REAL=      n VIRTUAL= n )
NUMBER OF FIELDS=        n INDEXES=    n FILES=   n
NUMBER OF DEFINES=       n
TOTAL LENGTH OF ALL FIELDS = n
```

where:

NUMBER OF ERRORS

Indicates the number of syntactical errors in the Master File.

NUMBER OF SEGMENTS

Is the number of segments in the Master File, including cross-referenced segments.

REAL

Is the number of segments that are not cross-referenced. These segments have types Sn, SHn, U, or blank.

VIRTUAL

Is the number of segments that are cross-referenced. These segments have types KU, KLU, KM, KL, DKU, or DKM.

NUMBER OF FIELDS

Is the number of fields described in the Master File.

INDEXES

Is the number of indexed fields. These fields have the attribute FIELDTYPE=I or INDEX=I in the Master File.

FILES

Is the number of data sources containing the fields.

NUMBER OF DEFINES

Is the number of virtual fields in the Master File. This message displays only if virtual fields are defined.

TOTAL LENGTH

Is the total length of all fields as defined in the Master File by either the FORMAT attribute (if the data source is a FOCUS data source) or the ACTUAL attribute (if the data source is a non-FOCUS data source).

Example

Using the CHECK File Command

Entering the following command

```
CHECK FILE EMPLOYEE
```

produces the following information:

```
NUMBER OF ERRORS=          0
NUMBER OF SEGMENTS=        11      ( REAL=   6 VIRTUAL=   5 )
NUMBER OF FIELDS=          34      INDEXES=   0 FILES=    3
TOTAL LENGTH OF ALL FIELDS = 365
```

Determining Common Errors

- If the data source is a non-FOCUS data source, check the TOTAL LENGTH OF ALL FIELDS that displays near the top of your screen to verify the accuracy of the field lengths you have specified for the data source. One of the most common causes of errors in generating reports from non-FOCUS data sources is incorrectly specified field lengths. The number given as the total length of all fields should be equal to the logical record length of the non-FOCUS data source.

In general, if the total length of all fields is not equal to the logical record length of the non-FOCUS data source, you have specified the length of at least one field incorrectly. Your external data may not be read correctly if you do not correct the error.

- If the following warning message is generated

```
(FOC1829) WARNING. FIELDNAME IS NOT UNIQUE WITHIN A SEGMENT: fieldname
```

it is because duplicate fields (those having the same field names and aliases) are not allowed in the same segment. The second occurrence is never accessed.

When the CHECK command is issued for a data source that has more than one field of the same name within the same segment, a FOC1829 message is generated along with a warning indicating the duplicate field names, such as the following:

```
(FOC1829) WARNING. FIELDNAME IS NOT UNIQUE WITHIN A SEGMENT: BB
WARNING: FOLLOWING FIELDS CANNOT BE ACCESSED
BB  IN SEGMENT SEGA      (VS SEGB )
```

When the DUPLICATE option is added, the output contains a warning message like the following:

```
WARNING: FOLLOWING FIELDS APPEAR MORE THAN ONCE
AA  IN SEGMENT SEGB      (VS SEGA )
```

The PICTURE Option

The PICTURE option displays a diagram of the structure defined by the Master File. Each segment is represented by a box. There are four types of boxes, which indicate whether a segment (including the root segment) is non-unique or unique and whether it is real or cross-referenced. The four types of boxes are:

Real segments

Non-unique segment:

```

                segname
num          segtype
*****
*field1      **I
*field2      **
*field3      **
*field4      **
*            **
*****
*****

```

Unique segment:

```

                segname
num            U
*****
*field1       *I
*field2       *
*field3       *
*field4       *
*            *
*****

```

Cross-referenced segments

Non-unique segment:

```

                segname
num          KM (or KLM)
.....
:field1      ::K
:field2      ::
:field3      ::
:field4      ::
:            ::
:.....:
:.....:
                crfile

```

Unique segment

```

                segname
num          KU (or KLU)
.....
:field1      :K
:field2      :
:field3      :
:field4      :
:            :
:.....:
                crfile

```

where:

num

Is the number assigned to the segment in the structure.

segname

Is the name of the segment.

segtype

Is the segment type for a real, non-unique segment: Sn, SHn, or N (for blank segtypes).

field1 ...

Are the names of fields in the segment. Field names of 66 characters are truncated to 12 characters in CHECK FILE PICTURE operations.

I

Indicates an indexed field.

K

Indicates the key field in the cross-referenced segment.

crfile

Is the name of the cross-referenced data source if the segment is cross-referenced.

The diagram also shows the relationship between segments (see the following example). Parent segments are shown above children segments connected by straight lines.

Example**Using the CHECK FILE PICTURE Option**

The following diagram shows the structure of the JOB data source joined to the SALARY data source:

```

JOIN EMP_ID IN JOB TO EMP_ID IN SALARY
>
CHECK FILE JOB PICTURE
  NUMBER OF ERRORS=      0
  NUMBER OF SEGMENTS=   2 ( REAL=   1 VIRTUAL=   1 )
  NUMBER OF FIELDS=    7 INDEXES=   0 FILES=    2
  TOTAL LENGTH OF ALL FIELDS=  86
SECTION 01
                STRUCTURE OF FOCUS      FILE JOB      ON 02/08/00 AT 12.33.04

                JOBSEG
01             S1
*****
*EMP_ID       **
*FIRST_NAME   **
*LAST_NAME    **
*JOB_TITLE    **
*             **
*****
                I
                I
                I
                I SALSEG
02             I KU
.....
:EMP_ID       :K
:SALARY       :
:EXEMPTIONS  :
:             :
:             :
:.....:
JOINED SALARY

```

The HOLD Option

The HOLD option generates a temporary HOLD file. HOLD files are explained in the *Creating Reports With WebFOCUS Language* manual. This HOLD file contains detailed information regarding file, segment, and field attributes, which you can display in reports using TABLE requests.

Certain fields in this HOLD file are of special interest. Unless otherwise noted, these fields are named the same as attributes in Master Files; each field stores the values of the similarly-named attribute. The fields can be grouped into file attributes, segment attributes, and field attributes.

File Attributes:

FILENAME
SUFFIX
DFCENT, FYRTHRESH

Note that these attributes are included in the HOLD file, if they exist in the original Master File and you specify the ALL option.

Segment Attributes:

SEGNAME
SEGTYPE

Note that this field does not indicate the number of segment key fields. Segment types S1, S2, and so on are shown as type S. The same is true with segment type SHn.

SKEYS

The number of segment key fields. For example, if the segment type is S2, SKEYS has the value 2.

SEGN0

The number assigned to the segment within the structure. This is displayed in the picture.

LEVEL

The level of the segment within the structure. The root segment is on Level 1, its children are on Level 2, and so on.

PARENT
CRKEY
FIELDNAME

Field Attributes:

ALIAS
FORMAT
ACTUAL

Note that if you include the FORMAT field in the TABLE request, you should not use the full field name FORMAT. Rather, you should use the alias USAGE or a unique truncation of the FORMAT field name (the shortest unique truncation is FO).

DFCENT, YRTHRESH

Note that these attributes are included in the HOLD file, if they exist in the original Master File and you specify the ALL option.

Example**Using the CHECK FILE HOLD Option**

This sample procedure creates a HOLD file describing the EMPLOYEE data source. It then writes a report that displays the names of cross-referenced segments in the EMPLOYEE data source, their segment types, and the attributes of their fields: field names, aliases, and formats.

```
CHECK FILE EMPLOYEE HOLD
TABLE FILE HOLD
HEADING
"FIELDNAMES, ALIASES, AND FORMATS"
"OF CROSS-REFERENCED FIELDS IN THE EMPLOYEE DATA SOURCE"
" "
PRINT FIELDNAME ALIAS USAGE BY SEGNAME BY SEGTYPE
WHERE SEGTYPE CONTAINS 'K'
END
```

The output is:

```
PAGE 1

FIELDNAMES, ALIASES, AND FORMATS
OF CROSS-REFERENCED FIELDS IN THE EMPLOYEE DATA SOURCE

SEGNAME      SEGTYPE      FIELDNAME      ALIAS      FORMAT
-----      -
ATTNDSEG     KM           DATE_ATTEND    DA         I6YMD
              EMP_ID       EID           A9
COURSEG      KLU          COURSE_CODE    CC         A6
              COURSE_NAME  CD            A30
JOBSEG       KU           JOBCODE        JC         A3
              JOB_DESC     JD            A25
SECSEG       KLU          SEC_CLEAR      SC         A6
SKILLSEG     KL           SKILLS         A4
              SKILL_DESC   SD            A30
```

Example**Using the CHECK FILE HOLD ALL Option**

Assume the Employee data source contains the following FILE declaration:

```
FILENAME = EMPLOYEE, SUFFIX = FOC, FDEFCENT = 19, FYRTHRESH = 50
```

The following request:

```
CHECK FILE EMPLOYEE HOLD ALL
TABLE FILE HOLD
PRINT FDEFCENT FYRTHRESH
END
```

produces the following output:

```
FDEFCENT      FYRTHRESH
-----      -
              19          50
```

Specifying an Alternate File Name With the HOLD Option

An AS name may be provided for the temporary HOLD file generated by the CHECK command. If a name is not specified, the default name is HOLD and any existing default file will be replaced.

Note:

- The AS name may not be longer than 8 characters, or it defaults to the name HOLD and no warning is issued.
- When the AS option is specified in combination with other CHECK options, the AS holdname specification must appear last.

TITLE, HELPMESSAGE, and TAG Attributes

When you use the HOLD option of the CHECK command, the TITLE text is placed in the TITLE field of the FLDATTR segment, the HELPMESSAGE text in the HELPMESSAGE field of the FLDATTR segment, and the TAG names in the TAGNAME field of the SEGATTR segment.

When no JOINS are in effect, or when a JOIN command is issued without a TAG name, the TAGNAME field by default contains the name of the data source specified in the CHECK command. When JOINS are issued in conjunction with the TAG name feature, the TAGNAME field contains the TAG name for the host and cross-referenced data sources.

Note: The addition of these three fields caused an increase in the LRECL of the HOLD FOCTEMP file to 370 when SET FIELDNAME=NEW and to 238 when SET FIELDNAME=OLD.

Virtual Fields in the Master File

With the HOLD option, virtual fields are placed in the segment in which they would be stored if they were real fields in the data source. This is not necessarily the physical location of the field in the Master File, but the lowest segment that must be accessed in order to evaluate the expression defining the field. Fields whose values are not dependent on retrieval default to the top segment. The value of FLDSEG in the FLDATTR segment is zero for these fields. The format of FLDSEG is I2S in the Master File, which causes zero to be displayed as blank in reports. FLDSEG may be dynamically reformatted in a TABLE request (FLDSEG/I2) to force the display of zero.

Once data has been entered into a data source, you can no longer make arbitrary changes to the Master File. Some changes are entirely harmless and can be made at any time; others are prohibited unless the data is reentered or the data source rebuilt. A few others can be made if corresponding changes are made in several places.

You can use a text editor to make permitted changes to the Master File. The checking procedure, CHECK, should be used after any change.

CHAPTER 9

Providing Data Source Security: DBA

Topics:

- Introduction to Data Source Security
- Implementing Data Source Security
- Specifying an Access Type: The ACCESS Attribute
- Limiting Data Source Access: The RESTRICT Attribute
- Placing Security Information in a Central Master File
- Hiding Restriction Rules: The ENCRYPT Command
- WebFOCUS FOCEXEC Security

As Database Administrator, you can use WebFOCUS DBA security features to provide security for any WebFOCUS data source. You can use these security features to limit the number of records or reads a user can request in a report.

You can also use DBA security features to provide security for non-WebFOCUS data sources. However, the RESTRICT command (*Restricting an Existing FOCUS Data Source* on page 9-23) is not available for those data sources. Note that DBA security cannot protect a data source from non-WebFOCUS access.

Introduction to Data Source Security

The DBA facility provides a number of security options:

- You can limit the users who have access to a given data source using the **USER** attribute discussed in *Identifying Users With Access Rights: The USER Attribute* on page 9-6.
- You can restrict a user's access rights to read, write, or update only using the **ACCESS** attribute discussed in *Specifying an Access Type: The ACCESS Attribute* on page 9-9.
- You can restrict a user's access to certain fields or segments using the **RESTRICT** attribute discussed in *Limiting Data Source Access: The RESTRICT Attribute* on page 9-12.
- You can ensure that only records that pass a validation test are retrieved using the **RESTRICT** attribute discussed in *Limiting Data Source Access: The RESTRICT Attribute* on page 9-12.
- You can limit the values a user can write to the data source or you can limit which values a user can alter using the **RESTRICT** attribute discussed in *Limiting Data Source Access: The RESTRICT Attribute* on page 9-12.
- You can point to passwords and restrictions stored in another Master File with the **DBAFILE** attribute discussed in *Placing Security Information in a Central Master File* on page 9-17.
- You can use the WebFOCUS DBA exit routine to let an external security system set the WebFOCUS password. For more information, see the *WebFOCUS Security and Administration* manual.
- You can place security on FOCEXECs, which is discussed in *WebFOCUS FOCEXEC Security* on page 9-24.

Implementing Data Source Security

You provide WebFOCUS security on a file-by-file basis. Implementing DBA security features is a straightforward process in which you specify:

- The names or passwords of WebFOCUS users granted access to a data source.
- The type of access the user is granted.
- The segments, fields, or ranges of data values to which the user's access is restricted.

The declarations (called security declarations) start following the END command in a Master File and tell WebFOCUS that security is needed for the data source and what type of security you want. Each security declaration can consist of one or several of the following attributes:

- The DBA attribute gives the name or password of the Database Administrator for the data source. The Database Administrator has unlimited access to the data source and its Master File.
- The USER attribute identifies a user as a legitimate user of the data source. Only users whose name or password is specified in the Master File of a WebFOCUS data source with security placed on it have access to that data source.
- The ACCESS attribute defines the type of access a given user has. The four types of access available are:
 - RW, which allows a user to both read and write to a data source.
 - R, which allows a user to read data in a data source only.
 - W, which allows a user to write new segment instances to a data source only.
 - U, which allows a user to update records in a data source only.
- The RESTRICT attribute specifies certain segments or fields to which the user is not granted access. It can also be used to restrict the data values a user can see or perform transactions on.
- The NAME and VALUE attributes are part of the RESTRICT declaration.

You describe your data source security by specifying values for these attributes in a comma-delimited format, just as you specify any other attribute in the Master File.

The word END on a line by itself in the Master File terminates the segment and field attributes and indicates that the access limits follow. If you place the word END in a Master File, it must be followed by at least a DBA attribute.

Example

Implementing Data Source Security in a Master File

The following is a Master File that uses security features:

```

FILENAME = PERS, SUFFIX = FOC,$
SEGMENT = IDSEG, SEGTYPE = S1,$
  FIELD = SSN           ,ALIAS = SSN       ,FORMAT = A9   , $
  FIELD = FULLNAME     ,ALIAS = FNAME     ,FORMAT = A40  , $
  FIELD = DIVISION     ,ALIAS = DIV       ,FORMAT = A8   , $
SEGMENT=COMPSEG, PARENT=IDSEG, SEGTYPE=S1,$
  FIELD = SALARY       ,ALIAS = SAL       ,FORMAT = D8   , $
  FIELD = DATE         ,ALIAS = DATE     ,FORMAT = YMD  , $
  FIELD = INCREASE     ,ALIAS = INC       ,FORMAT = D6   , $
END
DBA=JONES76,$
USER=TOM      ,ACCESS=RW, $
USER=BILL    ,ACCESS=R  ,RESTRICT=SEGMENT ,NAME=COMPSEG , $
USER=JOHN    ,ACCESS=R  ,RESTRICT=FIELD  ,NAME=SALARY  , $
              ,ACCESS=R  ,RESTRICT=FIELD  ,NAME=INCREASE,$
USER=LARRY   ,ACCESS=U  ,RESTRICT=FIELD  ,NAME=SALARY  , $
USER=TONY    ,ACCESS=R  ,RESTRICT=VALUE ,NAME=IDSEG,
  VALUE=DIVISION EQ 'WEST' , $
USER=MARY    ,ACCESS=W  ,RESTRICT=VALUE ,NAME=SALTEST,
  VALUE=INCREASE+SALARY GE SALARY,$
              ,NAME=HISTTEST,
  VALUE=DIV NE ' ' AND DATE GT 0,$

```

Reference

Special Considerations for Data Source Security

When using the JOIN command, it is possible to bypass the DBA information in a data source. This is a security exposure created because in a JOIN structure the DBA information is read from the host Master File. This problem is solved by using the DBAFILE feature discussed in *Placing Security Information in a Central Master File* on page 9-17. All data sources in the joined structure will get security information as coded in the DBAFILE.

Identifying the DBA: The DBA Attribute

The first security attribute should be a password that identifies the Database Administrator. This password can be up to eight characters long. Since nothing else is needed, this line is terminated by the usual delimiter (,\$).

Note:

- Every data source having access limits must have a DBA.
- Groups of cross-referenced data sources must have the same DBA value.
- Partitioned data sources, which are read together in the USE command, must have the same DBA value.
- The Database Administrator has unlimited access to the data source and all cross-referenced data sources. Therefore, no field, segment, or value restrictions can be specified with the DBA attribute.
- You cannot encrypt and decrypt Master Files or restrict existing data sources without the DBA password.
- You should thoroughly test every security attribute before the data source is used. It is particularly important to test the VALUE limits to make sure they do not contain errors. Value tests are executed as if they were extra screening conditions or VALIDATE statements typed after each request statement. Since users are unaware of the value limits, errors caused by the value limits may confuse them.

Example

Identifying the DBA Using the DBA Attribute

```
DBA=JONES76,$
```

Procedure

Changing a DBA Password

The DBA has the freedom to change any of the security attributes. If you change the DBA password in the Master File, you must use the RESTRICT command for existing data sources (discussed in *Restricting an Existing FOCUS Data Source* on page 9-23) to inform each WebFOCUS data source affected by the change. Unless this is done, WebFOCUS will assume that the new description is an attempt to bypass the restriction rules. You use the following procedure for each data source affected:

1. Edit the Master File, changing the DBA value from old to new.
2. Issue the command:


```
SET PASS=old_DBA_value
```
3. Issue the command:


```
RESTRICT
filename
END
```
4. Issue the command:


```
SET PASS=new_value
```

Including the DBA Attribute in a HOLD File

With the SET HOLDSTAT command, you can identify a data source containing DBA information and comments to be automatically included in HOLD and PCHOLD Master Files. For more information about the SET HOLDSTAT command, see the *Developing Reporting Applications* manual.

Identifying Users With Access Rights: The USER Attribute

The USER attribute is a password that identifies the users who have legitimate access to the data source. A USER attribute cannot be specified alone; it must be followed by at least one ACCESS restriction (discussed in *Specifying an Access Type: The ACCESS Attribute* on page 9-9) to specify what sort of ACCESS the user is granted.

Before using a secured data source, a user must enter his or her password using the SET PASS command. If that password is not included in the Master File, the user is denied access to the data source. When the user does not have a password or has one that is inadequate for the type of access requested, the following message displays:

```
(FOC047) THE USER DOES NOT HAVE SUFFICIENT ACCESS RIGHTS TO THE FILE:  
filename
```

Syntax

How to Set the USER Attribute

Any user whose name or password is not declared in the Master File is denied access to that data source. The syntax of the USER attribute is

```
USER = name
```

where:

```
name
```

Is a password of up to eight characters for the user.

For example:

```
USER=TOM,...
```

You can specify a blank password. Such a password does not require the user to issue a SET PASS= command. A blank password may still have access limits and is convenient when a number of users have the same access rights. An example of setting a user's password to blank, and access to read only follows:

```
USER= , ACCESS=R,$
```

Establishing User Identity

A user must enter his or her password before using any WebFOCUS data source that has security specified for it. A single user may have different passwords in different files. For example, in file ONE the rights of password BILL apply, but in file TWO the rights of password LARRY apply. Use the SET PASS command to establish the passwords.

Syntax

How to Establish User Identity

```
SET {PASS|USER} = name [ [IN {file}* [NOCLEAR]} ] , name [IN file] ... ]
```

where:

name

Is the user's name or password.

file

Is the name of the Master File to which the password applies.

*

Indicates that *name* replaces all passwords active in all files.

NOCLEAR

Provides a way to replace all passwords in the list of active passwords while retaining the list.

Example Establishing User Identity

In the following example, the password TOM is in effect for all data sources that do not have a specific password designated for them:

```
SET PASS=TOM
```

For the next example, in file ONE the password is BILL, and in file TWO the password is LARRY. No other files have passwords set for them:

```
SET PASS=BILL IN ONE, LARRY IN TWO
```

Here, all files have password SALLY except files SIX and SEVEN, which have password DAVE:

```
SET PASS=SALLY, DAVE IN SIX  
SET PASS=DAVE IN SEVEN
```

The password is MARY in file FIVE and FRANK in all other files:

```
SET PASS=MARY IN FIVE,FRANK
```

A list of the files for which a user has set specific passwords is maintained. To see the list of files, issue:

```
? PASS
```

When the user sets a password IN * (all files), the list of active passwords collapses to one entry with no associated file name. To retain the file name list, use the NOCLEAR option.

In the next example, the password KEN replaces all passwords active in all files, and the table of active passwords is folded to one entry:

```
SET PASS=KEN IN *
```

In the following, MARY replaces all passwords in the existing table of active passwords (which consists of files NINE and TEN) but FRANK is the password for all other files. The option NOCLEAR provides a shorthand way to replace all passwords in a specific list:

```
SET PASS=BILL IN NINE,TOM IN TEN  
SET PASS=MARY IN * NOCLEAR,FRANK
```

Note: The FIND function does not work with COMBINED data sources secured with different passwords.

Users must issue their passwords using the SET PASS command during each session in which they use a secured data source. They may issue their passwords at any time before using the data source and can issue a different password afterward to access another data source.

Specifying an Access Type: The ACCESS Attribute

The ACCESS attribute specifies what sort of access a user is granted. Every security declaration, except the DBA declaration, must have a USER attribute and an ACCESS attribute.

The following is a complete security declaration, consisting of a USER attribute and an ACCESS attribute.

```
USER=TOM, ACCESS=RW,$
```

This declaration gives Tom read and write (for adding new segment instances) access to the data source.

You can assign the ACCESS attribute one of four values. These are:

```
ACCESS=R      Read only
```

```
ACCESS=W      Write only
```

```
ACCESS=RW     Read the data source and write new segment instances
```

```
ACCESS=U      Update only
```

Access levels affect what kind of commands a user can issue. Before you decide what access levels to assign to a user, you must consider what commands that user will need. If a user does not have sufficient access rights to use a given command, the following message displays:

```
(FOC047) THE USER DOES NOT HAVE SUFFICIENT ACCESS RIGHTS TO THE FILE:  
filename
```

ACCESS levels determine what a user can do to the data source. You use the RESTRICT attribute (discussed in *Limiting Data Source Access: The RESTRICT Attribute* on page 9-12) to limit the fields, values, or segments to which a user has access. Every USER attribute must be assigned an ACCESS attribute. The RESTRICT attribute is optional; without it, the user has unlimited access to fields and segments within the data source.

Types of Access

The type of access granting use of various WebFOCUS commands is shown in the following table. When more than one type of access is shown, any type of access marked will allow the user at least some use of that command. Often, however, the user will be able to use the command in different ways, depending on the type of access granted.

Command	R	W	RW	U	DBA
CHECK	X	X	X	X	X
CREATE			X		X
DECRYPT					X
DEFINE	X		X		X
ENCRYPT					X
MATCH	X		X		X
REBUILD			X		X
RESTRICT					X
TABLE	X		X		X

CHECK Command. Users without the DBA password or read/write access are allowed limited access to the CHECK command. However, when the HOLD option is specified, the warning ACCESS LIMITED BY PASSWORD is produced, and restricted fields are propagated to the HOLD file depending on the DBA RESTRICT attribute. Refer to *Limiting Data Source Access: The RESTRICT Attribute* on page 9-12 for more information on the RESTRICT attribute.

CREATE Command. Only users with the DBA password or read/write (RW) access rights can issue a CREATE command.

DECRYPT Command. Only users with the DBA password can issue a DECRYPT command.

DEFINE Command. As with all reporting commands, a user need only have an access of R (read only) to use the DEFINE command. An access of R permits the user to read records from the data source and prepare reports from them. The only users who cannot use the DEFINE command are those whose access is W (write only) or U (update only).

ENCRYPT Command. Only users with the DBA password can use the ENCRYPT command.

REBUILD Command. Only users with the DBA password or read/write (RW) access rights can issue the REBUILD command. This command is only for FOCUS data sources.

RESTRICT Command. Only users with the DBA password may use the RESTRICT command.

TABLE or MATCH Command. A user who has access of R or RW may use the TABLE command. Users with access of W or U may not.

Reference

RESTRICT Attribute Keywords

The RESTRICT attribute keywords affect the resulting HOLD file as follows:

FIELD

Fields named with the NAME parameter are not included in the HOLD file.

SEGMENT

The segments named with the NAME parameter are included, but fields in those segments are not.

SAME

The behavior is the same as for the user named in the NAME parameter.

NOPRINT

Fields named in the NAME or SEGNAME parameter are included since the user can reference these.

VALUE

Fields named in the VALUE parameter are included since the user can reference these.

If you issue the CHECK command with the PICTURE option, the RESTRICT attribute keywords affect the resulting picture as follows:

FIELD

Fields named with the NAME parameter are not included in the picture.

SEGMENT

The boxes appear for segments named with the NAME parameter, but fields in those segments do not.

SAME

The behavior is the same as for the user named in the NAME parameter.

NOPRINT

This option has no effect on the picture.

VALUE

This option has no effect on the picture.

Limiting Data Source Access: The RESTRICT Attribute

The ACCESS attribute determines what a user can do with a data source. The optional RESTRICT attribute further restricts a user's access to certain fields, values, or segments.

Syntax

How to Limit Data Source Access

```
...RESTRICT=level, NAME={name|SYSTEM} [, VALUE=test], $
```

where:

level

Can be one of the following:

FIELD specifies that the user cannot access the fields named with the NAME parameter.

SEGMENT specifies that the user cannot access the segments named with the NAME parameter.

SAME specifies that the user has the same restrictions as the user named in the NAME parameter. No more than four nested SAME users are valid.

NOPRINT specifies that the field named in the NAME or SEGMENT parameter can be mentioned in a request statement, but will not be displayed.

name

Is the name of the field or segment you wish to restrict. When used after NOPRINT, this can be only a field name. NAME=SYSTEM, which can only be used only with value tests, restricts every segment in the data source, including descendant segments. Multiple fields or segments can be specified by issuing the RESTRICT attribute several times for one user.

VALUE

Specifies that the user can have access to only those values that meet the test described in the *test* parameter.

test

Is the value test that the data must meet before the user can have access to it.

Example

Limiting Data Source Access

```
USER=BILL ,ACCESS=R ,RESTRICT=SEGMENT ,NAME=COMPSEG, $
```

Restricting Access to a Field or a Segment

The RESTRICT attribute identifies the segments or fields that the user will not be able to access. Anything not named in the RESTRICT attribute will be accessible.

Without the RESTRICT attribute, the user has access to the entire data source. Users may be limited to reading, writing, or updating new records, but every record in the data source is available for the operation.

Syntax

How to Restrict Access to a Field or a Segment

```
...RESTRICT=level, NAME=name, $
```

where:

level

Can be one of the following:

FIELD specifies that the user cannot access the fields named with the NAME parameter.

SEGMENT specifies that the user cannot access the segments named with the NAME parameter.

SAME specifies that the user has the same restrictions as the user named in the NAME parameter.

NOPRINT specifies that the field named in the NAME or SEGMENT parameter can be mentioned in a request statement but will not be displayed. When used after NOPRINT, NAME can be only a field name.

name

Is the name of the field or segment you wish to restrict. When used after NOPRINT, this can be only a field name.

NAME=SYSTEM, which can be used only with value tests, restricts every segment in the data source, including descendant segments. Multiple fields or segments can be specified by issuing the RESTRICT attribute several times for one user.

Note:

- If a field or segment is mentioned in the NAME attribute, it cannot be retrieved by the user. If such a field or segment is mentioned in a request statement, it will be rejected as beyond the user's access rights. With NOPRINT, the field or segment can be mentioned, but the data will not be displayed. The data will appear as blanks for alphanumeric format or zeroes for numeric fields.
- You can restrict multiple fields or segments by providing multiple RESTRICT statements. For example, if you wish to restrict Harry from using both field A and segment B, you issue the following access limits:

```
USER=HARRY, ACCESS=R,          RESTRICT=FIELD , NAME=A,$  
                                RESTRICT=SEGMENT, NAME=B,$
```

- You can restrict as many segments and fields as you like.
- Using RESTRICT=SAME is a convenient way to reuse a common set of restrictions for more than one password. If you specify RESTRICT=SAME and provide a user name or password as it is specified in the USER attribute for the NAME value, the new user will be subject to the same restrictions as the one named in the NAME attribute. You can then add additional restrictions, as they are needed.

Example

Restricting Access to a Segment

In the following example, Bill has read-only access to everything in the data source except the COMPSEG segment:

```
USER=BILL ,ACCESS=R ,RESTRICT=SEGMENT ,NAME=COMPSEG,$
```

Example

Reusing a Common Set of Access Restrictions

In the following example, both Sally and Harry have the same access privileges as BILL. In addition, Sally is not allowed to read the SALARY field.

```
USER=BILL ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,  
          VALUE=DIVISION EQ 'WEST', $  
USER=SALLY ,ACCESS=R ,RESTRICT=SAME ,NAME=BILL,  
                   RESTRICT=FIELD ,NAME=SALARY,$  
USER=HARRY ,ACCESS=R ,RESTRICT=SAME ,NAME=BILL, $
```

Note: A restriction on a segment also affects access to its descendants.

Restricting Access to a Value

You can also restrict the values to which a user has access by providing a test condition in your RESTRICT attribute. The user is restricted to using only those values that satisfy the test condition.

You can restrict values in one of two ways: you can restrict the values the user can read from the data source, or you can restrict what the user can write to a data source. These restrictions are two separate functions: one does not imply the other. You use the ACCESS attribute to specify whether the values the user reads or the values the user writes are restricted.

You restrict the values a user can read by setting ACCESS=R and RESTRICT=VALUE. This type of restriction prevents the user from seeing any data values other than those that meet the test condition provided in the RESTRICT attribute. A RESTRICT attribute with ACCESS=R functions as an involuntary IF statement in a report request. Therefore, the syntax for ACCESS=R value restrictions must follow the rules for an IF test in a report request.

Syntax

How to Restrict Values a User Can Read

```
...ACCESS=R, RESTRICT=VALUE, NAME=name, VALUE=test,$
```

where:

name

Is the name of the segment on which you are performing the tests. To specify all segments in the data source, specify NAME=SYSTEM.

test

Is the test being performed.

Example

Restricting Values a User Can Read

```
USER=TONY ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,  
VALUE=DIVISION EQ 'WEST' , $
```

With this restriction, Tony can see records from only the western division.

You type the test expression after VALUE=. The syntax of the test condition is the same as that used by the TABLE command to screen records, except the word IF does not precede the phrase. (Screening conditions in the TABLE command are discussed in the *Creating Reports With WebFOCUS Language* manual.) Should several fields have tests performed on them, separate VALUE attributes must be provided. Each test must name the segment to which it applies. For example:

```
USER=DICK ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,  
VALUE=DIVISION EQ 'EAST' OR 'WEST' , $  
NAME=IDSEG,  
VALUE=SALARY LE 10000 , $
```

If a single test condition exceeds the allowed length of a line, it can be provided in sections. Each section must start with the attribute VALUE= and end with the terminator (,\$). For example:

```
USER=SAM, ACCESS=R, RESTRICT=VALUE ,NAME=IDSEG,  
VALUE=DIVISION EQ 'EAST' OR 'WEST' , $  
VALUE=OR 'NORTH' OR 'SOUTH' , $
```

Note: The second and subsequent lines of a value restriction must begin with the keyword OR.

You can apply the test conditions to the parent segments of the data segments on which the tests are applicable. Consider the following example:

```
USER=DICK ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,  
VALUE=DIVISION EQ 'EAST' OR 'WEST' , $  
NAME=IDSEG,  
VALUE=SALARY LE 10000 , $
```

The field named SALARY is actually part of a segment named COMPSEG. Since the test is specified with NAME=IDSEG, however, the test is made effective for requests on its parent, IDSEG. In this case, the request PRINT FULLNAME would print the full names of only people who meet this test, that is, whose salary is less than or equal to \$10,000, even though the test is performed on a field that is part of a descendant segment of IDSEG. If, however, the test was made effective on COMPSEG, that is, NAME=COMPSEG, then the full name of everyone in the data source could be retrieved, but with the salary information of only those meeting the test condition.

Example

Restricting Both Read and Write Values for a User

```
USER=TILLY ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,  
VALUE=DIVISION EQ 'NORTH' , $  
ACCESS=W ,RESTRICT=VALUE ,NAME=DIVTEST,  
VALUE=DIVISION EQ 'NORTH' , $
```

Placing Security Information in a Central Master File

The DBAFILE attribute enables you to place all of the passwords and restrictions for many Master Files in one central file. Each individual Master File points to this central control file. Groups of Master Files with the same DBA password may share a common DBAFILE which itself has the same DBA password.

There are several benefits to this technique. The primary ones are:

- Passwords have to be stored only once when they are applicable to a group of data sources. This simplifies password administration.
- Data sources with different DBA passwords can now be JOINed or COMBINED. In addition, individual DBA information remains in effect for each data source in a JOIN or COMBINE.

The central DBAFILE is a standard Master File. Other Master Files can use the password and security restrictions listed in the central file by specifying its file name with the DBAFILE attribute.

Note:

- All Master Files that specify the same DBAFILE have the same DBA password.
- The central DBAFILE may include additional attributes before the END statement that signifies the presence of DBA information. The DBA password in the DBAFILE is the same as the password in all the Master Files that refer to it. This prevents individuals from substituting their own security. All of these Master Files should be encrypted.
- The DBAFILE may contain a list of passwords and restrictions following the DBA password. These passwords apply to all data sources that reference this DBAFILE. In the example above, PASS=BILL, with ACCESS=R (read only), applies to all data sources that contain the attribute DBAFILE=FOUR.
- After the common passwords, the DBAFILE may specify data source-specific passwords and additions to general passwords. You implement this feature by including FILENAME attributes in the DBA section of the DBAFILE (for example, FILENAME=TWO). Consult *File Naming Requirements for DBAFILE* on page 9-19 for additional information about the FILENAME attribute.
- Data source-specific restrictions override general restrictions for the specified data source. In the case of a conflict, passwords in the FILENAME section take precedence. For example, a DBAFILE might contain ACCESS=RW in the common section, but specify ACCESS=R for the same password by including a FILENAME section for a particular data source.
- Value restrictions accumulate; all value restrictions must be satisfied before retrieval. In the preceding example, note the two occurrences of PASS=JOE. JOE is a common password for all data sources, but in FILENAME=THREE it carries an extra restriction, RESTRICT=..., which applies only to data source THREE.

Syntax

How to Place Security Attributes in a Central Master File

```
END
DBA=dbname, DBAFILE=filename , $
```

where:

dbname

Is the same as the *dbname* in the central file.

filename

Is the name of the central file.

You can specify passwords and restrictions in a DBAFILE that apply to every Master File that points to that DBAFILE; you can also include passwords and restrictions for specific Master Files by including FILENAME attributes in the DBAFILE.

Example

Placing Security Attributes in a Central Master File

The following example shows a group of Master Files that share a common DBAFILE named FOUR:

```
ONE MASTER
FILENAME=ONE
.
.
END
DBA=ABC, DBAFILE=FOUR, $
TWO MASTER
FILENAME=TWO
.
.
END
DBA=ABC, DBAFILE=FOUR, $
THREE MASTER
FILENAME=THREE
.
.
END
DBA=ABC,
DBAFILE=FOUR, $
FOUR MASTER
FILENAME=FOUR, $
SEGNAME=mmmmm, $
FIELDNAME=fffff, $
END
DBA=ABC, $
    PASS=BILL, ACCESS=R, $
    PASS=JOE, ACCESS=R, $
FILENAME=TWO, $
    PASS=HARRY, ACCESS=RW, $
FILENAME=THREE, $
    PASS=JOE, ACCESS=R, RESTRICT=... , $
    PASS=TOM, ACCESS=R, $
```

File Naming Requirements for DBAFILE

When a DBAFILE includes a FILENAME attribute for a specific Master File, the FILENAME attribute in the referencing Master File must be the same as the FILENAME attribute in the DBA section of the DBAFILE. This prevents users from renaming a Master File to a name not known by the DBAFILE.

Example

DBAFILE Naming Conventions

```
ONE MASTER
FILENAME=XONE
.
.
.
END
DBA=ABC, DBAFILE=FOUR, $
```

```
FOUR MASTER
FILENAME=FOUR
.
.
.
END
DBA=ABC, $
.
.
.
FILENAME=XONE, $
.
.
.
```

ONE MASTER is referred to in requests as TABLE FILE ONE. However, both ONE MASTER and the DBA section of the DBAFILE, FOUR MASTER, specify FILENAME=XONE.

Connection to an Existing DBA System With DBAFILE

If there is no mention of the new attribute, DBAFILE, there will be no change in the characteristics of an existing system. In the current system, when a series of data sources is JOINed, the first data source in the list is the controlling data source. Its passwords are the only ones examined. For a COMBINE, only the last data source's passwords take effect. All data sources must have the same DBA password.

In the new system, the DBA sections of all data sources in a JOIN or COMBINE are examined. If DBAFILE is included in a Master File, then its passwords and restrictions are read. To make the DBA section of a data source active in a JOIN list or COMBINE, specify DBAFILE for that data source.

Once you start to use the new system, you should convert all of your Master Files. For Database Administrators who want to convert existing systems but do not want a separate physical DBAFILE, the DBAFILE attribute can specify the data source itself.

Example

Connecting to an Existing DBA System With DBAFILE

```
FILENAME=SEVEN,  
  SEGNAME=..  
  FIELDNAME=...  
  .  
  .  
  .  
END  
DBA=ABC,DBAFILE=SEVEN,$      (OR DBAFILE= ,)$  
  PASS=...  
  PASS=...
```

Combining Applications With DBAFILE

Since each data source now contributes its own restrictions, you can now JOIN and COMBINE data sources that come from different applications and have different DBA passwords. The only requirement is a valid password for each data source. You can therefore grant access rights for one application to an application under the control of a different DBA by assigning a password in your system.

You can assign screening conditions to a data source that are automatically applied to any report request that accesses the data source. See the *Creating Reports With WebFOCUS Language* manual for details.

Summary of Security Attributes

The following is a list of all the security attributes used in WebFOCUS:

Attribute	Alias	Maximum Length	Meaning
DBA	DBA	8	Value assigned is code name of the Database Administrator (DBA) who has unrestricted access to the data source.
USER	PASS	8	Values are arbitrary code names, identifying users for whom security restrictions will be in force.
ACCESS	ACCESS	8	Levels of access for this user. Values are: R read only W write new segments only RW read and write U update values only
RESTRICT	RESTRICT	8	Types of restrictions to be imposed for this access level. Values are: SEGMENT FIELD VALUE SAME NOPRINT
NAME	NAME	66	Name of segment or field restricted or of the program to be called.
VALUE	VALUE	80	Test expression which must be true when RESTRICT=VALUE is the type of limit.
DBAFILE	DBAFILE	8	Names the Master File that contains passwords and restrictions to use.

Hiding Restriction Rules: The ENCRYPT Command

Since the restriction information for a WebFOCUS data source is stored in its Master File, you will want to encrypt the Master File in order to prevent users from examining the restriction rules. Only the Database Administrator can encrypt a description. Thus, you must set PASS=DBAname before you issue the ENCRYPT command. The syntax of the ENCRYPT command varies from operating system to operating system.

Syntax

How to Hide Restriction Rules: ENCRYPT Command

```
ENCRYPT FILE filename
```

where:

```
filename
```

Is the name of the file to be encrypted.

Example

Encrypting and Decrypting a Master File

The following is an example of the complete procedure:

```
SET PASS=JONES76  
ENCRYPT FILE PERS
```

The process can be reversed if you wish to change the restrictions. The command to restore the description to a readable form is DECRYPT.

The DBA password must be issued with the SET command before the file can be decrypted. For example:

```
SET PASS=JONES76  
DECRYPT FILE PERS
```

Encrypting Data

You may also use the ENCRYPT command within the Master File to encrypt some or all of its segments. When encrypted files are stored on their external media (disk or tape) they are secure from unauthorized examination.

Encryption takes place on the segment level; that is, the entire segment is encrypted. The request for encryption is made in the Master File by setting the attribute ENCRYPT to ON.

Example

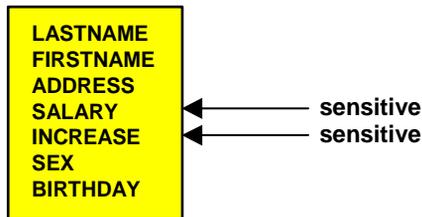
Encrypting Data

```
SEGMENT=COMPSEG, PARENT=IDSEG, SEGTYPE=S1, ENCRYPT=ON,$
```

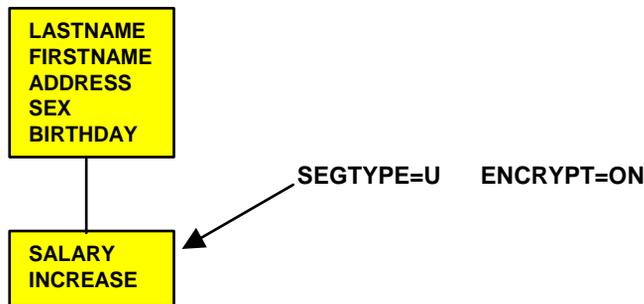
You must specify the ENCRYPT attribute before you enter any data in the data source. The message NEW FILE... must appear when the encryption is first requested. Encryption cannot be requested later by a change to the Master File and cannot be removed once it has been requested and any data has been entered in the data source.

Performance Considerations for Encrypted Data

There is a small loss in processing efficiency when data is encrypted. You can minimize this loss by grouping the sensitive data fields together on a segment and making them a separate segment of `SEGTYPE=U`, unique segment, beneath their original segment. For example, suppose the data items on a segment are:



They should be grouped as:



Restricting an Existing FOCUS Data Source

If you have existing files to which you want to add security limitations, you need to use the `RESTRICT` command.

(Note: This is not the `RESTRICT` attribute described in *Limiting Data Source Access: The RESTRICT Attribute* on page 9-12.)

If an existing WebFOCUS data source has no DBA rules and you want to add DBA rules, perform the following:

1. Edit the Master File and add the DBA rules.
2. Issue the `RESTRICT` command to write the DBA password to the data source.

Note: The `RESTRICT` command cannot be used for non-WebFOCUS data sources.

Setting a Password Externally

Passwords can also be set automatically by an external security system such as RACF[®], CA-ACF2[®], or CA-Top Secret[®]. Passwords issued this way are set when WebFOCUS is first entered and may be permanent (that is, not alterable by subsequent SET USER, SET PASS or -PASS commands); or they may be default passwords that can be subsequently overridden; or they may be permanent for some users, defaults for other users, and not set at all for yet other users.

The advantage of setting WebFOCUS passwords externally is that the password need not be known by the user, does not require prompting, and does not have to be embedded in a PROFILE FOCEXEC or an encrypted FOCEXEC.

Passwords set this way must match the passwords specified in the Master Files of the data sources being accessed.

WebFOCUS FOCEXEC Security

Most data security issues are best handled by WebFOCUS DBA exit routines. For more information about WebFOCUS DBA exit routines, see the *WebFOCUS Security and Administration* manual. However, an additional data security facility that can be incorporated within Dialogue Manager is to encrypt and decrypt FOCEXECs.

Encrypting and Decrypting a FOCEXEC

You may want to keep the actual text of a stored FOCEXEC confidential while allowing users to execute the FOCEXEC. You may want to do this either because there is confidential information stored in the FOCEXEC or because you do not want the FOCEXEC changed by unauthorized users. You can protect a stored FOCEXEC from unauthorized users with the ENCRYPT command.

Any user can execute an encrypted FOCEXEC, but you must decrypt the FOCEXEC to view it. Only a user with the DBA password can decrypt the FOCEXEC.

Syntax

How to Encrypt and Decrypt a FOCEXEC

You use the following procedure to encrypt the FOCEXEC named SALERPT:

```
SET PASS = DOHIDE  
ENCRYPT FILE SALERPT FOCEXEC
```

You use the following procedure to decrypt the FOCEXEC named SALERPT:

```
SET PASS = DOHIDE  
DECRYPT FILE SALERPT FOCEXEC
```

CHAPTER 10

Creating and Rebuilding a Data Source

Topics:

- Creating a New Data Source: The CREATE Command
- Rebuilding a Data Source: The REBUILD Command
- Optimizing File Size: The REBUILD Subcommand
- Changing Data Source Structure: The REORG Subcommand
- Indexing Fields: The INDEX Subcommand
- Creating an External Index: The EXTERNAL INDEX Subcommand
- Checking Data Source Integrity: The CHECK Subcommand
- Changing the Data Source Creation Date and Time: The TIMESTAMP Subcommand
- Converting Legacy Dates: The DATE NEW Subcommand
- Migrating to a Fusion Data Source: The MIGRATE Subcommand
- Creating a Fusion Multi-Dimensional Index: The MDINDEX Subcommand

You can create a new data source, or reinitialize an existing data source, using the CREATE command.

Once a data source exists, you may find it necessary to reorganize it in order to use disk space more effectively; to change the contents, index, or structure of the data source; to change legacy date fields to smart date fields; or to convert a FOCUS data source to a Fusion data source. You can do all of this and more using the REBUILD command.

You can use the CREATE and REBUILD commands with FOCUS and Fusion data sources. You can also use the CREATE command to create relational tables for which you have the appropriate data adapter.

Creating a New Data Source: The CREATE Command

You can create a new, empty FOCUS data source for a Master File using the CREATE command. You can also use the CREATE command to erase the data in an existing FOCUS data source.

The CREATE command also works for Fusion data sources and, with the appropriate data adapter installed, a relational table (such as a DB2 or Teradata table). For information, see the documentation for the relevant data adapter.

Note that on MVS, you must issue a CREATE command for a new data source. For all other platforms, if the data source has not been initialized, a CREATE is automatically issued on the first MODIFY or Maintain request made against the data source.

Syntax

How to Use the CREATE Command

```
CREATE FILE mastername
```

where:

mastername

Is the name of the Master File that describes the data source.

After you enter the CREATE command, the following displays:

```
NEW FILE name ON date AT time
```

where:

name

Is the complete name of the new data source.

ON *date* AT *time*

Is the date and time at which the data source was created or recreated.

When you issue the CREATE command, if the data source already exists, the following message displays:

```
(FOC441) WARNING. THE FILE EXISTS ALREADY. CREATE WILL WRITE OVER IT  
REPLY :
```

To erase the data source and create a new, empty data source, enter Y. To cancel the command and leave the data source intact, enter END, QUIT, or N.

If you wish to give the data source absolute File Integrity protection, issue the following command prior to the CREATE command:

```
SET SHADOW=ON
```

Note: IBM no longer guarantees that CMS file mode A6 ensures absolute file integrity.

Note the following when issuing CREATE on MVS:

- If you do not allocate the data source prior to issuing the CREATE command, the data source is created as a temporary data set. To retain the data source, copy it to a permanent data set with the DYNAM COPY command.
- The CREATE command pre-formats the primary space allocation and initializes the data source entry in the File Directory Table. A Master File must exist for the data source in a PDS allocated to ddname MASTER.
- Issuing MODIFY or Maintain commands against uninitialized data sources (those for which no CREATE was issued) results in a read error.

Example

Recreating a FOCUS Data Source in Windows NT

To recreate the EMPLOYEE data source, issue the following command:

```
CREATE FILE EMPLOYEE
```

The following message displays:

```
(FOC441) WARNING. THE FILE EXISTS ALREADY. CREATE WILL WRITE OVER IT  
REPLY :
```

You would reply:

```
YES
```

The following message displays:

```
NEW FILE C:EMPLOYEE.FOC      ON 03/02/2001 AT 15.48.57
```

The EMPLOYEE data source still exists on disk, but it contains no records.

Rebuilding a Data Source: The REBUILD Command

You can make a structural change to a FOCUS or Fusion data source after it has been created using the REBUILD command. Using REBUILD and one of its subcommands REBUILD, REORG, INDEX, EXTERNAL INDEX, CHECK, TIMESTAMP, DATE NEW, MDINDEX, and MIGRATE you can:

- Rebuild a disorganized data source (REBUILD).
- Delete instances according to a set of screening conditions (REBUILD or REORG).
- Redesign an existing data source. This includes adding and deleting segments, adding and deleting data fields, indexing different fields, changing the size of alphanumeric data fields and more (REORG).
- Index up to seven new fields before rebuilding or creating the data source (INDEX).
- Create an external index database that facilitates indexed retrieval when joining or locating records (EXTERNAL INDEX).
- Check the structural integrity of the data source (CHECK). Check when the FOCUS data source was last changed (TIMESTAMP).
- Convert legacy date formats to smart date formats (DATE NEW).
- Build or modify a multi-dimensional index for Fusion data sources (MDINDEX).
- Convert FOCUS Master Files and data sources to Fusion Master Files and data sources (MIGRATE).

You can use the REBUILD facility:

- **As a batch procedure**, by entering the REBUILD command, the desired subcommand, and any responses to subcommand prompts on separate lines of a procedure.

Before using the REBUILD facility, you should be aware of several requirements and recommendations regarding file allocation, security authorization, and backup.

Reference

Before You Use REBUILD: Requirements and Recommendations

Before you use the REBUILD facility, there are two requirements that you need to consider:

- **Allocation.** Usually, you do not have to allocate workspace prior to using a REBUILD command; it is automatically allocated. However, adequate workspace must be available. As a rule of thumb, have space 10 to 20% larger than the size of your file available for the REBUILD and REORG options.

The file name REBUILD is always assigned to the workspace. In the DUMP phase of the REORG command, the allocation statement is displayed in case you want to perform the LOAD phase at a different time.

- **Security authorization.** If the data source you are rebuilding is protected by a database administrator, you must be authorized for read and write access in order to perform any REBUILD activity. For more information on data source security, see Chapter 9, *Providing Data Source Security: DBA*.

Although it is not a requirement, we recommend that you:

- **Backup.** Create a backup copy of the original Master File and data source before using any of the REBUILD subcommands.

Procedure

How to Use the REBUILD Facility

The following steps describe how to use the REBUILD facility:

1. Initiate the REBUILD facility by entering:

```
REBUILD
```

2. Select a subcommand by supplying its name or its number. The following list shows the subcommand names and their corresponding numbers:

1. REBUILD	(Optimize the database structure)
2. REORG	(Alter the database structure)
3. INDEX	(Build/modify the database index)
4. EXTERNAL INDEX	(Build/modify an external index database)
5. CHECK	(Check the database structure)
6. TIMESTAMP	(Change the database timestamp)
7. DATE NEW	(Convert old date formats to smart date formats)
8. MDINDEX	(Build/modify a multidimensional index. FUSION only)
9. MIGRATE	(Convert FOCUS masters/DBs to FUSION)

Your subsequent responses depend on the subcommand you select. Generally, you will only need to give the name of the data source and possibly one or two other items of information.

Controlling the Frequency of REBUILD Messages

When REBUILD processes a data source, it displays status messages periodically (for example, REFERENCE..AT SEGMENT 1000) to inform you of the progress of the rebuild. The default display interval is every 1000 segment instances processed during REBUILD's retrieval and load phases. The number of messages displayed is determined by the number of segment instances in the FOCUS data source being rebuilt, divided by the display interval.

Syntax

How to Control the Frequency of REBUILD Messages

REBUILD displays a message (REFERENCE..AT SEGMENT *segnum*) at periodic intervals to inform you of its progress as it processes a data source. You can control the frequency with which REBUILD displays this message by issuing the command

```
SET REBUILDMSG = {n|1000}
```

where:

n

Is any integer from 1,000 to 99,999,999 or 0 (to disable the messages).

A setting of less than 1000:

- Generates a warning message that describes the valid values (0 or greater than 999).
- Keeps the current setting. The current setting will be either the default of 1000, or the last valid integer greater than 999 that REBUILDMSG was set to.

Example

Controlling the Display of REBUILD Messages

The following messages are generated for a REBUILD CHECK where REBUILDMSG has been set to 4000, and the data source contains 19,753 records.

```
STARTING..
REFERENCE..AT SEGMENT      4000
REFERENCE..AT SEGMENT      8000
REFERENCE..AT SEGMENT     12000
REFERENCE..AT SEGMENT     16000
NUMBER OF SEGMENTS RETRIEVED= 19753
CHECK COMPLETED...
```

Optimizing File Size: The REBUILD Subcommand

You use the REBUILD subcommand for one of two reasons. Primarily, you use it to improve data access time and storage efficiency. After many deletions, the physical structure of your data does not match the logical structure. REBUILD REBUILD dumps data into a temporary workspace and then reloads it, putting instances back in their proper logical order. A second use of REBUILD REBUILD is to delete segment instances according to a set of screening conditions.

Normally, you use the REBUILD subcommand as a way of maintaining a clean data source. To check if you need to rebuild your data source, enter the ? FILE command (described in *Creating a Fusion Multi-Dimensional Index: The MDINDEX Subcommand* on page 10-32):

```
? FILE filename
```

If your data source is disorganized, the following message appears:

```
FILE APPEARS TO NEED THE -REBUILD-UTILITY
REORG PERCENT IS A MEASURE OF FILE DISORGANIZATION
0 PCT IS PERFECT -- 100 PCT IS BAD
REORG PERCENT x%
```

This message appears whenever the REORG PERCENT measure is more than 30%. The REORG PERCENT measure indicates the degree to which the physical placement of data in the data source differs from its logical, or apparent, placement.

The &FOCDISORG variable can be used immediately after the ? FILE command and also shows the percentage of disorganization in a data source. &FOCDISORG will show a data source's percentage of disorganization even if it is below 30% (see the *Developing Reporting Applications* manual).

Procedure

How to Use the REBUILD Subcommand

The following steps describe how to use the REBUILD subcommand:

1. Initiate the REBUILD facility by entering:

```
REBUILD
```

The following options are available:

- | | |
|-------------------|--|
| 1. REBUILD | (Optimize the database structure) |
| 2. REORG | (Alter the database structure) |
| 3. INDEX | (Build/modify the database index) |
| 4. EXTERNAL INDEX | (Build/modify an external index database) |
| 5. CHECK | (Check the database structure) |
| 6. TIMESTAMP | (Change the database timestamp) |
| 7. DATE NEW | (Convert old date formats to smartdate formats) |
| 8. MDINDEX | (Build/modify a multidimensional index. FUSION only) |
| 9. MIGRATE | (Convert FOCUS masters/DBs to FUSION) |

2. Select the REBUILD subcommand by entering:

```
REBUILD OR 1
```

3. Enter the name of the data source to be rebuilt.

On MVS, enter the *ddname*.

On CMS, enter *filename filetype filemode*.

On UNIX, Windows NT/2000, and OpenVMS, enter *filename*. The data source to be rebuilt will be referenced by a USE command. If no USE command is in effect, the data source will be searched for using the EDAPATH variable.

4. If you are simply rebuilding the data source and require no selection tests, enter:

```
NO
```

The REBUILD procedure will begin immediately.

If you wish to place screening conditions on the REBUILD subcommand, enter:

```
YES
```

Then enter the necessary selection tests, ending the last line with ,\$.

Test relations of EQ, NE, LE, GE, LT, GT, CO (contains), and OM (omits) are permitted. Tests are connected with the word AND, and lists of literals may be connected with the OR operator. A comma followed by a dollar sign (,\$) is required to terminate any test. For example, you might enter the following:

```
A EQ A1 OR A2 AND B LT 100 AND  
C GT 400 AND D CO 'CUR' , $
```

Statistics are displayed when the REBUILD REBUILD procedure is complete, including the number of segments retrieved and the number of segments included in the rebuilt data source.

Example

Using the REBUILD Subcommand in Windows NT

The following procedure:

1. REBUILD
2. REBUILD
3. EMPLOYEE
4. NO

1. Initiates the REBUILD facility.
2. Specifies the REBUILD subcommand.
3. Provides the name of the data source to rebuild.
4. Indicates that no record selection tests are required.

The data source will be rebuilt and the appropriate statistics will be generated.

Changing Data Source Structure: The REORG Subcommand

The REORG subcommand enables you to make a variety of changes to the Master File *after* data has been entered in the FOCUS data source. REBUILD REORG is a two-step procedure that first dumps the data into a temporary workspace and then reloads it under a new Master File.

You can use REBUILD REORG to:

- Add descendant segments to an existing segment.
- Delete segments.
- Add data fields to an existing segment.

Note: The fields must be added after the key fields.

- Delete data fields.
- Change the order of non-key data fields within a segment. Key fields may not be changed.
- Promote fields from unique segments to parent segments.
- Demote fields from parent segments to descendant unique segments.
- Index different fields.
- Increase or decrease the size of an alphanumeric data field.

REBUILD REORG will not enable you to:

- Change field format types (alphanumeric to numeric and vice versa, changing numeric format types).
- Change the value for SEGNAME attributes.
- Change the value for SEGTYPE attributes.
- Change field names that are indexed.

Procedure How to Use the REORG Subcommand

The following steps describe how to use the REORG subcommand:

1. Before making any changes to the original Master File, make a copy of it with another name.
2. Using an editor, make the desired edits to the copy of the Master File.
3. Initiate the REBUILD facility by entering:

```
REBUILD
```

The following options are available:

1. REBUILD (Optimize the database structure)
2. REORG (Alter the database structure)
3. INDEX (Build/modify the database index)
4. EXTERNAL INDEX (Build/modify an external index database)
5. CHECK (Check the database structure)
6. TIMESTAMP (Change the database timestamp)
7. DATE NEW (Convert old date formats to smartdate formats)
8. MDINDEX (Build/modify a multidimensional index. FUSION only)
9. MIGRATE (Convert FOCUS masters/DBs to FUSION)

4. Select the REORG subcommand by entering:

```
REORG OR 2
```

The options are:

1. DUMP (DUMP contents of the database)
2. LOAD (LOAD data into the database)

5. Initiate the DUMP phase of the procedure by entering:

```
DUMP OR 1
```

6. Enter the name of the data source you wish to reorganize. Be sure to use the name of the original Master File for this phase.

On MVS, enter the *ddname*.

On CMS, enter *filename filetype filemode*.

On UNIX, Windows NT/2000, and OpenVMS, enter *filename*. The data source to be rebuilt will be referenced by a USE command. If no USE command is in effect, the data source will be searched for using the EDAPATH variable.

7. You can specify selection tests by entering YES. Only data that meets your specifications will be dumped. It is more likely, however, that you will want to dump the entire data source. To do so, enter:

```
NO
```

Statistics are displayed during the DUMP procedure, including the number of segments dumped and the name and statistics for the temporary file used to hold the data.

8. Once the DUMP phase is complete, you are ready to begin the second phase of REBUILD REORG: LOAD. Enter:

```
REBUILD
```

9. Select the REORG subcommand by entering:

```
REORG OF 2
```

The options are:

1. DUMP (DUMP contents of the database)
2. LOAD (LOAD data into the database)

10. Initiate the LOAD phase of the procedure by entering:

```
LOAD OF 2
```

11. Enter the name of the data source you wish to load from the temporary file created during the dump phase. In most cases, this will be the new data source name.

You will see statistics when the REBUILD REORG procedure is complete, including the number of segments input.

At this stage, you have loaded the specified data from the original Master File into a new data source with the name you specified. It is important to remember that both the original Master File and data source remain. You have three choices:

- You may want to rename the original Master and data source to prevent possible confusion.
- You may rename the new Master and data source to the original name. As a result, any existing FOCEXECs referencing the original name will run against the new data source.
- You may delete the original Master and data source after you verify the new Master and data source are correct and complete.

On non-MVS platforms, if you enter the name of a data source that already exists, (the original Master File) you are notified that you will be appending data to a preexisting data source and asked if you wish to continue.

In MVS, you are not asked if you want to append to an existing data source; the data source is created. If you want to append, when you issue the LOAD command, enter LOAD NOCREATE.

Enter N to terminate REBUILD execution. Enter Y to add the records in the temporary REBUILD file to the original FOCUS data source.

If duplicate field names occur in a Master File, REBUILD REORG is not supported.

In MVS, you must issue a CREATE command for a new data source being loaded.

Example

Using the REORG Subcommand in Windows NT

The following procedure:

1. COPY EMPLOYEE.FOC EMPOLD.FOC
2. REBUILD
3. REORG
4. DUMP
5. EMPLOYEE
6. NO
7. ERASE EMPLOYEE.FOC
8. REBUILD
9. REORG
10. LOAD
11. EMPLOYEE

1. Makes a copy of the data source.
2. Initiates the REBUILD facility.
3. Specifies the REORG subcommand.
4. Initiates the DUMP phase.
5. Specifies the name of the data source to dump.
6. Indicates that no record selection tests are required.

The data source will be dumped and the appropriate statistics will be generated.

7. Erases the EMPLOYEE data source.
8. Initiates the REBUILD facility.
9. Specifies the REORG subcommand.
10. Initiates the LOAD phase.
11. Specifies the name of the data source to load.

The data source will be loaded and the appropriate statistics will be generated.

Indexing Fields: The INDEX Subcommand

To index a field after you have entered data into the data source, use the INDEX subcommand. You can index up to seven fields in addition to those previously specified in the Master File or since the last REBUILD or CREATE command. The only requirement is that each field specified must be described with the FIELDTYPE=I (or INDEX=I) attribute in the Master File. If you add more than seven index fields, REBUILD INDEX displays the following message:

```
(FOC720) THE NUMBER OF INDEXES ADDED AFTER FILE CREATION EXCEEDS 7
```

The INDEX option uses the operating system sort program. You must have disk space to which you can write. To calculate the amount of space needed, add 8 to the length of the index field in bytes and multiply the sum by twice the number of segment instances

```
(LENGTH + 8) * 2n
```

where:

n

Is the number of segment instances.

You may decide to wait until after loading data to add the FIELDTYPE=I attribute and index the field. This is because the separate processes of loading data and indexing can be faster than performing both processes at the same time when creating the data source. This is especially true for large data sources.

Sort libraries and workspace must be available. The REBUILD allocates default sort workspace in MVS, if you have not already allocated it. DDNAMEs SORTIN and SORTOUT must be allocated prior to issuing a REBUILD INDEX.

In CMS, the following command identifies a specific sort product

```
SET SORTLIB = sortname
```

where:

sortname

Can be SYNCSORT, DFSORT, or VMSORT.

A CMS GLOBAL TXTLIB command must be issued prior to REBUILD INDEX to identify the location of the sort program. If the GLOBAL TXTLIB command and the SET SORTLIB commands are not issued:

- The sort program is searched for in SORTLIB TXTLIB.
- A GLOBAL TXTLIB SORTLIB command is automatically issued.

If SORTLIB TXTLIB is not available at the time of REBUILD INDEX, the following message displays:

```
(FOC263) EXTERNAL FUNCTION OR LOAD MODULE NOT FOUND: SORT
```

Procedure **How to Use the INDEX Subcommand**

The following steps describe how to use the INDEX subcommand:

1. Add the FIELDTYPE=I attribute to the field or fields you are indexing in the Master File.
2. Initiate the REBUILD facility by entering:

```
REBUILD
```

The following options are available:

1. REBUILD (Optimize the database structure)
2. REORG (Alter the database structure)
3. INDEX (Build/modify the database index)
4. EXTERNAL INDEX (Build/modify an external index database)
5. CHECK (Check the database structure)
6. TIMESTAMP (Change the database timestamp)
7. DATE NEW (Convert old date formats to smartdate formats)
8. MDINDEX (Build/modify a multidimensional index. FUSION only)
9. MIGRATE (Convert FOCUS masters/DBs to FUSION)

3. Select the INDEX subcommand by entering:

```
INDEX OR 3
```

4. Enter the name of the Master File in which you will add the INDEX=I or FIELDTYPE=I attribute.
5. Enter the name of the field to index. If you are indexing all the fields that have FIELDTYPE=I, enter an asterisk (*).

Statistics are displayed when the REBUILD INDEX procedure is complete, including the field names that were indexed and the number of index values included.

Example **Using the INDEX Subcommand in Windows NT**

The following procedure:

1. REBUILD
2. INDEX
3. EMPLOYEE
4. EMP_ID

1. Initiates the REBUILD facility.
2. Specifies the INDEX subcommand.
3. Specifies the name of the Master File.
4. Specifies the name of the field to index.

The field will be indexed and the appropriate statistics will be generated.

Creating an External Index: The EXTERNAL INDEX Subcommand

Users with READ access to a local FOCUS data source can create an index database that facilitates indexed retrieval when joining or locating records. An external index is a FOCUS data source that contains index, field, and segment information for one or more specified FOCUS or Fusion data sources. The external index is independent of its associated FOCUS or Fusion data source. External indexes offer equivalent performance to permanent indexes for retrieval and analysis operations. External indexes enable indexing on concatenated FOCUS data sources, indexing on real and defined fields, and indexing selected records from WHERE/IF tests. External indexes are created as temporary data sets unless pre-allocated to a permanent data set. They are not updated as the indexed data changes.

You create an external index with the REBUILD command. Internally, REBUILD begins a process which reads the databases that make up the index, gathers the index information, and creates an index database containing all field, format, segment, and location information.

You provide information about:

- Whether you want to add new records from a concatenated database to the index database.
- The name of the external index database that you want to build.
- The name of the data source from which the index information is obtained.
- The name of the field from which the index is to be created.
- Whether you want to position the index field within a particular segment.
- Any valid WHERE or IF record selection tests.

Sort libraries and workspace must be available. The REBUILD allocates default sort workspace in MVS, if you have not already. DDNAMEs SORTIN and SORTOUT must be allocated prior to issuing a REBUILD.

Procedure **How to Use the EXTERNAL INDEX Subcommand**

To create an external index from a concatenated database, follow these steps:

1. Assume that you have the following USE in effect:

```
USE CLEAR *
USE
EMPLOYEE
EMP2 AS EMPLOYEE
JOBFILE
EDUCFILE
END
```

Note that EMPLOYEE and EMP2 are concatenated and can be described by the EMPLOYEE Master File.

2. Initiate the REBUILD facility by entering:

```
REBUILD
```

The following options are available:

1. REBUILD (Optimize the database structure)
2. REORG (Alter the database structure)
3. INDEX (Build/modify the database index)
4. EXTERNAL INDEX (Build/modify an external index database)
5. CHECK (Check the database structure)
6. TIMESTAMP (Change the database timestamp)
7. DATE NEW (Convert old date formats to smartdate formats)
8. MDINDEX (Build/modify a multidimensional index. FUSION only)
9. MIGRATE (Convert FOCUS masters/DBs to FUSION)

3. Select the EXTERNAL INDEX subcommand by entering:

```
EXTERNAL INDEX OR 4
```

4. Specify whether to create a new index data source or add to an existing one by entering one of the following choices:

NEW
ADD

For this example, assume you are creating a new index database and respond by entering:

NEW

5. Specify the name of the external index database:

EMPIDX

6. Specify the name of the data source from which the index records are obtained:

EMPLOYEE

7. Specify the name of the field to index:

CURR_JOBCODE

8. Specify whether the index should be associated with a particular field by entering YES or NO. For this example, enter:

NO

9. Indicate whether you require any record selection tests by entering YES or NO.

For this example, enter:

NO

If you responded YES, you would next enter the record selection tests, ending them with the END command on a separate line.

For example:

```
IF DEPARTMENT EQ 'MIS'  
END
```

You will see statistics (output of the ? FDT query) about the index data source when the REBUILD EXTERNAL INDEX procedure is complete. This query is automatically issued at the end of the REBUILD EXTERNAL INDEX process in order to validate the contents of the index database.

Concatenating Index Databases

The external index feature enables indexed retrieval from concatenated FOCUS data sources. If you wish to concatenate databases that comprise the index, you must issue the appropriate USE command prior to the REBUILD. The USE must include all cross-referenced and LOCATION files. REBUILD EXTERNAL INDEX contains an add function that enables you to append new index records from a concatenated database to the index database, eliminating the need to recreate the index database.

The original data source from which the index was built may not be in the USE list when you add index records. If it is, REBUILD EXTERNAL INDEX generates the following message:

```
(FOC999) WARNING. EXTERNAL INDEX COMPONENT REUSED: ddname
```

Positioning Indexed Fields

The external index feature is useful for positioning retrieval of indexed values for defined fields within a particular segment in order to enhance retrieval performance. By entering at a lower segment within the hierarchy, data retrieved for the indexed field is affected, as the index field is associated with data outside its source segment. This enables the creation of a relationship between the source and target segments. The source segment is defined as the segment that contains the indexed field. The target segment is defined as any segment above or below the source segment within its path.

If the target segment is not within the same path, the following message is generated:

```
(FOC974) EXTERNAL INDEX ERROR. INVALID TARGET SEGMENT
```

A defined field may not be positioned at a higher segment.

While the source segment can be a cross-referenced or location segment, the target segment cannot be a cross-referenced segment. If an attempt is made to place the target on a cross-referenced segment, the following message is generated:

```
(FOC1000) INVALID USE OF CROSS REFERENCE FIELD
```

If you choose not to associate your index with a particular field, the source and target segments will be the same.

Activating an External Index

After building an external index database, you must associate it with the data sources from which it was created. This is accomplished with the USE command. The syntax is the same as when USE is issued prior to building the external index database, except the WITH or INDEX option is required.

Syntax

How to Activate an External Index

```
USE [ADD|REPLACE]
database_name [AS mastername]

index_database_name [WITH|INDEX] mastername
.
.
.
END
```

where:

ADD

Appends one or more new databases to the present USE list. Without the ADD option, the existing USE list is cleared and then replaced by the current list of USE databases.

REPLACE

Replaces an existing *database_name* in the USE list.

database_name

Is the name of the data source.

On MVS, enter the *ddname*.

On CMS, enter *filename filetype filemode*.

On UNIX, Windows NT/2000, and OpenVMS, enter *filename*. The data source to be rebuilt will be referenced by a USE command. If no USE command is in effect, the data source will be searched for using the EDAPATH variable.

You must include a data source name in the USE list for all cross-referenced and LOCATION files that are specified in the Master File.

AS

Is used with a Master File name to concatenate data sources.

mastername

Specifies the Master File.

index_database_name

Is the name of the external index database.

On MVS, enter the *ddname*.

On CMS, enter *filename filetype filemode*.

On UNIX, Windows NT/2000, and OpenVMS, enter [*pathname*]*filename.foc*. The data source to be rebuilt will be referenced by a USE command. If no USE command is in effect, the data source will be searched for using the EDAPATH variable.

WITH|INDEX

Is a keyword that creates the relationship between the component data sources and the index database. INDEX is a synonym for WITH.

Reference

Special Considerations for REBUILD EXTERNAL INDEX

Consider the following when working with external indexes:

- Up to eight indexes can be activated at one time in a USE list using the WITH statement. More than eight indexes may be activated in a session if you issue the USE CLEAR command and issue new USE statements.
- Up to 256 concatenated files may be indexed. However, only eight indexes may be activated at one time.
- Verification of the component files is now done for both the date and time stamp of file creation. Files with the same date and time stamp that are copied display the following messages:

```
(FOC995) ERROR. EXTERNAL INDEX DUPLICATE COMPONENT: fn  
REBUILD ABORTED
```

- MODIFY may use the external index with only the FIND or LOOKUP functions. The external index cannot be used as an entry point, such as:

```
MODIFY FILE filename.indexfld
```
- Indexes may not be created on field names longer than twelve characters.
- Text fields may not be used as indexed fields.
- The USE options NEW, READ, ON, LOCAL, and AS *master* ON *userid* READ are not supported for the external index database.
- The external index database need not be allocated since CREATE FILE automatically does a temporary allocation. If a permanent database is required then an allocation for the index database must be in place prior to the REBUILD EXTERNAL INDEX command.
- SORTIN and SORTOUT, work files that the REBUILD EXTERNAL INDEX process creates, must be allocated with adequate space. In order to estimate the space needed, use the following formula:

```
bytes = (field_length + 20) * number_of_occurrences
```

Checking Data Source Integrity: The CHECK Subcommand

It is rare for the structural integrity of a FOCUS data source to be damaged. Structural damage will occasionally occur, however, during a drive failure or if an incorrect Master File is used. In this situation, the REBUILD CHECK command performs two essential tasks:

- It checks pointers in the data source.
- Should it encounter an error, it displays a message and attempts to branch around the offending segment or instance.

Although CHECK is able to report on a good deal of data that would otherwise be lost, it is important to remember that frequently backing up your FOCUS data sources is the best method of preventing data loss.

CHECK will occasionally fail to uncover structural damage. If you have reason to believe that there is damage to your data source, though CHECK reports otherwise, there is a second method of checking data source integrity. This method entails using the ? FILE and TABLEF commands. Though this is not a REBUILD function, it is included at the end of this section because of its relevancy to CHECK.

Procedure How to Use the CHECK Subcommand

The following steps describe how to use the CHECK subcommand:

1. Initiate the REBUILD facility by entering:

```
REBUILD
```

The following options are available:

1. REBUILD (Optimize the database structure)
2. REORG (Alter the database structure)
3. INDEX (Build/modify the database index)
4. EXTERNAL INDEX (Build/modify an external index database)
5. CHECK (Check the database structure)
6. TIMESTAMP (Change the database timestamp)
7. DATE NEW (Convert old date formats to smartdate formats)
8. MDINDEX (Build/modify a multidimensional index. FUSION only)
9. MIGRATE (Convert FOCUS masters/DBs to FUSION)

2. Select the CHECK subcommand by entering:

```
CHECK OR 5
```

3. Enter the name of the data source to be checked.

On MVS, enter the *ddname*.

On CMS, enter *filename filetype filemode*.

On UNIX, Windows NT/2000, and OpenVMS, enter *filename*. The data source to be rebuilt will be referenced by a USE command. If no USE command is in effect, the data source will be searched for using the EDAPATH variable.

Statistics are displayed during the REBUILD CHECK procedure:

- If no errors are found, the statistics indicate the number of segments retrieved.
- If errors are found, the statistics indicate the type and location of each error:
 - DELETE indicates that the data has been deleted and should not have been retrieved.
 - OFFPAGE indicates that the address of the data is not on a page owned by this segment.
 - INVALID indicates that the type of linkage cannot be identified. It may be a destroyed portion of the data source.

Example Using the Check Subcommand (File Undamaged) in Windows NT

The following procedure:

1. REBUILD
2. CHECK
3. EMPLOYEE

1. Initiates the REBUILD facility.
2. Specifies the CHECK subcommand.
3. Provides the name of the data source to check.

The data source will be checked and the appropriate statistics will be generated.

Confirming Structural Integrity Using ? FILE and TABLEF

When you believe that there is damage to your data source, though REBUILD CHECK reports there is not, use the ? FILE and TABLEF commands to compare the number of segment instances reported after invoking each command. A disparity indicates a structural problem.

Procedure

How to Verify REBUILD CHECK Using ? FILE and TABLEF

1. Issue the following command:

```
? FILE filename
```

where:

```
filename
```

Is the name of the FOCUS data source you are examining.

A report displays information on the status of the data source. The number of instances for each segment is listed in the ACTIVE COUNT column.

2. To ensure that the TABLEF command in the next step counts all segment instances, including those in the short paths, issue the command:

```
SET ALL = ON
```

3. Enter:

```
TABLEF FILE filename  
COUNT field1 field2  
END
```

where:

```
filename
```

Is the name of the Master File of the FOCUS data source.

```
field1...
```

Are the names of fields in the data source. Name one field from each segment. It does not matter which field is named in the segment.

The report produced shows the number of field occurrences for those fields named and thus the number of segment instances for each segment. These numbers should match their respective segment instance numbers shown in the ? FILE command (except for unique segments which the TABLEF command shows to have as many instances are in the parent segment). If the numbers do not match, or if either the ? FILE command or TABLEF command ends abnormally, the data source is probably damaged.

Example

Checking the Integrity of the EMPLOYEE Data Source

User input is shown in bold; computer responses are in uppercase:

```
? FILE
STATUS OF FOCUS FILE: c:employee.foc   ON 05/13/2001 AT 16.17.32
      ACTIVE  DELETED   DATE OF    TIME OF    LAST TRANS
SEGNAME      COUNT    COUNT      LAST CHG   LAST CHG   NUMBER
EMPINFO           12                05/13/1999 16.17.22   448
FUNDTRAN          6                05/13/1999 16.17.22   12
PAYINFO           19                05/13/1999 16.17.22   19
ADDRESS           21                05/13/1999 16.17.22   21
SALINFO           70                05/13/1999 16.17.22  448
DEDUCT            448               05/13/1999 16.17.22  448
TOTAL SEGS        576
TOTAL CHAR        8984
TOTAL PAGES       8
LAST CHANGE                05/13/1999 16.17.22   448

SET ALL = ON
TABLEF FILE EMPLOYEE
COUNT EMP_ID BANK_NAME DAT_INC TYPE PAY_DATE DED_CODE
END

PAGE      1

      EMP_ID  BANK_NAME  DAT_INC  TYPE  PAY_DATE  DED_CODE
      COUNT   COUNT      COUNT    COUNT  COUNT     COUNT
      -----  -----  -----  -----  -----  -----
           12          12         19      21      70        448

NUMBER OF RECORDS IN TABLE=      488 LINES= 1
```

Note that the BANK_NAME count in the TABLEF report is different than the number of FUNDTRAN instances reported by the ? FILE query. This is because FUNDTRAN is a unique segment and is always considered present as an extension of its parent.

Changing the Data Source Creation Date and Time: The *TIMESTAMP* Subcommand

A FOCUS data source's date and time stamp are updated each time the data source is changed by CREATE, REBUILD, Maintain, or MODIFY. You can update a data source's date and time stamp without making changes to the data source by using REBUILD's *TIMESTAMP* subcommand.

Procedure

How to Use the *TIMESTAMP* Subcommand

The following steps describe how to use the *TIMESTAMP* subcommand:

1. Initiate the REBUILD facility by entering:

```
REBUILD
```

The following options are available:

1. REBUILD (Optimize the database structure)
2. REORG (Alter the database structure)
3. INDEX (Build/modify the database index)
4. EXTERNAL INDEX (Build/modify an external index database)
5. CHECK (Check the database structure)
6. *TIMESTAMP* (Change the database timestamp)
7. DATE NEW (Convert old date formats to smartdate formats)
8. MDINDEX (Build/modify a multidimensional index. FUSION only)
9. MIGRATE (Convert FOCUS masters/DBs to FUSION)

2. Select the *TIMESTAMP* subcommand by entering:

```
TIMESTAMP OR 6
```

3. Enter the name of the data source whose date and time stamp is to be updated.

On MVS, enter the *ddname*.

On CMS, enter *filename filetype filemode*.

On UNIX, Windows NT/2000, and OpenVMS, enter *filename*. The data source to be rebuilt will be referenced by a USE command. If no USE command is in effect, the data source will be searched for using the EDAPATH variable.

4. Enter one of the following options for the source of the date and time:

T (today's date). Updates the data source's date and time stamp with the current date and time.

D (search file for date). Updates the data source's date and time stamp with the last date and time at which the data source was actually changed. Each page of the data source is scanned and the most recent date and time recorded for a page is applied to the data source. This is the same as issuing the ? FILE query, and can be time consuming when the data source is very large. This option is used to keep an external index database synchronized with its component data source.

MMDDYY HHMMSS. Is a date and time that you specify, which REBUILD will use to update the data source's date and time stamp. The date and time that you enter must have the format mmddy hhmss or mmddyyy hhmss. There must be a space between the date and the time. If you use two digits for the year, REBUILD uses the values for DEFCENT and YRTHRESH to determine the century.

If you supply an invalid date or time, the following message displays:

```
(FOC961) INVALID DATE INPUT IN REBUILD TIME:
```

Converting Legacy Dates: The DATE NEW Subcommand

The REBUILD subcommand DATE NEW converts legacy dates (alphanumeric, integer, and packed-decimal fields with date display options) to smart dates (fields in date format) in your FOCUS data sources. The utility uses update-in-place technology. It updates your data source and creates a new Master File, yet does not change the structure or size of the data source. You must back up the data source before executing REBUILD with the DATE NEW subcommand. We recommend that you run the utility against the copy and then replace the original file with the updated backup.

How DATE NEW Converts Legacy Dates

REBUILD's DATE NEW subcommand overwrites the original legacy date field (an alphanumeric, integer, or packed-decimal field with date display options) with a smart date (a field in date format). When the storage size of the legacy date exceeds four bytes (the storage size of a smart date), a pad field is added to the data source following the date field:

- Formats A6YMD, A6MDY, and A6DMY are changed to formats YMD, MDY, and DMY, respectively, and have a 2-byte pad field added to the Master File.
- The storage size of integer dates (I6YMD, I6MDY, for example) is 4 bytes, so no pad field is added.
- All packed fields and A8 dates add a 4-byte pad field.

When a date is a key field (but not the last key for the segment), and it requires a pad field, the number of keys in the SEGTYPE is increased by one for each date field that requires padding.

DATE NEW changes only legacy dates to smart dates. The field's format in the Master File must be one of the following (month translation edit options T and TR may be included in the format):

```
A8YYMD A8MDYY A8DMYY A6YMD A6MDY A6DMY A6YYM A6MYM A4YM A4MY  
I8YYMD I8MDYY I8DMYY I6YMD I6MDY I6DMY I6YYM I6MYM I4YM I4MY  
P8YYMD P8MDYY P8DMYY P6YMD P6MDY P6DMY P6YYM P6MYM P4YM P4MY
```

If you have a field that stores date values but does not have one of these formats, DATE NEW does not change it. If you have a field with one of these formats that you do not want changed, temporarily delete the date edit options from the format, run REBUILD DATE NEW, and then restore the edit options to the format.

Reference

DATE NEW Usage Notes

- The DBA password for the data source must be issued prior to issuing REBUILD.
- The original Master File cannot be encrypted.
- All files must be available locally during the REBUILD, including LOCATION files.
- The Master File cannot have GROUP fields.
- Some error numbers are available in &FOCERRNUM while all error numbers are available in &&FOCREBUILD. Test both &&FOCREBUILD and &FOCERRNUM for errors when writing procedures to rebuild your data sources.
- To avoid any potential problems, clear all LETs and JOINS before issuing REBUILD.
- DEFCENT/YRTHRESH are respected at the global, data source, and field level.
- Correct all invalid date values in the data source before executing REBUILD/DATE NEW. The utility converts all invalid dates to zero. Invalid dates used as keys may lead to duplicate keys in the data source.
- Adequate workspace must be available for the temporary REBUILD file. As a rule of thumb, have space 10 to 20% larger than the size of the existing file available.
- REBUILD/INDEX is performed automatically if an index exists.
- REBUILD/REBUILD is performed automatically after REBUILD/DATE NEW when any key is a date.
- Sort libraries and workspace must be available (as with REBUILD/INDEX). The REBUILD allocates default sort workspace in MVS, if you have not already. DDNAMEs SORTIN and SORTOUT must be allocated prior to issuing a REBUILD.

What DATE NEW Does Not Convert

REBUILD's DATE NEW subcommand is a remediation tool for your FOCUS data sources and date fields only. It does not remediate:

- DEFINE attributes in the Master File.
- ACCEPT attributes in the Master File.
- DBA restrictions (for example, VALUE restrictions) in the Master File or central security repository (DBAFILE).
- Cross-references to other date fields in this or other Master Files.
- Any references to date fields in your FOCEXEC.

Example

Using the DATE NEW Subcommand in Windows NT

The following procedure:

```
1. SET DFC = 19, YRT = 50
2. REBUILD
3. DATE NEW
4. NEWEMP.MAS
5. YES
```

1. Sets the DEFCENT and YRTHRESH parameters that determine which century to use for each date.
2. Initiates the REBUILD facility.
3. Specifies the DATE NEW subcommand.
4. Provides the name of the Master File that specifies the dates to convert.
5. Indicates that the data source has been backed up.

The dates will be converted and the appropriate statistics will be generated, including the number of segments changed.

The new Master File is an updated copy of the original Master File except that:

- The USAGE format for legacy date fields is updated to delete the format and length. The date edit options are retained. For example, A6YMDTR becomes YMDTR.
- Padding fields are added for those dates that need them:

```
FIELDNAME= ,ALIAS= ,FORMAT=An,$ PAD FIELD ADDED BY REBUILD
```

where *n* is the padding length (either 2 or 4). Note that the FIELDNAME and ALIAS are blank.

- The SEGTYPE attribute is updated for segments that have remediated dates as keys when the date requires padding and the date is not the last field in the key. The SEGTYPE number will be increased by the number of pad fields added to the key.
- If the SEGTYPE is missing for any segment, the following line is added immediately prior to the \$ terminator for that segment:

```
SEGTYPE= segtype,$ OMITTED SEGTYPE ADDED BY REBUILD
```

where *segtype* is determined by REBUILD.

- If the USAGE attribute for any field—including date fields—is missing, the following line is added, immediately prior to the \$ terminator for that field:

```
USAGE= fmt,$ OMITTED USAGE ADDED BY REBUILD
```

where *fmt* is the format of the previous field in the Master File. REBUILD automatically assigns the previous field's format to any field coded without an explicit USAGE= statement.

Example

Sample Master File: Before and After Conversion by DATE NEW

Before Conversion

```
FILE=filename
SEGNAME=segname, SEGTYPE=S2
FIELD=KEY1,,USAGE=A6YMD,$

FIELD=KEY2,,USAGE=I6MDY,$
FIELD=FIELD3,,USAGE=A8YYMD,$
```

After Conversion

```
FILE=filename
SEGNAME=segname, SEGTYPE=S3
FIELD=KEY1,,USAGE= YMD,$
FIELD=, ,USAGE=A2,$ PAD FIELD ADDED BY
REBUILD
FIELD=KEY2,,USAGE= MDY,$
FIELD=FIELD3,,USAGE= YYMD,$
FIELD=, ,USAGE=A4,$ PAD FIELD ADDED BY
REBUILD
```

When REBUILD's DATE NEW subcommand converts this Master File:

- The SEGTYPE changes from an S2 to S3 to incorporate a 2-byte pad field.
- Format A6YMD changes to smart date format YMD.
- A 2-byte pad field with a blank field name and alias is added to the Master File.
- Format I6MDY changes to smart date format MDY (no padding needed).
- Format A8YYMD changes to smart date format YYMD.
- A 4-byte pad field with a blank field name and alias is added to the Master File.

Action Taken on a Date Field During REBUILD/DATE NEW

REBUILD/DATE NEW performs a REBUILD/REBUILD or REBUILD/INDEX automatically when a date field is a key or a date field is indexed. The following chart shows the action taken on a date field during the REBUILD/DATE NEW process.

Date Is a Key	Index	Result
No	None	NUMBER OF SEGMENTS CHANGED = <i>n</i>
No	Yes	REBUILD/INDEX on date field.
Yes	None	REBUILD/REBUILD is performed.
Yes	On any field	REBUILD/REBUILD is performed. REBUILD/INDEX is performed for the indexed fields.

Migrating to a Fusion Data Source: The MIGRATE Subcommand

The MIGRATE subcommand is useful for migrating selected portions or an entire FOCUS data source to Fusion. For a more detailed explanation of how to use this subcommand, see the Fusion documentation.

Creating a Fusion Multi-Dimensional Index: The MDINDEX Subcommand

The MDINDEX subcommand is used to create or maintain a multi-dimensional index within a Fusion data source. For a more detailed explanation of how to use this subcommand, see the Fusion documentation.

APPENDIX A

Master Files and Diagrams

Topics:

- EMPLOYEE Data Source
- JOBFILE Data Source
- EDUCFILE Data Source
- SALES Data Source
- CAR Data Source
- LEDGER Data Source
- FINANCE Data Source
- REGION Data Source
- COURSE Data Source
- EMPDATA Data Source
- TRAINING Data Source
- VideoTrk, MOVIES, and ITEMS Data Sources
- VIDEOTR2 Data Source
- Gotham Grinds Data Sources
- Century Corp Data Sources

This appendix contains descriptions and structure diagrams for the sample data sources used throughout the documentation.

EMPLOYEE Data Source

EMPLOYEE contains sample data about a company's employees. Its segments are:

EMPINFO

Contains employee IDs, names, and positions.

FUNDTRAN

Specifies employees' direct deposit accounts. This segment is unique.

PAYINFO

Contains the employee's salary history.

ADDRESS

Contains employees' home and bank addresses.

SALINFO

Contains data on employees' monthly pay.

DEDUCT

Contains data on monthly pay deductions.

EMPLOYEE also contains cross-referenced segments belonging to the JOBFILE and EDUCFILE files, also described in this appendix. The segments are:

JOBSEG (from JOBFILE)

Describes the job positions held by each employee.

SKILLSEG (from JOBFILE)

Lists the skills required by each position.

SECSEG (from JOBFILE)

Specifies the security clearance needed for each job position.

ATTNDSEG (from EDUCFILE)

Lists the dates that employees attended in-house courses.

COURSEG (from EDUCFILE)

Lists the courses that the employees attended.

EMPLOYEE Master File

```

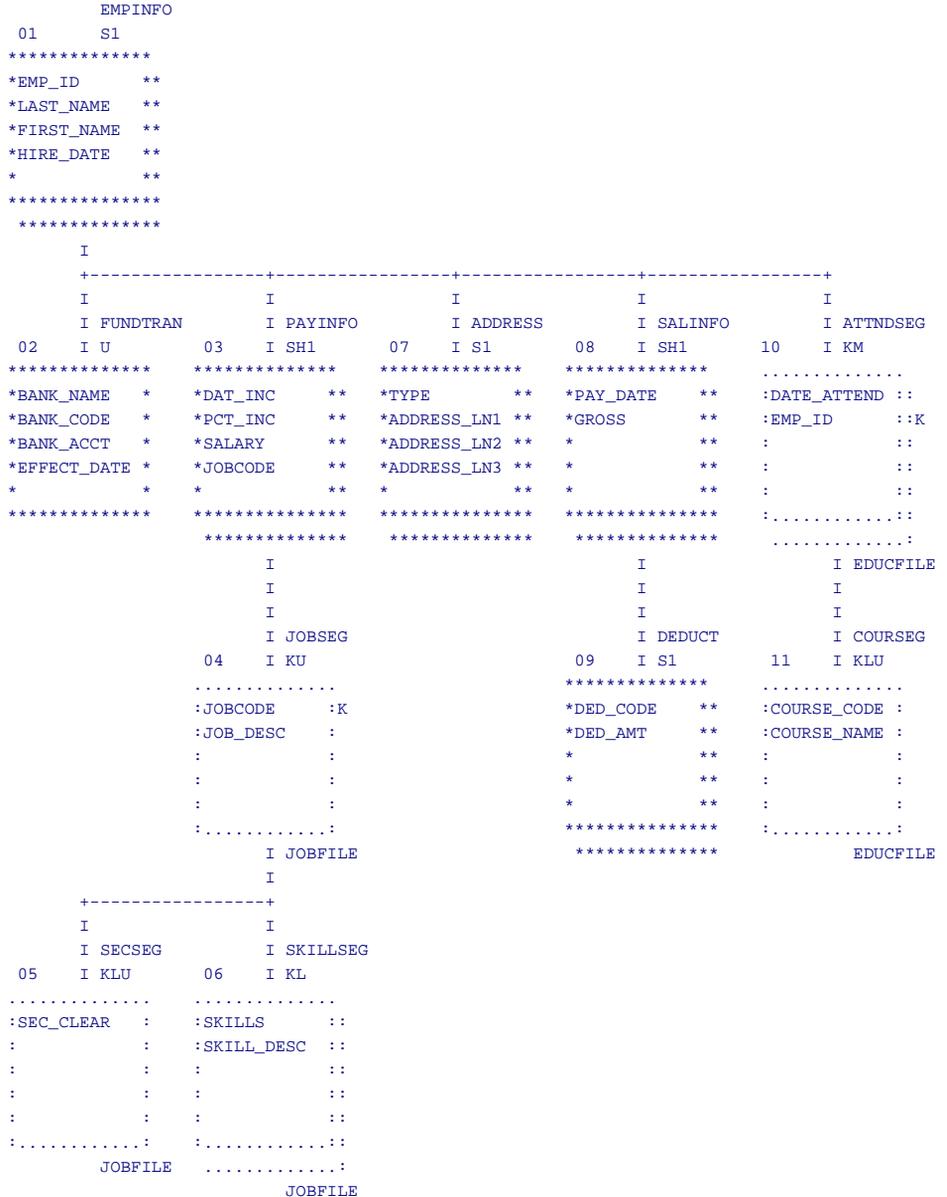
FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9, $
  FIELDNAME=LAST_NAME, ALIAS=LN, FORMAT=A15, $
  FIELDNAME=FIRST_NAME, ALIAS=FN, FORMAT=A10, $
  FIELDNAME=HIRE_DATE, ALIAS=HDT, FORMAT=I6YMD, $
  FIELDNAME=DEPARTMENT, ALIAS=DPT, FORMAT=A10, $
  FIELDNAME=CURR_SAL, ALIAS=CSAL, FORMAT=D12.2M, $
  FIELDNAME=CURR_JOBCODE, ALIAS=CJC, FORMAT=A3, $
  FIELDNAME=ED_HRS, ALIAS=OJT, FORMAT=F6.2, $
SEGNAME=FUNDTRAN, SEGTYPE=U, PARENT=EMPINFO
  FIELDNAME=BANK_NAME, ALIAS=BN, FORMAT=A20, $
  FIELDNAME=BANK_CODE, ALIAS=BC, FORMAT=I6S, $
  FIELDNAME=BANK_ACCT, ALIAS=BA, FORMAT=I9S, $
  FIELDNAME=EFFECT_DATE, ALIAS=EDATE, FORMAT=I6YMD, $
SEGNAME=PAYINFO, SEGTYPE=SH1, PARENT=EMPINFO
  FIELDNAME=DAT_INC, ALIAS=DI, FORMAT=I6YMD, $
  FIELDNAME=PCT_INC, ALIAS=PI, FORMAT=F6.2, $
  FIELDNAME=SALARY, ALIAS=SAL, FORMAT=D12.2M, $
  FIELDNAME=JOBCODE, ALIAS=JBC, FORMAT=A3, $
SEGNAME=ADDRESS, SEGTYPE=S1, PARENT=EMPINFO
  FIELDNAME=TYPE, ALIAS=AT, FORMAT=A4, $
  FIELDNAME=ADDRESS_LN1, ALIAS=LN1, FORMAT=A20, $
  FIELDNAME=ADDRESS_LN2, ALIAS=LN2, FORMAT=A20, $
  FIELDNAME=ADDRESS_LN3, ALIAS=LN3, FORMAT=A20, $
  FIELDNAME=ACCTNUMBER, ALIAS=ANO, FORMAT=I9L, $
SEGNAME=SALINFO, SEGTYPE=SH1, PARENT=EMPINFO
  FIELDNAME=PAY_DATE, ALIAS=PD, FORMAT=I6YMD, $
  FIELDNAME=GROSS, ALIAS=MO_PAY, FORMAT=D12.2M, $
SEGNAME=DEDUCT, SEGTYPE=S1, PARENT=SALINFO
  FIELDNAME=DED_CODE, ALIAS=DC, FORMAT=A4, $
  FIELDNAME=DED_AMT, ALIAS=DA, FORMAT=D12.2M, $
SEGNAME=JOBSEG, SEGTYPE=KU., PARENT=PAYINFO, CRFILE=JOBFILE,
  CRKEY=JOBCODE, $
SEGNAME=SECSEG, SEGTYPE=KLU, PARENT=JOBSEG, CRFILE=JOBFILE, $
SEGNAME=SKILLSEG, SEGTYPE=KL, PARENT=JOBSEG, CRFILE=JOBFILE, $
SEGNAME=ATTNDSEG, SEGTYPE=KM, PARENT=EMPINFO, CRFILE=EDUCFILE,
  CRKEY=EMP_ID, $
SEGNAME=COURSEG, SEGTYPE=KLU, PARENT=ATTNDSEG, CRFILE=EDUCFILE, $

```

EMPLOYEE Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE EMPLOYEE ON 09/15/00 AT 10.16.27



JOBFILE Data Source

JOBFILE contains sample data about a company's job positions. Its segments are:

JOBSEG

Describes what each position is. The field JOBCODE in this segment is indexed.

SKILLSEG

Lists the skills required by each position.

SECSEG

Specifies the security clearance needed, if any. This segment is unique.

JOBFILE Master File

```

FILENAME=JOBFILE, SUFFIX=FOC
SEGNAME=JOBSEG, SEGTYPE=S1
  FIELDNAME=JOBCODE, ALIAS=JC, FORMAT=A3, INDEX=I,$
  FIELDNAME=JOB_DESC, ALIAS=JD, FORMAT=A25, $
SEGNAME=SKILLSEG, SEGTYPE=S1, PARENT=JOBSEG
  FIELDNAME=SKILLS, ALIAS=, FORMAT=A4, $
  FIELDNAME=SKILL_DESC, ALIAS=SD, FORMAT=A30, $
SEGNAME=SECSEG, SEGTYPE=U, PARENT=JOBSEG
  FIELDNAME=SEC_CLEAR, ALIAS=SC, FORMAT=A6, $

```

JOBFILE Structure Diagram

SECTION 01

STRUCTURE OF FOCUS

FILE JOBFILE ON 01/05/96 AT 14.40.06

```

          JOBSEG
01      S1
*****
*JOBCODE  **I
*JOB_DESC **
*          **
*          **
*          **
*****
          I
          +-----+
          I          I
          I SECSEG  I SKILLSEG
02      I U          03      I S1
*****          *****
*SEC_CLEAR *      *SKILLS  **
*          *      *SKILL_DESC **
*          *      *          **
*          *      *          **
*          *      *          **
*****          *****
                      *****

```

EDUCFILE Data Source

EDUCFILE contains sample data about a company's in-house courses. Its segments are:

COURSESEG

Contains data on each course.

ATTNDSEG

Specifies which employees attended the courses. Both fields in the segment are key fields. The field EMP_ID in this segment is indexed.

EDUCFILE Master File

```
FILENAME=EDUCFILE, SUFFIX=FOC
SEGNAME=COURSESEG, SEGTYPE=S1
  FIELDNAME=COURSE_CODE, ALIAS=CC, FORMAT=A6, $
  FIELDNAME=COURSE_NAME, ALIAS=CD, FORMAT=A30, $
SEGNAME=ATTNDSEG, SEGTYPE=SH2, PARENT=COURSESEG
  FIELDNAME=DATE_ATTEND, ALIAS=DA, FORMAT=I6YMD, $
  FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9, INDEX=I, $
```

EDUCFILE Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE EDUCFILE ON 01/05/96 AT 14.45.44

```

          COURSEG
01      S1
*****
* COURSE_CODE **
* COURSE_NAME **
*           **
*           **
*           **
*****
          I
          I
          I
          I ATTNDSEG
02      I SH2
*****
* DATE_ATTEND **
* EMP_ID      **I
*           **
*           **
*           **
*****
          *****
          *****
```

SALES Data Source

SALES contains sample data about a dairy company with an affiliated store chain. Its segments are:

STOR_SEG

Lists the stores buying the products.

DAT_SEG

Contains the dates of inventory.

PRODUCT

Contains sales data for each product on each date. The PROD_CODE field is indexed. The RETURNS and DAMAGED fields have the MISSING=ON attribute.

SALES Master File

```
FILENAME=KSALES, SUFFIX=FOC
SEGNAME=STOR_SEG, SEGTYPE=S1
  FIELDNAME=STORE_CODE, ALIAS=SNO, FORMAT=A3, $
  FIELDNAME=CITY, ALIAS=CTY, FORMAT=A15, $
  FIELDNAME=AREA, ALIAS=LOC, FORMAT=A1, $
SEGNAME=DATE_SEG, PARENT=STOR_SEG, SEGTYPE=SH1,
  FIELDNAME=PROD_CODE, ALIAS=PCODE, FORMAT=A3, FIELDTYPE=I, $
  FIELDNAME=UNIT_SOLD, ALIAS=SOLD, FORMAT=I5, $
  FIELDNAME=RETAIL_PRICE, ALIAS=RP, FORMAT=D5.2M, $
  FIELDNAME=DELIVER_AMT, ALIAS=SHIP, FORMAT=I5, $
  FIELDNAME=OPENING_AMT, ALIAS=INV, FORMAT=I5, $
  FIELDNAME=RETURNS, ALIAS=RTN, FORMAT=I3, MISSING=ON, $
  FIELDNAME=DAMAGED, ALIAS=BAD, FORMAT=I3, MISSING=ON, $
```

SALES Structure Diagram

SECTION 01

STRUCTURE OF FOCUS

FILE SALES ON 01/05/96 AT 14.50.28

```
          STOR_SEG
01      S1
*****
*STORE_CODE **
*CITY      **
*AREA     **
*         **
*         **
*****
          I
          I
          I
          I DATE_SEG
02      I SH1
*****
*DATE     **
*         **I
*         **
*         **
*         **
*****
          I
          I
          I
          I PRODUCT
03      I S1
*****
*PROD_CODE **I
*UNIT_SOLD **
*RETAIL_PRICE**
*DELIVER_AMT **
*         **
*****
          I
          I
```

CAR Data Source

CAR contains sample data about specifications and sales information for rare cars. Its segments are:

ORIGIN

Lists the country that manufactures the car. The field COUNTRY is indexed.

COMP

Contains the car name.

CARREC

Contains the car model.

BODY

Lists the body type, seats, dealer and retail costs, and units sold.

SPECS

Lists car specifications. This segment is unique.

WARANT

Lists the type of warranty.

EQUIP

Lists standard equipment.

The aliases in the CAR Master File are specified without the ALIAS keyword.

CAR Master File

```
FILENAME=CAR, SUFFIX=FOC
SEGNAME=ORIGIN, SEGTYPE=S1
  FIELDNAME=COUNTRY, COUNTRY, A10, FIELDTYPE=I, $
SEGNAME=COMP, SEGTYPE=S1, PARENT=ORIGIN
  FIELDNAME=CAR, CARS, A16, $
SEGNAME=CARREC, SEGTYPE=S1, PARENT=COMP
  FIELDNAME=MODEL, MODEL, A24, $
SEGNAME=BODY, SEGTYPE=S1, PARENT=CARREC
  FIELDNAME=BODYTYPE, TYPE, A12, $
  FIELDNAME=SEATS, SEAT, I3, $
  FIELDNAME=DEALER_COST, DCOST, D7, $
  FIELDNAME=RETAIL_COST, RCOST, D7, $
  FIELDNAME=SALES, UNITS, I6, $
SEGNAME=SPECS, SEGTYPE=U, PARENT=BODY
  FIELDNAME=LENGTH, LEN, D5, $
  FIELDNAME=WIDTH, WIDTH, D5, $
  FIELDNAME=HEIGHT, HEIGHT, D5, $
  FIELDNAME=WEIGHT, WEIGHT, D6, $
  FIELDNAME=WHEELBASE, BASE, D6.1, $
  FIELDNAME=FUEL_CAP, FUEL, D6.1, $
  FIELDNAME=BHP, POWER, D6, $
  FIELDNAME=RPM, RPM, I5, $
  FIELDNAME=MPG, MILES, D6, $
  FIELDNAME=ACCEL, SECONDS, D6, $
SEGNAME=WARRANT, SEGTYPE=S1, PARENT=COMP
  FIELDNAME=WARRANTY, WARR, A40, $
SEGNAME=EQUIP, SEGTYPE=S1, PARENT=COMP
  FIELDNAME=STANDARD, EQUIP, A40, $
```


LEDGER Data Source

LEDGER contains sample accounting data. It consists of one segment, TOP. This data source is specified primarily for FML examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

LEDGER Master File

```
FILENAME=LEDGER, SUFFIX=FOC,$
SEGNAME=TOP, SEGTYPE=S2,$
FIELDNAME=YEAR, , FORMAT=A4, $
FIELDNAME=ACCOUNT, , FORMAT=A4, $
FIELDNAME=AMOUNT, , FORMAT=I5C,$
```

LEDGER Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE LEDGER ON 01/05/96 AT 15.17.08

```
TOP
01 S2
*****
*YEAR **
*ACCOUNT **
*AMOUNT **
* **
* **
*****
*****
```

FINANCE Data Source

FINANCE contains sample financial data for balance sheets. It consists of one segment, TOP. This data source is specified primarily for FML examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

FINANCE Master File

```
FILENAME=FINANCE, SUFFIX=FOC,$
SEGNAME=TOP, SEGTYPE=S2,$
FIELDNAME=YEAR , , FORMAT=A4, $
FIELDNAME=ACCOUNT, , FORMAT=A4, $
FIELDNAME=AMOUNT , , FORMAT=D12C,$
```

FINANCE Structure Diagram

```
SECTION 01
                STRUCTURE OF FOCUS      FILE FINANCE  ON 01/05/96 AT 15.17.08
                TOP
01              S2
*****
*YEAR          **
*ACCOUNT       **
*AMOUNT        **
*              **
*              **
*****
*****
```

REGION Data Source

REGION contains sample account data for the east and west regions of the country. It consists of one segment, TOP. This data source is specified primarily for FML examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

REGION Master File

```
FILENAME=REGION, SUFFIX=FOC,$
SEGNAME=TOP,      SEGTYPE=S1,$
  FIELDNAME=ACCOUNT, , FORMAT=A4, $
  FIELDNAME=E_ACTUAL, , FORMAT=I5C,$
  FIELDNAME=E_BUDGET, , FORMAT=I5C,$
  FIELDNAME=W_ACTUAL, , FORMAT=I5C,$
  FIELDNAME=W_BUDGET, , FORMAT=I5C,$
```

REGION Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE REGION ON 01/05/96 AT 15.18.48

```

          TOP
01       S1
*****
*ACCOUNT      **
*E_ACTUAL     **
*E_BUDGET     **
*W_ACTUAL     **
*             **
*****
*****
```

COURSE Data Source

COURSE contains sample data about education courses. It consists of one segment, CRSELIST.

COURSE Master File

```
FILENAME=COURSE,    SUFFIX=FOC
SEGNAME=CRSELIST,  SEGTYPE=S1
  FIELDNAME=COURSECODE,  ALIAS=CCOD,    FORMAT=A7,    INDEX=I,    $
  FIELDNAME=CTITLE,      ALIAS=COURSE,  FORMAT=A35,   $
  FIELDNAME=SOURCE,      ALIAS=ORG,     FORMAT=A35,   $
  FIELDNAME=CLASSIF,     ALIAS=CLASS,  FORMAT=A10,   $
  FIELDNAME=TUITION,     ALIAS=FEE,    FORMAT=D8.2,  MISSING=ON, $
  FIELDNAME=DURATION,    ALIAS=DAYS,   FORMAT=A3,    MISSING=ON, $
  FIELDNAME=DESCRIPTN1,  ALIAS=DESC1,  FORMAT=A40,   $
  FIELDNAME=DESCRIPTN2,  ALIAS=DESC2,  FORMAT=A40,   $
  FIELDNAME=DESCRIPTN2,  ALIAS=DESC3,  FORMAT=A40,   $
```

COURSE Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE COURSE ON 05/21/99 AT 12.26.05

```
          CRSELIST
01      S1
*****
*COURSECODE  **I
*CTITLE      **
*SOURCE      **
*CLASSIF     **
*            **
*****
*****
```

EMPDATA Data Source

EMPDATA contains sample data about a company's employees. It consists of one segment, EMPDATA. The PIN field is indexed. The AREA field is a temporary field.

EMPDATA Master File

```

FILENAME=EMPDATA, SUFFIX=FOC
SEGNAME=EMPDATA, SEGTYPE=S1
  FIELDNAME=PIN,           ALIAS=ID,           FORMAT=A9,   INDEX=I,   $
  FIELDNAME=LASTNAME,     ALIAS=LN,           FORMAT=A15,   $
  FIELDNAME=FIRSTNAME,   ALIAS=FN,           FORMAT=A10,   $
  FIELDNAME=MIDINITIAL,  ALIAS=MI,           FORMAT=A1,    $
  FIELDNAME=DIV,          ALIAS=CDIV,         FORMAT=A4,    $
  FIELDNAME=DEPT,         ALIAS=CDEPT,        FORMAT=A20,   $
  FIELDNAME=JOBCLASS,    ALIAS=CJCLAS,       FORMAT=A8,    $
  FIELDNAME=TITLE,       ALIAS=CFUNC,        FORMAT=A20,   $
  FIELDNAME=SALARY,      ALIAS=CSAL,         FORMAT=D12.2M, $
  FIELDNAME=HIREDATE,    ALIAS=HDAT,         FORMAT=YMD,   $
$
DEFINE AREA/A13=DECODE DIV (NE 'NORTH EASTERN' SE 'SOUTH EASTERN'
CE 'CENTRAL' WE 'WESTERN' CORP 'CORPORATE' ELSE 'INVALID AREA');$

```

EMPDATA Structure Diagram

```

SECTION 01
          STRUCTURE OF FOCUS      FILE EMPDATA ON 01/05/96 AT 14.49.09
          EMPDATA
01          S1
*****
*PIN          **I
*LASTNAME     **
*FIRSTNAME    **
*MIDINITIAL   **
*             **
*****
*****

```

TRAINING Data Source

TRAINING contains sample data about training courses for employees. It consists of one segment, TRAINING. The PIN field is indexed. The EXPENSES, GRADE, and LOCATION fields have the MISSING=ON attribute.

TRAINING Master File

```
FILENAME=TRAINING, SUFFIX=FOC
SEGNAME=TRAINING, SEGTYPE=SH3
  FIELDNAME=PIN,          ALIAS=ID,          FORMAT=A9,    INDEX=I,    $
  FIELDNAME=COURSESTART, ALIAS=CSTART, FORMAT=YMD,   $
  FIELDNAME=COURSECODE,  ALIAS=CCOD,  FORMAT=A7,    $
  FIELDNAME=EXPENSES,    ALIAS=COST,  FORMAT=D8.2,  MISSING=ON  $
  FIELDNAME=GRADE,       ALIAS=GRA,   FORMAT=A2,    MISSING=ON, $
  FIELDNAME=LOCATION,     ALIAS=LOC,   FORMAT=A6,    MISSING=ON, $
```

TRAINING Structure Diagram

```
SECTION 01
          STRUCTURE OF FOCUS      FILE TRAINING ON 12/12/94 AT 14.51.28
          TRAINING
          01      SH3
          *****
          *PIN          **I
          *COURSESTART **
          *COURSECODE  **
          *EXPENSES    **
          *              **
          *****
          *****
```

VideoTrk, MOVIES, and ITEMS Data Sources

VideoTrk contains sample data about customer, rental, and purchase information for a video rental business. It can be joined to the MOVIES or ITEMS data source. VideoTrk and MOVIES are used in examples that illustrate the use of the Maintain facility.

VideoTrk Master File

```

FILENAME=VIDEOTRK,  SUFFIX=FOC
SEGNAME=CUST,      SEGTYPE=S1
    FIELDNAME=CUSTID,    ALIAS=CIN,          FORMAT=A4,    $
    FIELDNAME=LASTNAME,  ALIAS=LN,          FORMAT=A15,   $
    FIELDNAME=FIRSTNAME, ALIAS=FN,          FORMAT=A10,   $
    FIELDNAME=EXPDATE,   ALIAS=EXDAT,       FORMAT=YMD,   $
    FIELDNAME=PHONE,     ALIAS=TEL,         FORMAT=A10,   $
    FIELDNAME=STREET,    ALIAS=STR,         FORMAT=A20,   $
    FIELDNAME=CITY,      ALIAS=CITY,        FORMAT=A20,   $
    FIELDNAME=STATE,     ALIAS=PROV,        FORMAT=A4,    $
    FIELDNAME=ZIP,       ALIAS=POSTAL_CODE, FORMAT=A9,    $
SEGNAME=TRANSDAT,  SEGTYPE=SH1,  PARENT=CUST
    FIELDNAME=TRANSDATE, ALIAS=OUTDATE,    FORMAT=YMD,   $
SEGNAME=SALES,     SEGTYPE=S2,    PARENT=TRANSDAT
    FIELDNAME=PRODCODE,  ALIAS=PCOD,       FORMAT=A6,    $
    FIELDNAME=TRANSCODE, ALIAS=TCOD,       FORMAT=I3,    $
    FIELDNAME=QUANTITY,  ALIAS=NO,         FORMAT=I3S,   $
    FIELDNAME=TRANSTOT,  ALIAS=TTOT,       FORMAT=F7.2S, $
SEGNAME=RENTALS,  SEGTYPE=S2,    PARENT=TRANSDAT
    FIELDNAME=MOVIECODE, ALIAS=MCOD,       FORMAT=A6,    INDEX=I, $
    FIELDNAME=COPY,      ALIAS=COPY,       FORMAT=I2,    $
    FIELDNAME=RETURNDATE, ALIAS=INDATE,     FORMAT=YMD,   $
    FIELDNAME=FEE,       ALIAS=FEE,        FORMAT=F5.2S, $

```

VideoTrk Structure Diagram

SECTION 01

STRUCTURE OF FOCUS

FILE VIDEOTRK ON 05/21/99 AT 12.25.19

```

          CUST
01      S1
*****
*CUSTID      **
*LASTNAME    **
*FIRSTNAME   **
*EXPDATE     **
*            **
*****
          I
          I
          I
          I TRANSDAT
02      I SH1
*****
*TRANSDATE   **
*            **
*            **
*            **
*            **
*****
          I
          +-----+
          I            I
          I SALES      I RENTALS
03      I S2          04      I S2
*****          *****
*PRODCODE    **      *MOVIECODE  **I
*TRANSCODE   **      *COPY        **
*QUANTITY    **      *RETURNDATE **
*TRANSTOT    **      *FEE         **
*            **      *            **
*****          *****
          *****          *****

```

MOVIES Master File

```

FILENAME=MOVIES,      SUFFIX=FOC
SEGNAME=MOVINFO,     SEGTYPE=S1
  FIELDNAME=MOVIECODE, ALIAS=MCOD,  FORMAT=A6, INDEX=I, $
  FIELDNAME=TITLE,     ALIAS=MTL,   FORMAT=A39, $
  FIELDNAME=CATEGORY,  ALIAS=CLASS, FORMAT=A8, $
  FIELDNAME=DIRECTOR,  ALIAS=DIR,   FORMAT=A17, $
  FIELDNAME=RATING,    ALIAS=RTG,   FORMAT=A4, $
  FIELDNAME=RELDATE,   ALIAS=RDAT,  FORMAT=YMD, $
  FIELDNAME=WHOLESALEPR, ALIAS=WPRC,  FORMAT=F6.2, $
  FIELDNAME=LISTPR,    ALIAS=LPRC,  FORMAT=F6.2, $
  FIELDNAME=COPIES,    ALIAS=NOC,   FORMAT=I3, $

```

MOVIES Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE MOVIES ON 05/21/99 AT 12.26.05

```

          MOVINFO
01      S1
*****
*MOVIECODE   **I
*TITLE       **
*CATEGORY    **
*DIRECTOR    **
*            **
*****
*****

```

ITEMS Master File

```
FILENAME=ITEMS,    SUFFIX=FOC
SEGNAME=ITMINFO,  SEGTYPE=S1
  FIELDNAME=PRODCODE, ALIAS=PCOD,  FORMAT=A6,  INDEX=I,  $
  FIELDNAME=PRODNAME, ALIAS=PROD,  FORMAT=A20,      $
  FIELDNAME=OURCOST,  ALIAS=WCost,  FORMAT=F6.2,      $
  FIELDNAME=RETAILPR, ALIAS=PRICE,  FORMAT=F6.2,      $
  FIELDNAME=ON_HAND,  ALIAS=NUM,    FORMAT=I5,        $
```

ITEMS Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE ITEMS ON 05/21/99 AT 12.26.05

```
          ITMINFO
01      S1
*****
*PRODCODE  **I
*PRODNAME  **
*OURCOST   **
*RETAILPR  **
*          **
*****
*****
```

VIDEOTR2 Data Source

VIDEOTR2 contains sample data about customer, rental, and purchase information for a video rental business. It consists of four segments.

VIDEOTR2 Master Files

```

FILENAME=VIDEOTR2,  SUFFIX=FOC
SEGNAME=CUST,      SEGTYPE=S1
  FIELDNAME=CUSTID,  ALIAS=CIN,          FORMAT=A4,          $
  FIELDNAME=LASTNAME, ALIAS=LN,          FORMAT=A15,         $
  FIELDNAME=FIRSTNAME, ALIAS=FN,          FORMAT=A10,         $
  FIELDNAME=EXPDATE,  ALIAS=EXDAT,        FORMAT=YMD,         $
  FIELDNAME=PHONE,   ALIAS=TEL,          FORMAT=A10,         $
  FIELDNAME=STREET,  ALIAS=STR,          FORMAT=A20,         $
  FIELDNAME=CITY,    ALIAS=CITY,          FORMAT=A20,         $
  FIELDNAME=STATE,   ALIAS=PROV,          FORMAT=A4,          $
  FIELDNAME=ZIP,     ALIAS=POSTAL_CODE,   FORMAT=A9,          $
SEGNAME=TRANSDAT,  SEGTYPE=SH1,  PARENT=CUST
  FIELDNAME=TRANSDATE, ALIAS=OUTDATE,    FORMAT=HYMDI,      $
SEGNAME=SALES,     SEGTYPE=S2,  PARENT=TRANSDAT
  FIELDNAME=TRANSCODE, ALIAS=TCOD,        FORMAT=I3,         $
  FIELDNAME=QUANTITY,  ALIAS=NO,          FORMAT=I3S,        $
  FIELDNAME=TRANSTOT,  ALIAS=TTOT,        FORMAT=F7.2S,      $
SEGNAME=RENTALS,  SEGTYPE=S2,  PARENT=TRANSDAT
  FIELDNAME=MOVIECODE, ALIAS=MCOD,        FORMAT=A6,  INDEX=I, $
  FIELDNAME=COPY,      ALIAS=COPY,        FORMAT=I2,         $
  FIELDNAME=RETURNDATE, ALIAS=INDATE,     FORMAT=YMD,         $
  FIELDNAME=FEE,       ALIAS=FEE,          FORMAT=F5.2S,      $

```

VIDEOTR2 Structure Diagram

SECTION 01

STRUCTURE OF FOCUS

FILE VIDEOTR2 ON 09/27/00 AT 16.45.48

```
          CUST
01      S1
*****
*CUSTID      **
*LASTNAME    **
*FIRSTNAME   **
*EXPDATE     **
*            **
*****
          I
          I
          I
          I  TRANSDAT
02      I SH1
*****
*TRANSDATE   **
*            **
*            **
*            **
*            **
*****
          I
          +-----+
          I              I
          I SALES        I RENTALS
03      I S2            04      I S2
*****                *****
*TRANSCODE   **      *MOVIECODE   ** I
*QUANTITY    **      *COPY         **
*TRANSTOT    **      *RETURNDATE  **
*            **      *FEE          **
*            **      *            **
*****                *****
*            **      *            **
```

Gotham Grinds Data Sources

Gotham Grinds is a group of data sources that contain sample data about a specialty items company.

- GGDEMOG contains demographic information about the customers of Gotham Grinds, a company that sells specialty items like coffee, gourmet snacks, and gifts. It consists of one segment, DEMOG01.
- GGORDER contains order information for Gotham Grinds. It consists of two segments, ORDER01 and ORDER02.
- GGPRODS contains product information for Gotham Grinds. It consists of one segment, PRODS01.
- GGSALES contains sales information for Gotham Grinds. It consists of one segment, SALES01.
- GGSTORES contains information for each of Gotham Grinds' 12 stores in the United States. It consists of one segment, STORES01.

GGDEMOG Master File

```

FILENAME=GGDEMOG, SUFFIX=FOC
SEGNAME=DEMOG01, SEGTYPE=S1
  FIELD=ST, ALIAS=E02, FORMAT=A02, INDEX=I, TITLE='State',
  DESC='State', $
  FIELD=HH, ALIAS=E03, FORMAT=I09, TITLE='Number of Households',
  DESC='Number of Households', $
  FIELD=AVGHHSZ98, ALIAS=E04, FORMAT=I09, TITLE='Average Household Size',
  DESC='Average Household Size', $
  FIELD=MEDHHI98, ALIAS=E05, FORMAT=I09, TITLE='Median Household Income',
  DESC='Median Household Income', $
  FIELD=AVGHHI98, ALIAS=E06, FORMAT=I09, TITLE='Average Household Income',
  DESC='Average Household Income', $
  FIELD=MALEPOP98, ALIAS=E07, FORMAT=I09, TITLE='Male Population',
  DESC='Male Population', $
  FIELD=FEMPOP98, ALIAS=E08, FORMAT=I09, TITLE='Female Population',
  DESC='Female Population', $
  FIELD=P15TO1998, ALIAS=E09, FORMAT=I09, TITLE='15 to 19',
  DESC='Population 15 to 19 years old', $
  FIELD=P20TO2998, ALIAS=E10, FORMAT=I09, TITLE='20 to 29',
  DESC='Population 20 to 29 years old', $
  FIELD=P30TO4998, ALIAS=E11, FORMAT=I09, TITLE='30 to 49',
  DESC='Population 30 to 49 years old', $
  FIELD=P50TO6498, ALIAS=E12, FORMAT=I09, TITLE='50 to 64',
  DESC='Population 50 to 64 years old', $
  FIELD=P65OVR98, ALIAS=E13, FORMAT=I09, TITLE='65 and over',
  DESC='Population 65 and over', $

```

GGDEMOG Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE GGDEMOG ON 05/21/99 AT 12.26.05

```
          GGDEMOG
01      S1
*****
*ST          **I
*HH          **
*AVGHHSZ98   **
*MEDHHI98    **
*            **
*****
*****
```

GGORDER Master File

```
FILENAME=GGORDER, SUFFIX=FOC,$
SEGNAME=ORDER01, SEGTYPE=S1,$
  FIELD=ORDER_NUMBER, ALIAS=ORDNO1, FORMAT=I6, TITLE='Order,Number',
  DESC='Order Identification Number',$
  FIELD=ORDER_DATE, ALIAS=DATE, FORMAT=MDY, TITLE='Order,Date',
  DESC='Date order was placed',$
  FIELD=STORE_CODE, ALIAS=STCD, FORMAT=A5, TITLE='Store,Code',
  DESC='Store Identification Code (for order)',$
  FIELD=PRODUCT_CODE, ALIAS=PCD, FORMAT=A4, TITLE='Product,Code',
  DESC='Product Identification Code (for order)',$
  FIELD=QUANTITY, ALIAS=ORDUNITS, FORMAT=I8, TITLE='Ordered,Units',
  DESC='Quantity Ordered',$
SEGNAME=ORDER02, SEGTYPE=KU, PARENT=ORDER01, CRFILE=GGPRODS, CRKEY=PCD,
CRSEG=PRODS01 ,,$
```

GGORDER Structure Diagram

SECTION 01

STRUCTURE OF FOCUS

FILE GGORDER ON 09/27/00 AT 16.45.48

```

          GGORDER
01          S1
*****
*ORDER_NUMBER**
*ORDER_DATE   **
*STORE_CODE   **
*PRODUCT_CODE**
*              **
*****
          I
          I
          I
          I ORDER02
02          I KU
.....
:PRODUCT_ID   :K
:PRODUCT_DESC:
:VENDOR_CODE  :
:VENDOR_NAME  :
:              :
:.....:

```

GGPRODS Master File

```

FILENAME=GGPRODS, SUFFIX=FOC
SEGNAME=PRODS01, SEGTYPE=S1
  FIELD=PRODUCT_ID, ALIAS=PCD, FORMAT=A4, INDEX=I, TITLE='Product,Code',
  DESC='Product Identification Code', $
  FIELD=PRODUCT_DESCRIPTION, ALIAS=PRODUCT, FORMAT=A16, TITLE='Product',
  DESC='Product Name', $
  FIELD=VENDOR_CODE, ALIAS=VCD, FORMAT=A4, INDEX=I, TITLE='Vendor ID',
  DESC='Vendor Identification Code', $
  FIELD=VENDOR_NAME, ALIAS=VENDOR, FORMAT=A23, TITLE='Vendor Name',
  DESC='Vendor Name', $
  FIELD=PACKAGE_TYPE, ALIAS=PACK, FORMAT=A7, TITLE='Package',
  DESC='Packaging Style', $
  FIELD=SIZE, ALIAS=SZ, FORMAT=I2, TITLE='Size',
  DESC='Package Size', $
  FIELD=UNIT_PRICE, ALIAS=UNITPR, FORMAT=D7.2, TITLE='Unit,Price',
  DESC='Price for one unit', $

```

GGPRODS Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE GGPRODS ON 05/21/99 AT 12.26.05

```
GGPRODS
01      S1
*****
*PRODUCT_ID  **I
*VENDOR_CODE **I
*PRODUCT_DESC**
*VENDOR_NAME **
*           **
*****
*****
```

GGSALES Master File

```
FILENAME=GGSALES, SUFFIX=FOC
SEGNAME=SALES01, SEGTYPE=S1
FIELD=SEQ_NO, ALIAS=SEQ, FORMAT=I5, TITLE='Sequence#',
DESC='Sequence number in database', $
FIELD=CATEGORY, ALIAS=E02, FORMAT=A11, INDEX=I, TITLE='Category',
DESC='Product category', $
FIELD=PCD, ALIAS=E03, FORMAT=A04, INDEX=I, TITLE='Product ID',
DESC='Product Identification code (for sale)', $
FIELD=PRODUCT, ALIAS=E04, FORMAT=A16, TITLE='Product',
DESC='Product name', $
FIELD=REGION, ALIAS=E05, FORMAT=A11, INDEX=I, TITLE='Region',
DESC='Region code', $
FIELD=ST, ALIAS=E06, FORMAT=A02, INDEX=I, TITLE='State',
DESC='State', $
FIELD=CITY, ALIAS=E07, FORMAT=A20, TITLE='City',
DESC='City', $
FIELD=STCD, ALIAS=E08, FORMAT=A05, INDEX=I, TITLE='Store ID',
DESC='Store identification code (for sale)', $
FIELD=DATE, ALIAS=E09, FORMAT=I8YYMD, TITLE='Date',
DESC='Date of sales report', $
FIELD=UNITS, ALIAS=E10, FORMAT=I08, TITLE='Unit Sales',
DESC='Number of units sold', $
FIELD=DOLLARS, ALIAS=E11, FORMAT=I08, TITLE='Dollar Sales',
DESC='Total dollar amount of reported sales', $
FIELD=BUDUNITS, ALIAS=E12, FORMAT=I08, TITLE='Budget Units',
DESC='Number of units budgeted', $
FIELD=BUDDOLLARS, ALIAS=E13, FORMAT=I08, TITLE='Budget Dollars',
DESC='Total sales quota in dollars', $
```

GGSALES Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE GGSALES ON 05/21/99 AT 12.26.05

```

GGSALES
01      S1
*****
*SEQ_NO      **
*CATEGORY    **I
*PCD         **I
*REGION      **I
*           **
*****
*****

```

GGSTORES Master File

```

FILENAME=GGSTORES, SUFFIX=FOC
SEGNAME=STORES01, SEGTYPE=S1
FIELD=STORE_CODE, ALIAS=E02, FORMAT=A05, INDEX=I, TITLE='Store ID',
DESC='Franchisee ID Code', $
FIELD=STORE_NAME, ALIAS=E03, FORMAT=A23, TITLE='Store Name',
DESC='Store Name', $
FIELD=ADDRESS1, ALIAS=E04, FORMAT=A19, TITLE='Contact',
DESC='Franchisee Owner', $
FIELD=ADDRESS2, ALIAS=E05, FORMAT=A31, TITLE='Address',
DESC='Street Address', $
FIELD=CITY, ALIAS=E06, FORMAT=A22, TITLE='City',
DESC='City', $
FIELD=STATE, ALIAS=E07, FORMAT=A02, INDEX=I, TITLE='State',
DESC='State', $
FIELD=ZIP, ALIAS=E08, FORMAT=A06, TITLE='Zip Code',
DESC='Postal Code', $

```

GGSTORES Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE GGSTORES ON 05/21/99 AT 12.26.05

```

GGSTORES
01      S1
*****
*STORE_CODE  **I
*STATE       **I
*STORE_NAME  **
*ADDRESS1    **
*           **
*****
*****

```

Century Corp Data Sources

Century Corp is a consumer electronics manufacturer that distributes products through retailers around the world. Century Corp has thousands of employees in plants, warehouses, and offices worldwide. Their mission is to provide quality products and services to their customers.

Century Corp is a group of data sources that contain financial, human resources, inventory, and order information.

- CENTCOMP contains location information for stores. It consists of one segment, COMPINFO.
- CENTFIN contains financial information. It consists of one segment, ROOT_SEG.
- CENTHR contains human resources information. It consists of one segment, EMPSEG.
- CENTINV contains inventory information. It consists of one segment, INVINFO.
- CENTORD contains order information. It consists of four segments, OINFO, STOSEG, PINFO, and INVSEG.
- CENTQA contains problem information. It consists of three segments, PROD_SEG, INVSEG, and PROB_SEG.

CENTCOMP Data Source

```

FILE=CENTCOMP, SUFFIX=FOC, FDFC=19, FYRT=00
SEGNAME=COMPINFO, SEGTYPE=S1, $
  FIELD=STORE_CODE, ALIAS=SNUM, FORMAT=A6, INDEX=I,
  TITLE='Store Id#:',
  DESCRIPTION='Store Id#', $
  FIELD=STORENAME, ALIAS=SNAME, FORMAT=A20,
  WITHIN=STATE,
  TITLE='Store Name:',
  DESCRIPTION='Store Name', $
  FIELD=STATE, ALIAS=STATE, FORMAT=A2,
  WITHIN=PLANT,
  TITLE='State:',
  DESCRIPTION=State, $
DEFINE REGION/A5=DECODE STATE ('AL' 'SOUTH' 'AK' 'WEST' 'AR' 'SOUTH'
'AZ' 'WEST' 'CA' 'WEST' 'CO' 'WEST' 'CT' 'EAST'
'DE' 'EAST' 'DC' 'EAST' 'FL' 'SOUTH' 'GA' 'SOUTH' 'HI' 'WEST'
'ID' 'WEST' 'IL' 'NORTH' 'IN' 'NORTH' 'IA' 'NORTH'
'KS' 'NORTH' 'KY' 'SOUTH' 'LA' 'SOUTH' 'ME' 'EAST' 'MD' 'EAST'
'MA' 'EAST' 'MI' 'NORTH' 'MN' 'NORTH' 'MS' 'SOUTH' 'MT' 'WEST'
'MO' 'SOUTH' 'NE' 'WEST' 'NV' 'WEST' 'NH' 'EAST' 'NJ' 'EAST'
'NM' 'WEST' 'NY' 'EAST' 'NC' 'SOUTH' 'ND' 'NORTH' 'OH' 'NORTH'
'OK' 'SOUTH' 'OR' 'WEST' 'PA' 'EAST' 'RI' 'EAST' 'SC' 'SOUTH'
'SD' 'NORTH' 'TN' 'SOUTH' 'TX' 'SOUTH' 'UT' 'WEST' 'VT' 'EAST'
'VA' 'SOUTH' 'WA' 'WEST' 'WV' 'SOUTH' 'WI' 'NORTH' 'WY' 'WEST'
'NA' 'NORTH' 'ON' 'NORTH' ELSE ' ');
TITLE='Region:',
DESCRIPTION=Region, $

```

CENTCOMP Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE CENTCOMP ON 07/30/01 AT 10.20.49

```

          COMPINFO
01      S1
*****
*STORE_CODE  **I
*STORENAME   **
*STATE       **
*            **
*            **
*****
*****

```

CENTFIN Data Source

```
FILE=CENTFIN, SUFFIX=FOC, FDFC=19, FYRT=00
SEGNAME=ROOT_SEG, SEGTYPE=S4, $
  FIELD=YEAR, ALIAS=YEAR,  FORMAT=YY,
  WITHIN='*Time Period', $
  FIELD=QUARTER, ALIAS=QTR, FORMAT=Q,
  WITHIN=YEAR,
  TITLE=Quarter,
  DESCRIPTION=Quarter, $
  FIELD=MONTH, ALIAS=MONTH, FORMAT=M,
  TITLE=Month,
  DESCRIPTION=Month, $
  FIELD=ITEM, ALIAS=ITEM,  FORMAT=A20,
  TITLE=Item,
  DESCRIPTION=Item, $
  FIELD=VALUE, ALIAS=VALUE, FORMAT=D12.2,
  TITLE=Value,
  DESCRIPTION=Value, $

DEFINE ITYPE/A12=IF EDIT(ITEM,'9$$$$$$$$$$$$$$$$') EQ 'E'
  THEN 'Expense' ELSE IF EDIT(ITEM,'9$$$$$$$$$$$$$$$$') EQ 'R'
  THEN 'Revenue' ELSE 'Asset';,
  TITLE=Type,
  DESCRIPTION='Type of Financial Line Item', $

DEFINE MOTEXT/MT=MONTH;,$
```

CENTFIN Structure Diagram

```
SECTION 01

                STRUCTURE OF FOCUS      FILE CENTFIN  ON 07/30/01 AT 10.25.52

                ROOT_SEG
01              S4
*****
*YEAR          **
*QUARTER       **
*MONTH         **
*ITEM          **
*              **
*****
*****
```

CENTHR Data Source

```

FILE=CENTHR, SUFFIX=FOC,
SEGNAME=EMPSEG, SEGTYPE=S1, $
  FIELD=ID_NUM, ALIAS=ID#, FORMAT=I9,
    TITLE='Employee, ID#',
    DESCRIPTION='Employee Identification Number', $
  FIELD=LNAME, ALIAS=LN,  FORMAT=A14,
    TITLE='Last, Name',
    DESCRIPTION='Employee Last Name', $
  FIELD=FNAME, ALIAS=FN,  FORMAT=A12,
    TITLE='First, Name',
    DESCRIPTION='Employee First Name', $
  FIELD=PLANT, ALIAS=PLT,  FORMAT=A3,
    TITLE='Plant, Location',
    DESCRIPTION='Location of the manufacturing plant',
    WITHIN='*Location', $
  FIELD=START_DATE, ALIAS=SDATE, FORMAT=YYMD,
    TITLE='Starting, Date',
    DESCRIPTION='Date of employment', $
  FIELD=TERM_DATE, ALIAS=TERM_DATE, FORMAT=YYMD,
    TITLE='Termination, Date',
    DESCRIPTION='Termination Date', $
  FIELD=STATUS, ALIAS=STATUS, FORMAT=A10,
    TITLE='Current, Status',
    DESCRIPTION='Job Status', $
  FIELD=POSITION, ALIAS=JOB, FORMAT=A2,
    TITLE=Position,
    DESCRIPTION='Job Position', $
  FIELD=PAYSCALE, ALIAS=PAYLEVEL, FORMAT=I2,
    TITLE='Pay, Level',
    DESCRIPTION='Pay Level',
    WITHIN='*Wages', $
  DEFINE POSITION_DESC/A17=IF POSITION EQ 'BM' THEN
    'Plant Manager' ELSE
    IF POSITION EQ 'MR' THEN 'Line Worker' ELSE
    IF POSITION EQ 'TM' THEN 'Line Manager' ELSE
    'Technician';
    TITLE='Position, Description',
    DESCRIPTION='Position Description',
    WITHIN='PLANT', $
  DEFINE BYEAR/YY=START_DATE;
    TITLE='Beginning, Year',
    DESCRIPTION='Beginning Year',
    WITHIN='*Starting Time Period', $
  DEFINE BQUARTER/Q=START_DATE;
    TITLE='Beginning, Quarter',
    DESCRIPTION='Beginning Quarter',
    WITHIN='BYEAR',
  DEFINE BMONTH/M=START_DATE;
    TITLE='Beginning, Month',
    DESCRIPTION='Beginning Month',
    WITHIN='BQUARTER', $

```

```
DEFINE EYEAR/YY=TERM_DATE;
  TITLE='Ending,Year',
  DESCRIPTION='Ending Year',
  WITHIN='*Termination Time Period', $
DEFINE EQUARTER/Q=TERM_DATE;
  TITLE='Ending,Quarter',
  DESCRIPTION='Ending Quarter',
  WITHIN='EYEAR', $
DEFINE EMONTH/M=TERM_DATE;
  TITLE='Ending,Month',
  DESCRIPTION='Ending Month',
  WITHIN='EQUARTER', $
DEFINE RESIGN_COUNT/I3=IF STATUS EQ 'RESIGNED' THEN 1
  ELSE 0;
  TITLE='Resigned,Count',
  DESCRIPTION='Resigned Count', $
DEFINE FIRE_COUNT/I3=IF STATUS EQ 'TERMINAT' THEN 1
  ELSE 0;
  TITLE='Terminated,Count',
  DESCRIPTION='Terminated Count', $
DEFINE DECLINE_COUNT/I3=IF STATUS EQ 'DECLINED' THEN 1
  ELSE 0;
  TITLE='Declined,Count',
  DESCRIPTION='Declined Count', $
DEFINE EMP_COUNT/I3=IF STATUS EQ 'EMPLOYED' THEN 1
  ELSE 0;
  TITLE='Employed,Count',
  DESCRIPTION='Employed Count', $
DEFINE PEND_COUNT/I3=IF STATUS EQ 'PENDING' THEN 1
  ELSE 0;
  TITLE='Pending,Count',
  DESCRIPTION='Pending Count', $
DEFINE REJECT_COUNT/I3=IF STATUS EQ 'REJECTED' THEN 1
  ELSE 0;
  TITLE='Rejected,Count',
  DESCRIPTION='Rejected Count', $
DEFINE FULLNAME/A28=LNAME||', '|FNAME;
  TITLE='Full Name',
  DESCRIPTION='Full Name: Last, First', WITHIN='POSITION_DESC', $
DEFINE SALARY/D12.2=IF BMONTH LT 4 THEN PAYLEVEL * 12321
  ELSE IF BMONTH GE 4 AND BMONTH LT 8 THEN PAYLEVEL * 13827
  ELSE PAYLEVEL * 14400;,
  TITLE='Salary',
  DESCRIPTION='Salary', $
DEFINE PLANTLNG/All=DECODE PLANT (BOS 'Boston' DAL 'Dallas'
  LA 'Los Angeles' ORL 'Orlando' SEA 'Seattle' STL 'St Louis'
  ELSE 'n/a'); $
```

CENTHR Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE CENTHR ON 07/30/01 AT 10.40.34

```

EMPSEG
01      S1
*****
*ID_NUM      **
*LNAME       **
*FNAME       **
*PLANT       **
*            **
*****
*****

```

CENTINV Data Source

```

FILE=CENTINV, SUFFIX=FOC, FDFC=19, FYRT=00
SEGNAME=INVINFO, SEGTYPE=S1, $
FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
  TITLE='Product Number:',
  DESCRIPTION='Product Number', $
FIELD=PRODNAME, ALIAS=PNAME, FORMAT=A30,
  WITHIN=PRODCAT,
  TITLE='Product Name:',
  DESCRIPTION='Product Name', $
FIELD=QTY_IN_STOCK, ALIAS=QIS, FORMAT=I7,
  TITLE='Quantity In Stock:',
  DESCRIPTION='Quantity In Stock', $
FIELD=PRICE, ALIAS=RETAIL, FORMAT=D12.2,
  TITLE='Price:',
  DESCRIPTION=Price, $
FIELD=COST, ALIAS=OUR_COST, FORMAT=D12.2,
  TITLE='Our Cost:',
  DESCRIPTION='Our Cost:', $

DEFINE PRODCAT/A22 = IF PRODNAME CONTAINS 'LCD'
THEN 'VCRs' ELSE IF PRODNAME
CONTAINS 'DVD' THEN 'DVD' ELSE IF PRODNAME CONTAINS 'Camcor'
THEN 'Camcorders'
ELSE IF PRODNAME CONTAINS 'Camera' THEN 'Cameras' ELSE IF PRODNAME
CONTAINS 'CD' THEN 'CD Players'
ELSE IF PRODNAME CONTAINS 'Tape' THEN 'Digital Tape Recorders'
ELSE IF PRODNAME CONTAINS 'Combo' THEN 'Combo Players'
ELSE 'PDA Devices'; WITHIN=PRODTYPE, TITLE='Product Category:', $

DEFINE PRODTYPE/A19 = IF PRODNAME CONTAINS 'Digital' OR 'DVD' OR 'QX'
THEN 'Digital' ELSE 'Analog'; WITHIN='*Product Dimension',
  TITLE='Product Type:', $

```

CENTINV Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE CENTINV ON 07/30/01 AT 10.43.35

```

          INVINFO
01          S1
*****
*PROD_NUM   **I
*PRODNAME   **
*QTY_IN_STOCK**
*PRICE      **
*           **
*****
*****
    
```

CENTORD Data Source

```

FILE=CENTORD, SUFFIX=FOC,
SEGNAME=OINFO, SEGTYPE=S1, $
  FIELD=ORDER_NUM, ALIAS=ONUM, FORMAT=A5, INDEX=I,
  TITLE='Order Number:',
  DESCRIPTION='Order,Number', $
  FIELD=ORDER_DATE, ALIAS=ODATE, FORMAT=YYMD,
  TITLE='Date Of Order:',
  DESCRIPTION='Date,Of Order', $
  FIELD=STORE_CODE, ALIAS=SNUM, FORMAT=A6, INDEX=I,
  TITLE='Company ID#:',
  DESCRIPTION='Company ID#', $
  FIELD=PLANT, ALIAS=PLNT, FORMAT=A3, INDEX=I,
  TITLE='Manufacturing Plant',
  DESCRIPTION='Location Of Manufacturing Plant',
  WITHIN='*Location', $
  DEFINE YEAR/YY=ORDER_DATE;
  WITHIN='*Time Period', $
  DEFINE QUARTER/Q=ORDER_DATE;
  WITHIN='YEAR', $
  DEFINE MONTH/M=ORDER_DATE;
  WITHIN='QUARTER', $
SEGNAME=PINFO, SEGTYPE=S1, PARENT=OINFO, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
  TITLE='Product Number#:',
  DESCRIPTION='Product Number#', $
  FIELD=QUANTITY, ALIAS=QTY, FORMAT=I8C,
  TITLE='Quantity:',
  DESCRIPTION=Quantity, $
  FIELD=LINEPRICE, ALIAS=LINETOTAL, FORMAT=D12.2MC,
  TITLE='Line Total',
  DESCRIPTION='Line Total', $
  DEFINE LINE_COGS/D12.2=QUANTITY*COST;
  TITLE='Line Cost Of Goods Sold',
  DESCRIPTION='Line cost of goods sold', $
  DEFINE PLANTLNG/All=DECODE PLANT (BOS 'Boston' DAL 'Dallas'
  LA 'Los Angeles' ORL 'Orlando' SEA 'Seattle' STL 'St Louis'
  ELSE 'n/a');
SEGNAME=INVSEG, SEGTYPE=DKU, PARENT=PINFO, CRFILE=CENTINV,
CRKEY=PROD_NUM, CRSEG=INVINFO, $
SEGNAME=STOSEG, SEGTYPE=DKU, PARENT=OINFO, CRFILE=CENTCOMP,
CRKEY=STORE_CODE, CRSEG=COMPINFO, $
    
```

CENTORD Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE CENTORD ON 07/30/01 AT 10.17.52

```

OINFO
01      S1
*****
*ORDER_NUM  **I
*STORE_CODE **I
*PLANT      **I
*ORDER_DATE **
*
*****
      I
      +-----+
      I             I
      I STOSEG      I PINFO
02      I KU        03      I S1
.....
:STORE_CODE :K      *PROD_NUM  **I
:STORENAME  :      *QUANTITY  **
:STATE      :      *LINEPRICE **
:           :      *           **
:           :      *           **
:.....:      *****
JOINED  CENTCOMP *****
      I
      I
      I
      I INVSEG
      04      I KU
.....
:PROD_NUM   :K
:PRODNAME   :
:QTY_IN_STOCK:
:PRICE      :
:           :
:.....:
      JOINED  CENTINV

```

CENTQA Data Source

```
FILE=CENTQA, SUFFIX=FOC, FDFC=19, FYRT=00
SEGNAME=PROD_SEG, SEGTYPE=S1, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
  TITLE='Product,Number',
  DESCRIPTION='Product Number', $
SEGNAME=PROB_SEG, PARENT=PROD_SEG, SEGTYPE=S1, $
  FIELD=PROBNUM, ALIAS=PROBNO, FORMAT=I5,
  TITLE='Problem,Number',
  DESCRIPTION='Problem Number',
  WITHIN=PLANT,$
  FIELD=PLANT, ALIAS=PLT, FORMAT=A3, INDEX=I,
  TITLE=Plant,
  DESCRIPTION=Plant,
  WITHIN=PROBLEM_LOCATION,$
  FIELD=PROBLEM_DATE, ALIAS=PDATE, FORMAT=YYMD,
  TITLE='Date,Problem,Reported',
  DESCRIPTION='Date Problem Was Reported', $
  FIELD=PROBLEM_CATEGORY, ALIAS=PROBCAT, FORMAT=A20, $
  TITLE='Problem,Category',
  DESCRIPTION='Problem Category',
  WITHIN=*Problem,$
  FIELD=PROBLEM_LOCATION, ALIAS=PROBLOC, FORMAT=A10,
  TITLE='Location,Problem,Occurred',
  DESCRIPTION='Location Where Problem Occurred',
  WITHIN=PROBLEM_CATEGORY,$

  DEFINE PROB_YEAR/YY=PROBLEM_DATE;,
  TITLE='Year,Problem,Occurred',
  DESCRIPTION='Year Problem Occurred',
  WITHIN=*Time Period,$

  DEFINE PROB_QUARTER/Q=PROBLEM_DATE;
  TITLE='Quarter,Problem,Occurred',
  DESCRIPTION='Quarter Problem Occurred',
  WITHIN=PROB_YEAR,$

  DEFINE PROB_MONTH/M=PROBLEM_DATE;
  TITLE='Month,Problem,Occurred',
  DESCRIPTION='Month Problem Occurred',
  WITHIN=PROB_QUARTER,$

  DEFINE PROBLEM_OCCUR/I5 WITH PROBNUM=1;,
  TITLE='Problem,Occurrence'
  DESCRIPTION='# of times a problem occurs', $

  DEFINE PLANTLNG/A11=DECODE PLANT (BOS 'Boston' DAL 'Dallas'
  LA 'Los Angeles' ORL 'Orlando' SEA 'Seattle' STL 'St Louis'
  ELSE 'n/a');$

SEGNAME=INVSEG, SEGTYPE=DKU, PARENT=PROD_SEG, CRFILE=CENTINV,
CRKEY=PROD_NUM, CRSEG=INVINFO,$
```

CENTQA Structure Diagram

SECTION 01

STRUCTURE OF FOCUS FILE CENTQA ON 07/30/01 AT 10.46.43

```

          PROD_SEG
01      S1
*****
*PROD_NUM  **I
*          **
*          **
*          **
*          **
*****
          I
          +-----+
          I          I
          I INVSEG  I PROB_SEG
02      I KU          03      I S1
.....
:PROD_NUM   :K  *PROBNUM   **
:PRODNAME   :   *PLANT     **I
:QTY_IN_STOCK: *PROBLEM_DATE**
:PRICE      :   *PROBLEM_CAT>**
:           :   *          **
:.....: *****
JOINED CENTINV *****

```

APPENDIX B

Error Messages

Topic:

- Displaying Messages

If you need to see the text and explanation for any error message, you can display it in your session or find it in an errors file.

- For UNIX, Windows NT/2000, and OpenVMS, the extension is .err.
- For CMS, the file type is ERRORS.
- For MVS, the ddname is ERRORS.

Displaying Messages

To display the text and explanation for any message, issue the following query command in a stored procedure

```
? n
```

where:

```
n
```

Is the message number.

The message number and text will display, along with a detailed explanation of the message (if one exists). For example, issuing the following command

```
? 210
```

displays the following:

```
(FOC210)      THE DATA VALUE HAS A FORMAT ERROR:
```

```
                An alphabetic character has been found where all numerical digits are  
                required.
```

Index

Symbols

&FOCDISORG variable, 10-7
? FILE command, 10-7, 10-20, 10-22 to 10-23

A

A data type, 4-20
absolute file integrity, 10-2
ACCBLN parameter, 4-46
ACCEPT attribute, 4-46 to 4-49
 FOCUS data sources, 6-12
ACCEPTBLANK parameter, 4-46
ACCESS attribute, 9-3, 9-9, 9-15
access control, 9-2
Access Files, 1-2
 applications and, 1-3
 creating, 1-4
 DATASET attribute and, 2-6
access rights, 9-2
access to data, 9-2
 commands, 9-10
 restricting, 9-3, 9-12, 9-15
ACCESSFILE attribute and DATASET attribute,
 2-6
activating external indexes for FOCUS data
 sources, 10-18
ACTUAL attribute, 4-38 to 4-39
ADABAS data sources, 2-3
adding fields, 10-9 to 10-10, 10-12
adding segments, 10-9 to 10-10, 10-12

AIX (alternate index names) for VSAM data
 sources, 5-42
ALIAS attribute, 4-10 to 4-11
aliases of fields, 4-10 to 4-11
ALLBASE/SQL data sources, 2-3
allocating data sources in Master Files, 2-6 to 2-7
ALLOWCVTERR parameter, 4-30
alphanumeric data type, 4-20
 changing field size, 10-9 to 10-10, 10-12
alternate column titles, 4-49 to 4-50
alternate file views, 3-28 to 3-29
 CHECK FILE command and, 8-2
 long field names and, 4-5
alternate index names (AIX) for VSAM data
 sources, 5-42
alternate indexes for VSAM data sources, 5-40 to
 5-42
ancestral segments, 3-10, 3-12
 join linkage, 7-12
applications and data descriptions, 1-2 to 1-3, 2-1
AUTOPATH parameter, 6-3

B

base date, 4-29
blank lines between declarations, 1-6
blank spaces in declarations, 1-6
BUFND parameter, 5-39
BUFNI parameter, 5-39

C

CA-Datcom/DB data sources, 2-3
CA-IDMS/SQL data sources, 2-3
calculations on dates, 4-28
CAR data source, A-10 to A-12
CENTCOMP data source, A-30
CENTFIN data source, A-30
CENTHR data source, A-30
CENTINV data source, A-30
CENTORD data source, A-30
CENTQA data source, A-30
Century Corp data sources, A-30 to A-33, A-35 to A-39
CHECK command, 9-10
CHECK FILE command, 8-1 to 8-4
 DATASET attribute and, 2-6
 DBA and, 9-10
 HELPMESSAGE attribute and, 8-10
 HOLD option, 8-8 to 8-9
 long field names and, 4-5
 non-FOCUS data sources and, 8-4
 PICTURE option, 8-5, 8-7
 TAG attribute and, 8-10
 TITLE attribute and, 8-10
 virtual fields and, 8-10
CHECK option to REBUILD command, 10-20 to 10-21
child-parent segment relationships, 3-8 to 3-12
C-ISAM data sources, 2-3
CLUSTER component, 5-1
column title substitutions, 4-49 to 4-50
columns in relational tables, 1-2

COMBINE command and data security, 9-17 to 9-20
comma-delimited data sources, 2-3, 5-2
 repeating fields, 5-7 to 5-8
comments in Master Files, 1-7, 2-5
converting date formats, 10-26 to 10-28, 10-30
converting date values, 4-28
COURSE data source, A-16
CREATE command, 9-10, 10-1 to 10-3
creating data descriptions, 1-4
creating FOCUS data sources, 10-1 to 10-3
CRFILE attribute, 7-5 to 7-6, 7-9, 7-14
CRKEY attribute, 7-5 to 7-6, 7-14
cross-referenced data sources, 7-3, 7-5
 ancestral segments, 7-12
 descendant segments, 7-9 to 7-10
cross-referenced fields, 7-5
cross-referenced segments, 7-5
CRSEGNAME attribute, 7-5 to 7-6, 7-9

D

D data type, 4-14
 rounding of values, 4-19
data access, 9-2, 9-12
 levels of, 9-9, 9-12
 restricting, 9-3, 9-15, 9-23
DATA attribute, 2-6 to 2-7
data buffers for VSAM data sources, 5-39
data descriptions, 1-1 to 1-2
 applications and, 1-3
 creating, 1-4
 field declarations, 1-4, 4-2
 field relationships, 1-3, 3-1 to 3-4
 file declarations, 1-3, 2-1
 Master Files, 1-3, 1-5

- data encryption, 9-22
 - performance considerations, 9-23
- data paths, 3-12 to 3-14
- data security, 9-4
 - access levels, 9-9, 9-12
 - central Master File, 9-17 to 9-18
 - CHECK FILE command and, 9-10
 - COMBINE command and, 9-17 to 9-20
 - encrypting Master Files, 9-22
 - encrypting segments, 9-22
 - filters, 9-20
 - JOIN command and, 9-17 to 9-20
 - passwords, 9-7 to 9-8
 - restricting access, 9-13 to 9-16
 - special considerations, 9-4
- data source access control, 9-12, 9-23
- data source security, 9-1, 9-3
- data source types, 2-2 to 2-3
- data sources, 1-1
 - accessing, 9-2, 9-12, 9-23
 - allocating in Master Files, 2-6 to 2-7
 - describing field relationships, 1-3, 3-1 to 3-4
 - describing fields, 1-4, 4-2
 - describing files, 1-2 to 1-3, 2-1
 - documenting, 1-7, 2-5
 - multi-dimensional, 4-41 to 4-43
 - OLAP-enabling, 4-41 to 4-43
 - rotating, 3-28 to 3-29
- data types, 4-11 to 4-13, 4-38 to 4-39
 - alphanumeric, 4-20
 - date display options, 4-21
 - date storage, 4-23
 - dates, 4-21, 4-26 to 4-27, 4-31
 - date-time, 4-32 to 4-33, 4-36
 - floating-point double-precision, 4-14
 - floating-point single-precision, 4-15
 - integer, 4-13
 - internal representation, 6-16
 - numeric display options, 4-17 to 4-18
 - packed decimal, 4-16
 - rounding of numeric values, 4-19
 - text, 4-38
- database descriptions, 1-1 to 1-2
 - applications and, 1-3
 - creating, 1-4
 - field declarations, 1-4, 4-2
 - field relationships, 1-3, 3-1 to 3-4
 - file declarations, 1-3, 2-1
 - Master Files, 1-3, 1-5
- database rotation, 3-28 to 3-29
- Datacom/DB data sources, 2-3
- DATASET attribute, 2-6 to 2-7
 - FOCUS data source allocation, 2-9
 - FOCUS data sources, 2-6
 - sequential data sources, 2-10 to 2-11
 - VSAM data sources, 2-11 to 2-12
- date calculations, 4-28
- date conversions, 4-28, 10-26 to 10-28, 10-30
- date data types, 4-21, 4-26 to 4-27, 4-31
 - calculations, 4-28
 - converting values, 4-28
 - Dialogue Manager and, 4-30
 - display options, 4-21
 - extract files and, 4-30
 - graph considerations, 4-30
 - internal representation, 4-29
 - literals, 4-23, 4-26 to 4-27
 - non-standard formats, 4-30
 - RECAP command and, 4-30
 - separators, 4-25
 - storage, 4-23
 - translation, 4-25
- date display options, 4-21
- date literals, 4-23, 4-26 to 4-27
- DATE NEW option to REBUILD command, 10-26 to 10-28, 10-30
- date separators, 4-25
- date translation, 4-25
- DATEDISPLAY parameter, 4-29
 - ALLOWCVTERR and, 4-30
- DATEFORMAT parameter, 4-36
- date-time data type, 4-32 to 4-33, 4-36

- DB2 data sources, 2-3
- DBA (database administration), 9-1
 - attributes, 9-2
 - displaying decision tables, 9-15
 - security, 9-1
- DBA passwords, 9-5
- DBAFILE attribute, 9-17 to 9-18
 - file naming requirements, 9-19
- DBAs (Database Administrators), 9-3
 - identifying, 9-3
- DBATABLE procedure, 9-15
- DBMS data sources, 2-3
- decimal data types, 4-14 to 4-16
 - rounding of values, 4-19
- decision tables, 9-15
- declarations in Master Files, 1-5
 - documenting, 1-7
 - improving readability, 1-6
- decoding values, 7-6
- DECRYPT command, 9-10
- decrypting procedures, 9-24
- DEFCENT attribute, 4-2
 - CHECK FILE command and, 8-2, 8-8 to 8-9
 - date-time data type and, 4-36
- DEFINE attribute, 4-51 to 4-52
- DEFINE command, 9-10
- DEFINE fields in Master Files, 8-10
- defining dimensions, 4-41 to 4-43
- DEFINITION attribute, 4-50 to 4-51
- deleting fields, 10-9 to 10-10, 10-12
- deleting segments, 10-9 to 10-10, 10-12
- DESC attribute, 4-50 to 4-51
- descendant segments, 3-10 to 3-11
 - FOCUS data sources, 6-2
 - join linkage, 7-9 to 7-10
- describing data sources, 1-1 to 1-2
 - Access Files, 1-2
 - field declarations, 1-4, 4-2
 - field relationships, 1-3, 3-1 to 3-4
 - file declarations, 1-3, 2-1
 - Master Files, 1-2
- DESCRIPTION attribute, 4-50 to 4-51
- designing FOCUS data sources, 6-2 to 6-3
- DFSORT sort program, 10-13
- Digital Standard MUMPS data sources, 2-3
- dimensions, 4-41 to 4-43
- display formats for fields, 4-11 to 4-13
 - alphanumeric, 4-20
 - date display options, 4-21
 - date storage, 4-23
 - dates, 4-21, 4-26 to 4-27, 4-31
 - date-time, 4-32 to 4-33, 4-36
 - floating-point double-precision, 4-14
 - floating-point single-precision, 4-15
 - integer, 4-13
 - numeric display options, 4-17 to 4-18
 - packed decimal, 4-16
 - rounding of numeric values, 4-19
 - text, 4-38
- display options, 4-11 to 4-12
 - date, 4-21
 - numeric, 4-17 to 4-18
- DKM (dynamic keyed multiple) segments, 7-14 to 7-15
- DKU (dynamic keyed unique) segments, 7-14 to 7-15
- DMS data sources, 2-3
- documenting data sources, 1-7, 2-5
- documenting fields, 4-50 to 4-51
- double-precision floating-point data type, 4-14
 - rounding of values, 4-19
- DUMMY root segments, 5-25 to 5-28
- duplicate field names, 4-4 to 4-7
 - CHECK FILE command and, 8-2, 8-4

DUPLICATE option to CHECK FILE command,
8-2, 8-4

dynamic join relationships, 7-2, 7-14 to 7-15
static relationships compared to, 7-14 to 7-15

dynamic keyed multiple (DKM) segments, 7-14 to
7-15

dynamic keyed unique (DKU) segments, 7-14, 7-15

E

edit options, 4-11 to 4-12
date, 4-21
numeric, 4-17 to 4-18

EDUCFILE data source, A-7

EMPDATA data source, A-17

EMPLOYEE data source, A-2 to A-4

ENCRYPT attribute, 9-22

ENCRYPT command, 9-10, 9-22

encrypting procedures, 9-24

Enscribe data sources, 2-3

entry-sequenced data sources (ESDS), 5-1

error messages, B-1 to B-2

ESDS (entry-sequenced data sources), 5-1

external indexes for FOCUS data sources, 10-15 to
10-17, 10-19
activating, 10-18

extract files, 8-8
CHECK FILE command and, 8-2, 8-8 to 8-10

F

F data type, 4-15
rounding of values, 4-19

FDEFCENT attribute, 2-1
CHECK FILE command and, 8-2, 8-8 to 8-9
date-time data type and, 4-36

field aliases, 4-10 to 4-11

FIELD attribute, 4-3

field formats, 4-11 to 4-13, 4-38 to 4-39
alphanumeric, 4-20
date display options, 4-21
date storage, 4-23
dates, 4-21, 4-26 to 4-27, 4-31
date-time, 4-32 to 4-33, 4-36
floating-point double-precision, 4-14
floating-point single-precision, 4-15
integer, 4-13
numeric display options, 4-17 to 4-18
packed decimal, 4-16
rounding of numeric values, 4-19
text, 4-38

field names, 4-3 to 4-7
checking for duplicates with CHECK FILE
command, 8-2, 8-4
qualified, 4-9

FIELD option to RESTRICT attribute, 9-11

field values, 9-15 to 9-16
restricting access to, 9-15 to 9-16
validating, 4-46 to 4-49

FIELDNAME attribute, 4-3

FIELDNAME parameter, 4-4

- fields, 1-2, 4-1
 - adding, 10-9 to 10-10, 10-12
 - changing order of, 10-9 to 10-10, 10-12
 - deleting, 10-9 to 10-10, 10-12
 - describing, 1-4, 4-2
 - documenting, 4-50 to 4-51
 - indexing, 10-9 to 10-10, 10-12 to 10-14
 - naming, 4-3 to 4-7
 - redefining, 5-18 to 5-19
 - repeating, 5-2, 5-7, 5-9, 5-33
 - restricting access to, 9-13 to 9-14
- FIELDTYPE attribute, 6-13, 6-15
- file allocations in Master Files, 2-6 to 2-7
- FILE attribute, 2-2
- file descriptions, 1-1 to 1-2
 - applications and, 1-3
 - creating, 1-4
 - field declarations, 1-4, 4-2
 - field relationships, 1-3, 3-1 to 3-4
 - file declarations, 1-3, 2-1
 - Master Files, 1-3, 1-5
- FILENAME attribute, 2-2
- FILESUFFIX attribute, 2-2
- filler fields, 3-5 to 3-6, 5-2
- filters, 9-20
- FINANCE data source, A-14
- FIND option to ACCEPT attribute, 6-12
- fixed-format data sources, 2-3, 5-2
 - allocating in Master Files, 2-10 to 2-11
 - generalized record types, 5-29 to 5-30
 - multiple record types, 5-20 to 5-22, 5-32 to 5-33
 - nested repeating fields, 5-13 to 5-14
 - order of repeating fields, 5-17
 - parallel repeating fields, 5-12, 5-14
 - position of repeating fields, 5-15 to 5-16
 - positionally related records, 5-22 to 5-23
 - repeating fields, 5-9 to 5-11, 5-33
 - unrelated records, 5-25 to 5-26
- floating-point double-precision data type, 4-14
 - rounding of values, 4-19
- floating-point single-precision data type, 4-15
 - rounding of values, 4-19
- FOCUS data sources, 2-3, 6-1
 - ACCEPT attribute, 6-12
 - allocating in Master Files, 2-6 to 2-7, 2-9
 - changing, 6-4
 - creating, 10-1 to 10-3
 - data type representation, 6-16
 - designing, 6-2 to 6-3
 - external indexes, 10-15 to 10-17, 10-19
 - FIND option, 6-12
 - FORMAT attribute, 6-16
 - INDEX attribute, 6-13, 6-15
 - internal storage lengths, 6-16
 - joining, 6-3
 - key fields, 6-7
 - LOCATION attribute, 6-9 to 6-12
 - maintaining, 10-1, 10-4 to 10-5
 - migrating to Fusion, 10-31
 - MISSING attribute, 6-16
 - segment relationships, 6-8
 - segment sort order, 6-7 to 6-8
 - segments, 6-5
 - SEGTYPE attribute, 6-5, 6-7
 - structural integrity, 10-20 to 10-23
 - timestamps, 10-24
- FOCUS Database Server (FDS), 2-6, 2-9
- FORMAT attribute, 4-11 to 4-13
- free-format data sources, 2-3, 5-2, 5-4
 - describing, 5-4
 - repeating fields, 5-7 to 5-8
- Fusion data sources, 2-3, 10-1 to 10-2
 - migrating FOCUS data sources, 10-31
 - multi-dimensional indexes, 10-31
- FYRTHRESH attribute, 2-1
 - CHECK FILE command and, 8-2, 8-8 to 8-9
 - date-time data type and, 4-36

G

generalized record types, 5-29 to 5-30

GGDEMOG data source, A-25

GGORDER data source, A-25

GGPRODS data source, A-25

GGSALES data source, A-25

GGSTORES data source, A-25

Gotham Grinds data sources, A-25 to A-29

group keys, 5-1, 5-5 to 5-7

H

H data type, 4-32 to 4-33, 4-36

HELPMESSAGE attribute, 8-10

HOLD ALL option to CHECK FILE command, 8-2, 8-8 to 8-9

HOLD option to CHECK FILE command, 8-2, 8-8 to 8-9

HOLDSTAT parameter, 9-6

host data sources, 7-3

host fields, 7-5

host segments, 7-5

I

I data type, 4-13

 rounding of values, 4-19

IDCAMS utility, 5-40, 5-42

IDMS/DB data sources, 2-3

IMAGE/SQL data sources, 2-3

IMS data sources, 2-3

INDEX attribute, 6-13, 6-15
 joining data sources, 6-14

index buffers for VSAM data sources, 5-39

INDEX EXTERNAL option to REBUILD command, 10-15 to 10-17, 10-19

INDEX option to REBUILD command, 10-13 to 10-14

indexed fields, 10-9 to 10-10, 10-12 to 10-14
 long field names and, 4-5

INFOAccess data sources, 2-3

Information/Management data sources, 2-3

Informix data sources, 2-3

Ingres data sources, 2-3

integer data type, 4-13
 rounding of values, 4-19

internal representation of data types, 6-16

internal representation of dates, 4-29

ISAM data sources, 5-1

 describing, 5-5

 generalized record types, 5-29 to 5-30

 group keys, 5-6 to 5-7

 multiple record types, 5-20 to 5-22, 5-32 to 5-33

 nested repeating fields, 5-13

 order of repeating fields, 5-17

 parallel repeating fields, 5-12

 position of repeating fields, 5-15 to 5-16

 positionally related records, 5-22 to 5-23

 repeating fields, 5-9 to 5-11, 5-33

 unrelated records, 5-25 to 5-26

ITEMS data source, A-22

- J**
- JOBFILE data source, A-5 to A-6
 - JOIN command, 7-2
 - data security, 9-17 to 9-20
 - long field names and, 4-5
 - joining data sources, 3-7, 3-26, 7-1 to 7-2
 - ancestral segments in cross-referenced file, 7-12
 - descendant segments in cross-referenced file, 7-9 to 7-10
 - dynamic relationships, 7-14 to 7-15
 - FOCUS, 6-3
 - from many host files, 7-17, 7-19
 - from many segments in single host file, 7-17
 - INDEX attribute, 6-14
 - recursive relationships, 3-24 to 3-25, 7-21
 - static relationships, 7-3, 7-14 to 7-15
- K**
- key fields, 3-4
 - FOCUS data sources, 6-7
 - keyed multiple (KM) segments, 7-3, 7-7 to 7-8
 - keyed through linkage (KL) segments, 7-9 to 7-10, 7-12 to 7-15
 - keyed through linkage unique (KLU) segments, 7-9 to 7-10, 7-12 to 7-15
 - keyed unique (KU) segments, 7-3, 7-5 to 7-6
 - decoding values, 7-6
 - key-sequenced data sources (KSDS), 5-1
 - KL (keyed through linkage) segments, 7-9 to 7-10, 7-12 to 7-15
 - KLU (keyed through linkage unique) segments, 7-9 to 7-10, 7-12 to 7-15
 - KM (keyed multiple) segments, 7-3, 7-7 to 7-8
 - KSAM data sources, 2-3
 - KSDS (key-sequenced data sources), 5-1
 - KU (keyed unique) segments, 7-3, 7-5 to 7-6
 - decoding values, 7-6
- L**
- leaf segments, 3-11
 - LEDGER data source, A-13
 - legacy date conversions, 10-26 to 10-28, 10-30
 - literals for dates, 4-23, 4-26 to 4-27
 - load procedures, A-1
 - LOCATION attribute, 6-9 to 6-12
 - logical views of segments, 3-5 to 3-6
 - long field names, 4-4 to 4-7
 - temporary fields, 4-52
- M**
- many-to-many segment relationships, 3-20 to 3-22
 - MAPFIELD alias, 5-36 to 5-39
 - MAPVALUE fields, 5-36 to 5-39
 - Master Files, 1-2 to 1-3, 1-5, 9-2
 - allocating files, 2-6 to 2-7
 - applications and, 1-3
 - common errors, 8-4
 - creating, 1-4
 - declarations, 1-5
 - diagrams, 8-5, 8-7
 - dimensions, 4-41 to 4-43
 - documenting, 1-7, 2-5
 - encrypting, 9-22
 - file statistics, 8-3
 - improving readability, 1-6
 - joining data sources, 7-1 to 7-2
 - names of data sources, 2-2
 - OLAP-enabling, 4-41 to 4-43
 - samples, A-1
 - types of data sources, 2-2
 - validating, 1-7, 8-1 to 8-4
 - MATCH command, 9-10
 - MDINDEX option to REBUILD command, 10-31
 - Micronetics Standard MUMPS data sources, 2-3
 - MIGRATE option to REBUILD command, 10-31

migrating FOCUS data sources to Fusion, 10-31
Millenium data sources, 2-3
MISSING attribute, 4-39 to 4-40
 ALLOWCVTERR and, 4-30
Model 204 data sources, 2-3
MOVIES data source, A-21
multi-dimensional data sources, 4-41 to 4-43, 10-1 to 10-2
 migrating FOCUS data sources, 10-31
multi-dimensional indexes, 10-31
multi-path data structures, 3-13
multiple join relationships, 7-3
 dynamic, 7-14 to 7-15
 static, 7-7 to 7-8
multiple record types, 5-20 to 5-22, 5-32 to 5-33, 5-35
multiply occurring fields, 5-2, 5-7 to 5-11, 5-33, 5-35
 MAPFIELD and MAPVALUE, 5-36 to 5-39
 nested, 5-13 to 5-14
 order, 5-17
 parallel, 5-12, 5-14
 position, 5-15 to 5-16
 record length, 5-19
MUMPS data sources, 2-3

N

naming fields, 4-3 to 4-7
naming segments, 3-4 to 3-5
National Language Support (NLS), 1-5
Native Interface data sources, 2-3
nested repeating fields, 5-13 to 5-14
NETISAM Interface data sources, 2-3
NOMAD data sources, 2-3

NonStop SQL data sources, 2-3
NOPRINT option to RESTRICT attribute, 9-11
Nucleus data sources, 2-3
null values, 4-39 to 4-40
numeric data types, 4-13
 display options, 4-17 to 4-18
 floating-point double precision, 4-14
 floating-point single precision, 4-15
 integer, 4-13
 packed-decimal, 4-16
 rounding of values, 4-19
numeric display options, 4-17 to 4-18

O

OCCURS attribute, 5-10 to 5-11, 5-14
ODBC data sources, 2-3
OLAP-enabling data sources, 4-41 to 4-43
one-to-many join relationships, 7-3
 dynamic, 7-14 to 7-15
 static, 7-7 to 7-8
one-to-many segment relationships, 3-18 to 3-20
 FOCUS data sources, 6-2, 6-8
one-to-one join relationships, 7-3
 decoding values, 7-6
 dynamic, 7-14 to 7-15
 static, 7-3, 7-5 to 7-6
one-to-one segment relationships, 3-15 to 3-17
 FOCUS data sources, 6-2, 6-8
Open M/SQL data sources, 2-3
OpenIngres data sources, 2-3
Oracle data sources, 2-3
ORDER fields, 5-17

P

- P data type, 4-16
 - rounding of values, 4-19
- PACE data sources, 2-3
- packed decimal data type, 4-16
 - rounding of values, 4-19
- parallel repeating fields, 5-12, 5-14
- PARENT attribute, 3-8
- parent-child segment relationships, 3-8 to 3-12
- passwords, 9-2
 - changing, 9-5
 - setting externally, 9-24
- paths in data sources, 3-12 to 3-14
 - FOCUS data sources, 6-2
- PICTURE option to CHECK FILE command, 8-2, 8-5, 8-7
- POSITION attribute, 5-15 to 5-16
- positionally related records, 5-22 to 5-24
- primary keys, 3-4
- procedure security, 9-24
- Progress data sources, 2-3

Q

- QUALCHAR parameter, 4-5
- qualification character, 4-5
- qualified field names, 4-4 to 4-7, 4-9
 - levels of qualification, 4-9
 - temporary fields, 4-52

R

- Rdb data sources, 2-3
- read/write access, 9-9
- read-only access, 9-9, 9-15
- REBUILD command, 9-10, 10-1, 10-4 to 10-5
 - CHECK option, 10-20 to 10-21
 - DATE NEW option, 10-26 to 10-28, 10-30
 - INDEX EXTERNAL option, 10-15 to 10-17, 10-19
 - INDEX option, 10-13 to 10-14
 - MDINDEX option, 10-31
 - message frequency, 10-6
 - MIGRATE option, 10-31
 - REBUILD option, 10-7 to 10-8
 - REORG option, 10-9 to 10-10, 10-12
 - requirements, 10-5
 - TIMESTAMP option, 10-24
- REBUILD option to REBUILD command, 10-7 to 10-8
- record length in sequential data sources, 5-19
- record types, 5-2
 - generalized, 5-29 to 5-30
 - multiple, 5-20 to 5-22, 5-32 to 5-33, 5-35
- RECTYPE fields, 5-20 to 5-22, 5-29 to 5-30, 5-32 to 5-33, 5-35
 - MAPFIELD and MAPVALUE, 5-36 to 5-39
- recursive join relationships, 3-24 to 3-25, 7-21
- Red Brick data sources, 2-3
- redefining fields in non-FOCUS data sources, 5-18 to 5-19
- REGION data source, A-15
- relating segments, 3-7, 3-14, 3-26
 - many-to-many relationships, 3-20 to 3-22
 - one-to-many relationships, 3-18 to 3-20
 - one-to-one relationships, 3-15 to 3-17
 - parent-child relationships, 3-8 to 3-12
 - recursive relationships, 3-24 to 3-25
- REMARKS attribute, 2-5

REORG option to REBUILD command, 10-9 to 10-10, 10-12

REORG PERCENT measure, 10-7

reorganizing FOCUS data sources, 10-1, 10-4 to 10-5
requirements, 10-5

repeating fields, 5-2, 5-7 to 5-11, 5-33, 5-35
MAPFIELD and MAPVALUE, 5-36 to 5-39
nested, 5-13 to 5-14
order, 5-17
parallel, 5-12, 5-14
position, 5-15 to 5-16
record length, 5-19

RESTRICT attribute, 9-3, 9-10, 9-12
keywords, 9-10 to 9-11

RESTRICT command, 9-10, 9-23

restricting access, 9-13 to 9-14
to fields, 9-13 to 9-14
to segments, 9-13 to 9-14

retrieval paths and CHEK FILE command, 8-2

RETRIEVE option to CHECK FILE command, 8-2

RMS data sources, 2-3

root segments, 3-3, 3-10 to 3-11
describing, 3-8
FOCUS data sources, 6-2

rounding of numeric values, 4-19

S

SALES data source, A-8 to A-9

SAME option to RESTRICT attribute, 9-11

sample data sources, A-1
CAR, A-10 to A-12
Century Corp, A-30 to A-33, A-35 to A-39
COURSE, A-16
EDUCFILE, A-7
EMPDATA, A-17
EMPLOYEE, A-2 to A-4
FINANCE, A-14
Gotham Grinds, A-25 to A-29

sample data sources (*continued*)
ITEMS, A-22
JOBFILE, A-5 to A-6
LEDGER, A-13
MOVIES, A-21
REGION, A-15
SALES, A-8 to A-9
TRAINING, A-18
VIDEOTR2, A-23 to A-24
VideoTrk, A-19 to A-20

security, 9-1 to 9-2, 9-4
access levels, 9-9, 9-12
encrypting data segments, 9-22
encrypting Master Files, 9-22
encrypting procedures, 9-24
identifying users, 9-6
passwords, 9-7 to 9-8
storing DBA information centrally, 9-17 to 9-18

security attributes, 9-2, 9-21
ACCESS, 9-9
DBA, 9-5
DBAFILE, 9-17 to 9-18
RESTRICT, 9-12
USER, 9-6

SEGMENT attribute, 3-3 to 3-5
VSAM and ISAM considerations, 5-5

segment chains, 3-3

segment instances, 3-3

SEGMENT option to RESTRICT attribute, 9-11

segments, 1-3, 3-2
adding, 10-9 to 10-10, 10-12
data paths, 3-12 to 3-14
deleting, 10-9 to 10-10, 10-12
encrypting, 9-22
excluding fields from, 3-5 to 3-6
key fields, 3-4
naming, 3-4 to 3-5
parent-child relationships, 3-8 to 3-12
relating, 3-7, 3-14 to 3-22, 3-24 to 3-26
restricting access to, 9-13 to 9-14
sort order, 3-4

SEGNAME attribute, 3-3 to 3-5
VSAM and ISAM considerations, 5-5

- SEGTYPE attribute, 3-3 to 3-4, 3-8
 - displaying, 8-5, 8-7
 - FOCUS data sources, 6-5, 6-7 to 6-8
 - sequential data source considerations, 5-5
 - VSAM considerations, 5-5
 - separators for dates, 4-25
 - sequential data sources, 2-3, 5-1 to 5-2
 - allocating in Master Files, 2-10 to 2-11
 - describing, 5-5
 - fixed format, 5-2
 - free format, 5-4
 - generalized record types, 5-29 to 5-30
 - multiple record types, 5-20 to 5-22, 5-32 to 5-33
 - multiply occurring fields, 5-19
 - nested repeating fields, 5-13 to 5-14
 - order of repeating fields, 5-17
 - parallel repeating fields, 5-12, 5-14
 - position of repeating fields, 5-15 to 5-16
 - positionally related records, 5-22 to 5-23
 - record length, 5-19
 - repeating fields, 5-7 to 5-11, 5-33
 - unrelated records, 5-25 to 5-26
 - SET parameters
 - ACCBLN, 4-46
 - ACCEPTBLANK, 4-46
 - ALLOWCVTERR, 4-30
 - AUTOPATH, 6-3
 - DATEDISPLAY, 4-29
 - DATEFORMAT, 4-36
 - FIELDNAME, 4-4
 - HOLDSTAT, 9-6
 - PASS, 9-6 to 9-8
 - QUALCHAR, 4-5
 - SHADOW, 10-2
 - SORTLIB, 10-13
 - USER, 9-6 to 9-8
 - setting passwords externally, 9-24
 - SHADOW parameter, 10-2
 - single-path data structures, 3-12
 - single-precision floating-point data type, 4-15
 - rounding of values, 4-19
 - sort libraries, 10-13
 - sort order of segments, 3-4
 - SORTIN ddname, 10-13
 - SORTLIB parameter, 10-13
 - SORTOUT ddname, 10-13
 - SQL Server data sources, 2-3
 - SQL StorHouse data sources, 2-3
 - SQL Translator and long field names, 4-5
 - SQL/DS data sources, 2-3
 - static join relationships, 7-2 to 7-3, 7-7
 - decoding values, 7-6
 - dynamic relationships compared to, 7-14 to 7-15
 - multiple, 7-7 to 7-8
 - unique, 7-3, 7-5 to 7-6
 - storage of date data types, 4-23
 - structural integrity of FOCUS data sources, 10-20 to 10-23
 - structure diagrams, A-1
 - SUFFIX attribute, 2-2
 - VSAM and ISAM, 5-5
 - SUFFIX values, 2-3
 - SUPRA data sources, 2-3
 - Sybase data sources, 2-3
 - SYNCSORT sort program, 10-13
 - System 2000 data sources, 2-3
- ## T
- TABLE command, 9-10
 - TAG attribute and CHECK FILE command, 8-10
 - temporary fields, 4-51
 - creating, 4-52
 - in Master Files, 8-10
 - long field names, 4-52
 - long field names and, 4-5
 - qualified field names, 4-52
 - Teradata data sources, 2-3

text data type, 4-38
 LOCATION attribute, 6-11 to 6-12
 long field names and, 4-5

timestamp data type, 4-32 to 4-33, 4-36

TIMESTAMP option to REBUILD command, 10-24

timestamps for FOCUS data sources, 10-24

TITLE attribute, 4-49 to 4-50
 CHECK FILE command and, 8-10

TOTAL data sources, 2-3

TRAINING data source, A-18

translation of dates, 4-25

TurboIMAGE data sources, 2-3

TX data type, 4-38

U

Unify data sources, 2-3

unique join relationships, 7-3
 decoding values, 7-6
 dynamic, 7-14 to 7-15
 static, 7-3, 7-5 to 7-6

uniVerse data sources, 2-3

unrelated records, 5-25 to 5-28

update access, 9-9

USAGE attribute, 4-11 to 4-13

USE command, 10-7
 DATASET attribute and, 2-6

USER attribute, 9-6

user identification, 9-2 to 9-3

V

validating field values, 4-46 to 4-49

validating Master Files, 1-7

VALUE option to RESTRICT attribute, 9-11, 9-15 to 9-16

values, 9-15
 restricting access to, 9-15

VIDEOTR2 data source, A-23 to A-24

VideoTrk data source, A-19 to A-20

virtual fields, 4-51
 creating, 4-52
 in Master Files, 8-10
 long field names, 4-52
 qualified field names, 4-52

VMSORT sort program, 10-13

VSAM data sources, 2-3, 5-1
 allocating in Master Files, 2-11 to 2-12
 alternate indexes, 5-40 to 5-42
 data buffers, 5-39
 describing, 5-5
 generalized record types, 5-29 to 5-30
 group keys, 5-6 to 5-7
 IDCAMS utility, 5-40, 5-42
 index buffers, 5-39
 multiple record types, 5-20 to 5-22, 5-32 to 5-33, 5-35
 nested repeating fields, 5-13 to 5-14
 order of repeating fields, 5-17
 parallel repeating fields, 5-12, 5-14
 position of repeating fields, 5-15 to 5-16
 positionally related records, 5-22 to 5-24
 repeating fields, 5-9 to 5-11, 5-33, 5-35
 unrelated records, 5-25 to 5-28

W

WITHIN attribute, 4-41 to 4-43

write-only access, 9-9

Y

Y2K attributes in Master Files, 2-1, 4-2

CHECK FILE command, 8-8 to 8-9

CHECK FILE command and, 8-2

date-time data type, 4-36

Year 2000 attributes in Master Files, 4-2

CHECK FILE command, 8-8 to 8-9

CHECK FILE command and, 8-2

date-time data type, 4-36

Year 2000 in Master Files, 2-1

YRTHRESH attribute, 4-2

CHECK FILE command and, 8-2, 8-8 to 8-9

date-time data type and, 4-36

Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

Mail: Documentation Services – Customer Support
Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898

Fax: (212) 967-0460

E-mail: books_info@ibi.com

Web form: <http://www.informationbuilders.com/bookstore/derf.html>

Name: _____

Company: _____

Address: _____

Telephone: _____ Date: _____

E-mail: _____

Comments: