iWay

iWay Adapter Administration for UNIX, Windows, OpenVMS, OS/400, OS/390, and z/OS
Version 5 Release 3.3

DN3501279.1204

Preface

This manual describes configuration and management of Adapters.

Documentation Conventions

The following conventions apply throughout this manual:

Convention	**Description**
`THIS TYPEFACE` or `this typeface`	Denotes syntax that you must enter exactly as shown.
`this typeface`	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
`underscore`	Indicates a default setting.
this typeface	Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option you can click or select.
this typeface	Highlights a file name or command.
Key + Key	Indicates keys that you must press simultaneously.
`{ }`	Indicates two or three choices; type one of them, not the braces.
`[ ]`	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
`|`	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
`. . .`	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (…).
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

To view a current listing of our publications and to place an order, visit our World Wide Web site, http://www.iwaysoftware.com. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have questions about iWay Adapters?

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 iWay Data Adapters 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 (*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 (*xxxx.xx*).
- Your iWay Software configuration:
  - The iWay Software version and release. You can find your server version and release using the *Version* option in the Web Console. (**Note:** the MVS and VM servers do not use the Web Console.)
  - The communications protocol (for example, TCP/IP or LU6.2), including vendor and release.
- The stored procedure (preferably with line numbers) or SQL statements being used in server access.
- The database server release level.

- The database name and release level.
- 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, 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 just the code to access the 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 welcomes your opinions 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 Documentation Feedback form on our Web site, http://www.iwaysoftware.com.

Thank you, in advance, for your comments.

iWay Software Training and Professional Services

Interested in training? Our Education Department offers a wide variety of training courses for iWay Software 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.iwaysoftware.com) or call (800) 969-INFO to speak to an Education Representative.

Interested in technical assistance for your implementation? Our Professional Services department provides expert design, systems architecture, implementation, and project management services for all your business integration projects. For information, visit our World Wide Web site (http://www.iwaysoftware.com).

Contents

CHAPTER 1

Introduction to iWay Adapters

Topics:

In client/server architecture, adapters enable clients to manage data from virtually any data source, and on any operating system, through the use of SQL statements.

The client generates requests for data residing on the server. The server acts as a source of data, and can accept requests from multiple clients for data access and manipulation. Client/server architecture divides a traditional single system into a front-end and a back-end. The workload is distributed between the client and the server. Communications software establishes the link between client and server, and interfaces to the desired communications protocol.

A client application uses SQL as the standard access language for requests to all relational and non-relational data. Depending on the environment—that is, the hardware and software employed throughout the enterprise—an applicable communications protocol transmits the request for data to the server, and then returns the answer set to the client.

The server in client/server architecture is used for data access and manipulation. The server receives a request for data, processes it, and returns an answer set or message to the client. Adapters are among the various subsystems that comprise a server.

Functions of an Adapter

In this section:

The server uses adapters to access data sources. The server receives SQL requests from the client and passes them to the adapter in a standard format. The adapter takes the request, transforms it into the native data manipulation language (DML), and then issues calls to the data source using its API. In this way, the adapter insulates the server from details of the data source. An application can issue SQL statements or call stored procedures.

Adapters are available for many data sources. Every adapter is specifically designed for the data source that it accesses, and, as a result, is able to translate between SQL and the DML of the data source. Adapters provide solutions to product variations, including product differences in syntax, functionality, schema, data types, catalogs, data representations, message processing, and answer set retrieval.

How an Adapter Works

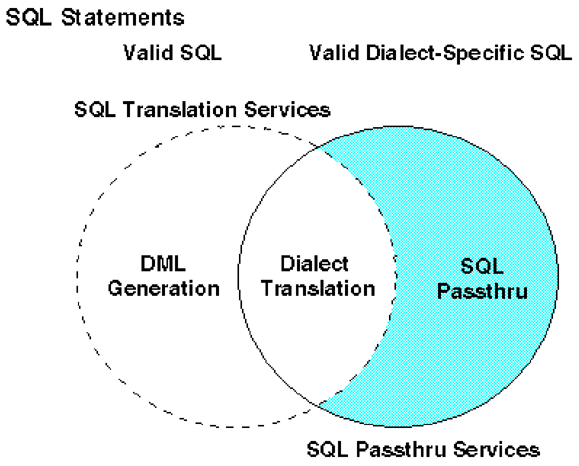
The adapter manages the communication between the data interface and the data source, passing data management requests to the data source and returning either answer sets or messages to the requestor.

To perform these functions, the adapter:

1. Translates the request to the applicable DML.
2. Attaches to the targeted data source, using standard attachment calls. The adapter then passes the request to the data source.
3. The data source processes the request.
4. The results or error conditions are returned to the client application for further processing.

Processing SQL Requests

A server can be configured to behave in different ways upon receipt of SQL requests from a client application. A server handles SQL requests for data in the following ways, as illustrated in the diagram:



- **Direct Passthru.** When Direct Passthru is enabled, the server passes SQL requests directly to the specified RDBMS for processing. The name of the targeted RDBMS (the database engine) is supplied in the server profile or, in some cases, by the client application. A Full-Function or Hub Server can operate temporarily in Direct Passthru mode when invoked by a client application. The user is responsible for activating and deactivating Direct Passthru as needed. Direct Passthru is described in more detail in the *SQL Reference* manual.
- **SQL Processing.** When the database engine is not set in the server profile or supplied by a client application, Direct Passthru is not enabled. Instead, a Hub or Full-Function Server invokes its default behavior: it accepts the incoming SQL request and verifies that it is valid. Then the server determines if it can process the incoming SQL request:
    - If the request meets certain requirements, the server passes it directly to the RDBMS for processing. This is called Automatic Passthru. Automatic Passthru is described in more detail in the *SQL Reference* manual.
    - If the syntax of the request does not conform to the syntax of the RDBMS, the server translates the request into internal DML and passes it to the adapter. The adapter generates adapter-specific DML and passes the request to the RDBMS for processing. This is called SQL translation. SQL translation is described in more detail in the *SQL Reference* manual.

Relational and Non-Relational Adapters

Adapters can retrieve answer sets from both relational and non-relational data sources. Since the architectures of relational and non-relational data structures vary, the relational and non-relational adapters adjust for these differences. For example, relational adapters are designed to handle data sources that contain data in rows and columns in tables, while non-relational adapters are designed to accommodate the architecture of each distinct data source, for instance, a hierarchical or network data source, or a sequential or indexed file system.

The following table lists key features of relational and non-relational adapters:

Feature	**Relational Adapters**	**Non-Relational Adapters**
DEFINE (virtual field) in Master File	Yes	Yes
Access Control	Yes	Yes
Transaction Management Commands	Yes	No

Relational and non-relational adapters:

- Allow the data source to perform the work required to join, sort, and aggregate data. Therefore, the volume of data source-to-server communication is reduced, resulting in better response times for users. The adapter tries to optimize the queries as best it can.
- Communicate with the data source through DML statements. You can view these DML statements with traces provided by the trace facilities. These traces are helpful for debugging your procedure or for adapter performance analysis. Note that traces:
  - Are common for all relational adapters.
  - Vary for non-relational adapters to accommodate the differences in data structures and DML calls.
- Support both DEFINE (virtual) fields and DBA.
- Relational adapters include a variety of COMMIT, connection, and thread control commands that enable Database Administrators to control the opening and closing of connections and choose when to commit or rollback transactions. This level of transaction control is not supported by non-relational adapters. In Full-Function Server mode, COMMIT/ROLLBACK will be propagated to all relational data sources local to the server. If the hub is active, a COMMIT will be issued against remote data servers as well. All PREPARE handles are cleared (this is a Broadcast COMMIT).

To address these similarities and differences, the *Server Administration* manual contains:

- *General* components with information that is common to all adapters, as well as information that is common either to all relational or to all non-relational adapters.
- *Customized* components with information that applies to specific adapters.

Supported Adapters

The following table lists supported adapters:

Windows	**UNIX-Based Computers**
Adabas/C	CICS Transactions
Adabas Stored Procedures	C-ISAM (Informix and Micro Focus)
CICS Transactions	DB2
C-ISAM (Informix, Micro Focus)	Enterprise Java Beans
DB2/2	Essbase
Enterprise Java Beans	Flat Files
Essbase	IMS Transactions
FOCUS	Informix
Flat Files	Ingres
IMS Transactions	JDBC
Informix	Lawson
Interplex (DMS/RDMS 2200/1100)	Microsoft SQL Server
JDBC	MQSeries
Lawson	Oracle E-Business Suite
Microsoft Access	PeopleSoft World
Microsoft SQL Server	PeopleSoft EnterpriseOne
Microsoft Analysis Server	Progress
MQSeries	Red Brick
MySQL	SAP BW
Nucleus	SAP R/3
ODBC	Siebel

Windows	**UNIX-Based Computers**
Oracle	Sybase ASE
Oracle E-Business Suite	Sybase IQ
PeopleSoft	Teradata
PeopleSoft World	UniVerse
PeopleSoft EnterpriseOne	Web Services
Progress	XML
Red Brick	
SAP BW	
SAP R/3	
Siebel	
Sybase ASE	
Sybase IQ	
Tamino	
Teradata	
UniVerse	
Web Services	
XML	

OS/390 and z/OS	**OpenVMS**	**OS/400**
Adabas	Adabas/C	CICS Transactions
Adabas Stored Procedures	CICS Transactions	DB2
CA-IDMS/DB	Flat Files	DBFILE
CA-IDMS/SQL	IMS Transactions	Enterprise Java Beans
CICS Transactions	Ingres	Flat Files
DATACOM/DB	Lawson	JDBC
DB2	Digital Standard Mumps	Lawson
Enterpise Java Beans	Oracle	Microsoft SQL Server
Essbase	Oracle E-Business Suite	MQSeries
Flat Files	PeopleSoft World	PeopleSoft World
IMS/DB	PeopleSoft EnterpriseOne	PeopleSoft EnterpriseOne
IMS Transactions	Progress	Siebel
Information Manager	Rdb	Web Services
Informix	RMS	XML
JDBC	Web Services	
Lawson	XML	
Microsoft SQL Server		
Millennium		
MODEL 204		
MQSeries		
Oracle		
PeopleSoft		
PeopleSoft World		
PeopleSoft EnterpriseOne		
SAP BW		
SAP R/3		
Supra		
System 2000		
Teradata		
TOTAL		
VSAM		
VSAM CICS		
Web Services		
XML		

Data Management

In this section:

As you manage your data, you may be required to modify your server and communications configuration files. The first step is understanding how and where data is described and the roles of the server and adapters in managing the processing flow.

Describing Data Sources

In order to access a table or view, you must first describe it using two files: a Master File and an associated Access File.

Master Files and Access Files can represent an entire table or part of a table. Also, several pairs of Master and Access Files can define different subsets of columns for the same table, or one pair of Master and Access Files can describe several tables.

Note: In these topics, the term table refers to both base tables and views in data sources. The Master File describes the columns of the data source table using keywords in comma-delimited format. The Access File includes additional parameters that complete the definition of the data source table. Some adapters require both files to fulfill queries, and to build the DML to access the non-SQL data sources.

Processing Requests

When requests are processed, control is passed from the server to an adapter and back. During the process, selected information is read from the Master and Access Files as described below.

The server processes a request as follows:

1. The request is parsed to identify the table.
2. The Master File for the table is read.
3. The SUFFIX value in the Master File is checked (SUFFIX indicates the type of data source).
4. Control is passed to the appropriate adapter.

The adapter then:

5. Locates the corresponding Access File.
6. Uses the information contained in the Master and Access Files to generate the DML statements (if necessary) required to accomplish the request.
7. Passes the DML statements to the data source.
8. Retrieves the answer set generated by the data source.
9. Returns control to the server.

Depending on the requirements of the request, additional processing may be performed on the returned data.

Master File

A Master File describes a logical data source. A logical data source can be made up of one or more physical data sources of the same type. Each segment is a physical data source.

Master Files contain three types of declarations:

Declaration Type	Description
File	Names the file and describes the type of data source.
Segment	Identifies a table, file, view, or segment.
Field	Describes the columns of the table, view, or fields in the file.

The following guidelines apply:

- A declaration consists of attribute-value pairs separated by commas.
- Each declaration must begin on a separate line. A declaration can span as many lines as necessary, as long as no single keyword-value pair spans two lines.
- Do not use system or reserved words as names for files, segments, fields, or aliases. Specifying a reserved word generates syntax errors.

Syntax: How to Specify a File Declaration in a Master File

A Master File begins with a file declaration, which has at least two attributes:

`FILENAME (FILE)`

Identifies the Master File.

`SUFFIX`

Identifies the adapter needed to interpret the request.

The syntax for a file declaration is

```
FILE[NAME]=file, SUFFIX=suffix [,$]
```

where:

`file`

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters.

`suffix`

Identifies the adapter needed to interpret the request. For example, SQLORA is the value for the Adapter for Oracle.

Syntax: How to Specify a Segment Declaration in a Master File

Each table described in a Master File requires a segment declaration. The segment declaration consists of at least two attributes:

`SEGNAME`

Identifies one table.

`SEGTYPE`

Identifies the physical storage of rows and the uniqueness of column values.

The syntax for a segment declaration is

```
SEGNAME=segname, SEGTYPE=S0 [,$]
```

where:

`segname`

Is the segment name which serves as a link to the actual table name. It may be the same as the name chosen for FILENAME, the actual table name, or an arbitrary name. It can consist of a maximum of 8 alphanumeric characters.

The SEGNAME value in the Master File must be the same as the SEGNAME value specified in the Access File, where the TABLENAME portion of the segment declaration contains the fully-qualified name of the table.

`S0`

Indicates that the RDBMS is responsible for both physical storage of rows and the uniqueness of column values (if a unique index or constraint exists). It always has a value of S0 (S zero).

Syntax: How to Specify a Field Declaration in a Master File

Each row in a table may consist of one or more columns. These columns are described in the Master File as fields with the following primary field attributes:

`FIELDNAME`

Identifies the name of a field.

`ALIAS`

Identifies the full column name.

`USAGE`

Identifies how to display a field on reports.

`ACTUAL`

Identifies the data type and length in bytes for a field.

`MISSING`

Identifies whether a field supports null data.

You can obtain values for these attributes by using the system catalog table ALL_TAB_COLUMNS for the existing table or view you wish to describe.

The syntax for a field declaration is

```
FIELD[NAME]=fieldname, [ALIAS=]sqlcolumn, [USAGE=]display_format,
     [ACTUAL=]storage_format [,MISSING={ON|OFF}], $
```

where:

fieldname

Is the name of the field. This value must be unique within the Master File. The name can consist of a maximum of 48 alphanumeric characters including letters, digits, and underscores. The name must begin with a letter. Special characters and embedded blanks are not recommended. The order of field declarations in the Master File is significant with regard to the specification of key columns. For more information, see *Primary Key* on page 1-15.

Tip: Since the name appears as the default column title for reports, for client applications, or EDADESCRIBE, select a name that is representative of the data.

It is not necessary to describe all the columns of the table in your Master File.

sqlcolumn

Is the full column name (the adapter uses it to generate SQL statements). This value must comply with the naming conventions for identifiers, where a name should start with a letter and may be followed by any combination of letters, digits, or underscores. Embedded spaces are not allowed.

display_format

Is the display format. The value must include the field type and length and may contain edit options.

The data type of the display format must be identical to that of the ACTUAL format. For example, a field with an alphanumeric USAGE data type must have an alphanumeric ACTUAL data type.

Fields or columns with decimal or floating point data types must be described with the correct scale (s) and precision (p). Scale is the number of positions to the right of the decimal point. Precision is the total length of the field.

For the server, the total display length of the field or column *includes* the decimal point and negative sign. In SQL, the total length of the field or column *excludes* the decimal point and negative sign. For example, a column defined as DECIMAL(5,2) would have a USAGE attribute of P7.2 to allow for the decimal point and a possible negative sign.

storage_format

Is the storage format of the data type and length in bytes. For more information on data type support, see the *SQL Reference* manual.

`ON`

Displays the character specified by the NODATA parameter for missing data. For related information, see *MISSING Attribute* on page 1-13.

`OFF`

Displays blanks or zeroes for fields having no value. This is the default. See *MISSING Attribute* on page 1-13.

MISSING Attribute

In a table, a null value represents a missing or unknown value; it is not the same as a blank or a zero. For example, a column specification that allows null values is used where a column need not have a value in every row (such as a raise amount in a table containing payroll data).

- The default NODATA character is a period.
- A column in a table that allows null data does not need to include the NULL clause in its table definition, since that is the default.
- In the Master File for that table, the column that allows null data must be described with the MISSING attribute value ON. The default for this attribute is OFF, which corresponds to the NOT NULL attribute in the table definition.

If the column allows null data but the corresponding field in the Master File is described with the MISSING attribute value OFF, null data values appear as zeroes or blanks.

Access File

Each Master File may have a corresponding Access File. The name of the Access File must be identical to that of the Master File, but the extension will be .acx instead of .mas.

The Access File serves as a link between the server and the data source by providing the means to associate a segment in the Master File with the table it describes. The Access File minimally identifies the table and primary key (if there is one). It may also indicate the logical sort order of data and identify storage areas for the table.

Syntax: How to Specify a Segment Declaration in an Access File

The segment declaration in the Access File establishes the link between one segment of the Master File and the actual table or view. Attributes that constitute the segment declaration are:

`SEGNAME`

Identifies one table.

`TABLENAME`

Identifies the table or view. It may contain the owner ID as well as the table name.

`KEYS`

Identifies how many columns constitute the primary key.

`KEYORDER`

Identifies the logical sort sequence of data by the primary key.

The syntax for a segment declaration in an Access File is

```
SEGNAME=segname, TABLENAME=owner.tablename databaselink
   [,KEYS={n|0}] [,KEYORDER={LOW|HIGH}] ,$
```

where:

`segname`

Is the same value as the SEGNAME value in the Master File.

`owner`

Is the user ID by default.

`tablename`

Is the name of the table or view.

`databaselink`

Is the DATABASE LINK name to be used in the currently connected database server.

`n`

Is the number of columns that constitute the primary key. It can be a value from 0 to 16. The default value is 0. For more information, see *Primary Key* on page 1-15.

`LOW`

Indicates an ascending primary key logical sort order. This value is the default.

`HIGH`

Indicates a descending primary key logical sort order.

Primary Key

A table's primary key consists of the column or combination of columns whose values uniquely identify each row of the table. In the employee table, for example, every employee is assigned a unique employee identification number. Each employee is represented by one and only one row of the table, and is uniquely identified by that identification number.

The primary key definition must be defined partly in the Master File and partly in the Access File:

- The order of field declarations in the Master File is significant to the specification of key columns. To define the primary key in a Master File, describe its component fields immediately after the segment declaration. You can specify the remaining fields in any order. In the Access File, the KEYS attribute completes the process of defining the primary key.
- To identify the primary key, the adapter uses the number of columns (*n*) indicated by the KEYS attribute in the Access File and the first *n* fields described in the Master File.

Typically, the primary key is supported by the creation of a unique index in the SQL language to prevent the insertion of duplicate key values. The adapter itself does not require any index on the column(s) comprising the primary key (although a unique index is certainly desirable for both data integrity and performance reasons).

Creating Virtual Fields

You use the DEFINE command to accomplish these tasks.

Syntax: How to Create Virtual Fields With the DEFINE Command

```
DEFINE fieldname/format [WITH fieldname]=expression ;$
```

where:

fieldname

Is a field name for the virtual field. It can consist of 1 to 48 characters. You must not qualify the field name.

format

Provides the display format for the field and follows the rules for USAGE formats. This operand is optional. If not specified, the default value is D12.2.

WITH *fieldname*

Must be coded when the expression is a constant. Any real field can be chosen from the same segment the DEFINE is associated with.

`expression`

Can be either a mathematical or a logical statement. It can consist of constants, database fields, and virtual fields. The expression must end with a semicolon followed by a dollar sign (;$).

Place your DEFINE statements after all of the field descriptions in the segment. If you are using the DESCRIPTION or TITLE attributes with virtual fields, you must place these attributes on a separate line.

Example: Defining a Virtual Field in a Master File

In the example that follows, the virtual field PROFIT is defined at the end of the segment named BODY.

```
SEGMENT=BODY, SEGTYPE=S0 , PARENT=CARREC,$
 FIELDNAME=BODYTYPE          ,ALIAS=BODYTYPE      ,A12,A12,$
FIELDNAME=DEALER_COST        ,ALIAS=DEALER_COST   ,D8, D8 ,$
FIELDNAME=RETAIL_COST        ,ALIAS=RETAIL_COST   ,D8, D8 ,$
DEFINE PROFIT/D8 = RETAIL_COST - DEALER_COST
 ;DESC=NET_COST, TITLE='NET,COST' ,$
```

As a result of this DEFINE statement, you can use PROFIT as a field name in reports. PROFIT is treated as a field with a value equal to the value of RETAIL_COST minus DEALER_COST.

Note:

- Since the complete data source needs to be read to calculate virtual fields, screening conditions on virtual fields may incur additional overhead.
- Virtual fields in the Master File for relational and remote data sources will, if referenced in a query, disable Automatic Passthru.

Cross-Century Dates

Many existing business applications use two digits to designate a year, instead of four digits. When they receive a value for a year, such as 00, they typically interpret it as 1900, assuming that the first two digits are 19, for the twentieth century. There is considerable risk that date-sensitive calculations in existing applications will be wrong unless an apparatus is provided for determining the century in question. This will impact almost every type of application, including those that process mortgages, insurance policies, anniversaries, bonds, inventory replenishment, contracts, leases, pensions, receivables, and customer records.

The cross-century dates feature enables you to solve this problem at the file and field level of your applications. You can retain your global settings while changing the file-level settings for greater flexibility.

You can enable this feature:

- Using SET commands.
- At the file level in a Master File.
- At the field level in a Master File.

Cross-Century Dates SET Commands

The server delivers SET commands that provide a means of interpreting the century if the first two digits of the year are not provided:

```
SET DEFCENT
SET YRTHRESH
```

If the first two digits are provided, they are simply accepted and validated.

Syntax: How to Implement a Cross-Century Date

The DEFCENT syntax is

```
SET DEFCENT=nn
```

where:

nn

Is 19 unless otherwise specified.

The YRTHRESH syntax is

```
SET YRTHRESH=nn
```

where:

nn

Is zero unless otherwise specified.

The combination of DEFCENT and YRTHRESH establishes a base year for a 100-year window. Any 2-digit year is assumed to fall within that window, and the first two digits are set accordingly. Years outside the declared window must be handled by user coding.

The default values for the two commands are SET DEFCENT=19, SET YRTHRESH=00. When you provide a year threshold, years greater than or equal to that value assume the value assigned by DEFCENT. Years lower than that threshold become DEFCENT plus 1.

To see how DEFCENT and YRTHRESH are applied to interpret 2-digit years, consider the following:

```
SET DEFCENT=19, SET YRTHRESH=80
```

This set of commands describes a range from 1980 to 2079. If a 2-digit year field contains the value 99, then the server interprets the year as 1999. If the year field is 79, then the year is interpreted as 2079. If the year field is 00, then the year is interpreted as 2000.

Master File Syntax

How to:

Add Cross-Century Date Settings at the File Level

Add Cross-Century Date Settings at the Field Level

Add Cross-Century Dates Using a DEFINE Command

Example:

Implementing Cross-Century Dates

Instead of using SET commands, you can include settings at the file level in a Master File, or at the field level in a Master File.

Syntax: How to Add Cross-Century Date Settings at the File Level

The FDEFCENT syntax is

```
{FDEFCENT|FDFC}=nn
```

where:

nn

Is 19, unless otherwise specified.

The FYRTHRESH syntax is

```
{FYRTHRESH|FYRT}=nn
```

where:

nn

Is zero, unless otherwise specified.

Syntax: How to Add Cross-Century Date Settings at the Field Level

At the field level, DEFCENT and YRTHRESH can be added. The DEFCENT syntax is

```
{DEFCENT|DFC}=nn
```

where:

nn

Is 19, unless otherwise specified.

The YRTHRESH syntax is

```
{YRTHRESH|YRT}=nn
```

where:

nn

Is zero, unless otherwise specified.

Syntax: How to Add Cross-Century Dates Using a DEFINE Command

```
DEFINE FILE EMPLOYEE
 fld/fmt [{DEFCENT|DFC} nn {YRTHRESH|YRT} nn] [MISSING...]=expression;
END
```

The DFC and YRT syntax must follow the field format information.

Example: Implementing Cross-Century Dates

The following example illustrates how century interpretation is implemented at both the file level and field level in a Master File.

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDEFCENT=20, FYRTHRESH=66,$
 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, DEFCENT=19,
YRTHRESH=75,$
```

The next example illustrates the conversion of a 2-digit year field with the DEFINE command:

```
DEFINE FILE EMPLOYEE
ESHIRE_DATE/YYMD = HIRE_DATE; (The format of HIRE_DATE is I6YM.)
ESHIRE DFC 19 YRT 80 = HIRE_DATE;
END
```

Metadata Services With SQLENGINE SET

In this section:

When the server is dedicated to accessing one Relational Database Management System (RDBMS) using the SET SQLENGINE command in the global profile, metadata calls to the server are processed against the native catalogs of the RDBMS.

How Applications Access Metadata

The metadata procedures used by applications to query the native catalog directly are:

- For an ODBC-enabled application, ODBC SQL calls.
- For an API-enabled application, EDARPC for ODBC*xxxx* calls.

The following table shows the relationship between the ODBC call and the API call:

ODBC Call	API Call
SQLTables	ODBCTABL
SQLColumns	ODBCCOLS
SQLPrimaryKeys	ODBCPKEY
SQLStatistics	ODBCSTAT
SQLProcedures	ODBCPROC
SQLProcedureColumns	ODBCPRCC
SQLSpecialColumns	ODBCSCOL
SQLColumnPrivileges	ODBCCPRV
SQLForeignKeys	ODBCFKEY
SQLTablePrivileges	ODBCTPRV

You can use the following two commands to control or override table and column metadata calls:

```
SQL sqlengine SET ODBCCOLSSORT
```

```
SQL sqlengine SET ODBCTABL
```

You can include these commands in any of the supported server profiles.

Obtaining Column Information (DB2 only)

When you issue either the SQLColumns or ODBCCOLS Metadata call from a client application, by default, the server requests the columns from DB2 in *colno* order.

Syntax: How to Request the Sort Order of Column Data

```
SET ODBCCOLSSORT {ON|OFF}
```

where:

`ON`

The column data is requested from the server with order by colno. This value is the default.

`OFF`

The column data is returned in unsorted order.

Obtaining a User-Defined Metadata

How to:

Request User-Defined Metadata

Example:

Returning a List of Tables

When a client application issues either an ODBC or API metadata call, the server will run internal procedures that issue default SQL against the native RDBMS catalogs. You can issue your own SQL to run in place of the default SQL. You can specify this type of override for any of the internal server routines that deal with metadata.

Syntax: How to Request User-Defined Metadata

```
SQL sqlengine SET ODBCxxxx procname
```

where:

`ODBCxxxx`

Is the internal server routine name. Possible values are: ODBCTABL, ODBCCOLS, ODBCPKEY, ODBCSTAT, ODBCPROC, ODBCPRCC, ODBCSCOL, ODBCCPRV, ODBCFKEY, and ODBCTPRV.

`procname`

Is the procedure to run when the server receives the metadata call. This procedure must be available through the FOCEXEC ddname allocation in the server JCL.

Note: If this override procedure is used on a server running under MVS, it will add approximately 600K of storage above the line for each user.

When coding an override procedure, care must be taken to maintain the select list (answer set layout). The select list must conform to the ODBC specification for the return from the SQLTables call. See the *ODBC 2.0 Programmer's Reference and SDK Guide* for the SQLTables specification layout. Also, when using this override procedure, the parameters that are sent with the Metadata call need to be parsed correctly.

The override procedure should take the following format:

1. Dialogue Manager code to parse metadata call parameters.

2.
   ```
   SQL sqlengine
   SELECT code;
   ```

3.
   ```
   TABLE
   ON TABLE PCHOLD FORMAT ALPHA
   END
   ```

Items 1 and 2 above are user coded; item 3 must always be present at the end of the procedure as coded above.

Example: Returning a List of Tables

The following sample code found in *qualif*.EDARPC.DATA(DB2ODBC1) returns a list of tables that the connected user is authorized to INSERT, UPDATE, DELETE, and SELECT. It is a sample of how to code a DB2 override procedure for ODBCTABL. It is one of several ways to code SQL to return an answer set for the ODBCTABL call. You can code any relevant query as long as the SELECT list is maintained.

In a server profile, issue SQL DB2 SET ODBCTABL DB2ODBC1, and then execute the following RPC request on the server:

```
ODBCTABL   ,<NULL>,,,,0,0,*
```

The following example matches the above ODBCTABL call:

```
-*
-* Dialogue Manager code to parse the ODBCTABL parameter list
-*
-DEFAULTS 1=' ',2='%',3='%'
-IF &2 NE '<NULL>' THEN GOTO LAB1;
-SET &2 = '%';
-LAB1
-IF &3 NE ' ' THEN GOTO LAB2;
-SET &3 = '%';
-LAB2
-*
-* SQL SELECT code
-*
SQL DB2
SELECT ' ',T2.CREATOR,T2.NAME,'TABLE',' '
FROM SYSIBM.SYSTABAUTH T1,SYSIBM.SYSTABLES T2
WHERE T1.GRANTEE  = USER
AND   T1.TTNAME   LIKE '&2'
AND   T1.TCREATOR LIKE '&3'
AND   T2.TYPE     = 'T'
AND  (T1.DELETEAUTH IN  ('G','Y')
OR    T1.INSERTAUTH IN  ('G','Y')
OR    T1.SELECTAUTH IN  ('G','Y')
OR    T1.UPDATEAUTH IN  ('G','Y'))
AND   T1.TTNAME   = T2.NAME
AND   T1.TCREATOR = T2.CREATOR
UNION
SELECT ' ',T2.CREATOR,T2.NAME,'VIEW',' '
FROM SYSIBM.SYSTABAUTH T1,SYSIBM.SYSTABLES T2
WHERE T1.GRANTEE  = USER
AND   T1.TTNAME   LIKE '&1'
AND   T1.TCREATOR LIKE '&2'
AND   T2.TYPE     = 'V'
AND  (T1.DELETEAUTH IN  ('G','Y')
OR    T1.INSERTAUTH IN  ('G','Y')
OR    T1.SELECTAUTH IN  ('G','Y')
OR    T1.UPDATEAUTH IN  ('G','Y'))
AND   T1.TTNAME   = T2.NAME
AND   T1.TCREATOR = T2.CREATOR
ORDER BY CREATOR,NAME;
-*
-* The following code must always be present
-*
TABLE
ON TABLE PCHOLD FORMAT ALPHA
END
```

Maintaining Upward Compatibility

In Version 4.3.x or earlier, an Extended Catalog (SYSOWNER table) was created at installation time. This provided additional control to the list of tables returned with the SQLTables or ODBCTABL call. In Version 5.1.0 and higher, this table is no longer created. If you need to use this table from a previous version of the server to continue to have control over the list of tables, issue the following in any supported server profile:

```
SQL sqlengine SET OWNERID ownerid
```

where:

sqlengine

Indicates the data source. You can omit this value if you previously issued the SET SQLENGINE command.

ownerid

Identifies the creator or owner of the Extended Catalog table SYSOWNER. If this command is used, the SQL generated to provide a list of tables will be limited by the owner names in the SYSOWNER table.

This command is only supported for customers who have configured a Relational Gateway in previous releases of the server and want to continue with the same configuration.

Additional Master File Attributes

In this section:

These topics describe how to modify your server and communications configuration files.

The following attributes enable you to provide descriptive information about tables and columns.

`REMARKS`

Is an optional attribute for documenting tables.

`DESCRIPTION`

Is an optional attribute for documenting columns.

`TITLE`

Is an optional attribute for supplying an alternate column title on a report to replace the FIELDNAME value, which is normally used.

For the server, client tools using the API or ODBC will not use the TITLE attribute. The TITLE attribute is available only when directly querying the server catalog.

Documenting a Table

The REMARKS attribute enables you to document a table.

Syntax: How to Document a Table

The syntax is

```
REMARKS=text ,$
```

where:

`text`

Is one line of text. It can consist of a maximum of 78 characters. If the text contains a comma, you must enclose the text in single quotation marks.

The REMARKS attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Example: Documenting an Oracle Table

The following example shows how to document the Oracle table SAMPLE.

```
FILE=SAMPLE,SUFFIX=SQLORA,REMARKS=This is a sample Oracle table.,$
```

Documenting a Column

The DESCRIPTION attribute enables you to provide a comment for a column in a table.

You can also add documentation to a column declaration—or a segment or table declaration—by typing a comment 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. For the server, a comment added in this manner will not be available to any client application.

If you are using DEFINE fields in a Master File, you must place the DESCRIPTION attribute on a line by itself. The semicolon after the DEFINE must appear on the same line as DESCRIPTION=.

Syntax: How to Document a Column

The syntax is

```
DESC[RIPTION]=text ,$
```

where:

text

Is one line of text. It can consist of a maximum of 44 characters.

If the text contains a comma, you must enclose the text in single quotation marks.

The DESCRIPTION attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Note: Whenever possible, place the description on the same line with the attributes FIELDNAME and ALIAS, to conserve space.

Example: Documenting a Column

The following example shows how to provide a description for the column UNITS. The single quotation marks are required because the description contains a comma.

```
FIELD=UNITS,ALIAS=QTY,FORMAT=I6,DESC='This is quantity sold, not
returned',$
```

Example: Documenting a DEFINE Field

The following example shows how to provide a description for the DEFINE field ITEMS_SOLD.

```
DEFINE ITEMS_SOLD/D8=ORDERED-INVENTORY
;DESC=DAMAGED ITEMS NOT INCLUDED,$
```

Supplying an Alternate Column Title

How to:

Change the Default Column Title

Example:

Changing the Default Column Title

Changing the Default Column Title to a Two-line Column Title

Controlling the Underline for a Column Title

Specifying a Column Title for a DEFINE Field

When you generate a report, each column title in the report defaults to the name of the column as it appears in the table. However, you can change the default column title by specifying the TITLE attribute.

For the server, client tools using the API or ODBC will not use the TITLE attribute. The TITLE attribute is available only when directly querying the server catalog, or when using WebFOCUS (Windows version).

You can override both the FIELDNAME and TITLE attributes with AS phrases in your report request. To override an existing TITLE attribute, use the SET TITLE command. To control whether the TITLE attribute is propagated in the Master File of a HOLD file, use the SET HOLDATTR command.

The TITLE attribute has no effect in a report if the column is used with a prefix operator such as AVE. You can supply an alternate column title for columns with prefix operators by using the AS phrase.

If you are using DEFINE fields in a Master File, you must place the TITLE attribute on a line by itself. The semicolon after the DEFINE must appear on the same line as TITLE=.

Syntax: How to Change the Default Column Title

The syntax is:

```
TITLE='text' ,$
TITLE='text,text,...' ,$
TITLE='text   /' ,$
```

where:

text

Is any string that consists of a maximum of 64 characters.

You can split the text across as many as five separate lines. Use single quotation marks to delimit the text, and commas to divide the text into separate lines on the report output.

You can include blanks at the end of an alternate column title by entering a slash (/) in the last position that will be blank, followed by a closing single quotation mark.

The TITLE attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Example: Changing the Default Column Title

The following example shows how to replace the default column title, LNAME, with a title of Client Name.

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

Example: Changing the Default Column Title to a Two-line Column Title

The following example shows how to replace the default column title, LNAME, with a two-line column title of Client Name.

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

Example: Controlling the Underline for a Column Title

The following example shows how to control the length of the underline for an alternate column title.

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

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

Example: Specifying a Column Title for a DEFINE Field

The following example shows how to replace the default column title for the DEFINE field, ITEMS_SOLD, with a two-line column title of Items Sold.

```
DEFINE ITEMS_SOLD/D8=ORDERED-INVENTORY
;TITLE='Items,Sold',$
```

```
Items
Sold
-----
```

Specifying an Online Help Message

The HELPMESSAGE attribute enables you to specify an online help message for a field on a data entry screen. It is available only for FOCUS data maintenance applications.

When a Help Message Appears

How to:

Specify a Help Message

Example:

Displaying a Help Message to List Valid Values for a Column

Displaying a Help Message When a Format Error Occurs

Messages specified with the HELPMESSAGE attribute appear in the TYPE area of the CRTFORM when you:

- Enter a value for a database field that is invalid according to the criteria defined by the ACCEPT attribute.
- Enter a value for a database field that fails a VALIDATE test.
- Enter a value for a database field that causes a format error.
- Place the cursor in a data entry field and press a help key.

Regardless of the condition that triggers the display of the help message, the same message will appear.

Syntax: How to Specify a Help Message

The syntax is

```
HELP[MESSAGE]=text ,$
```

where:

text

Is one line of text. It can consist of a maximum of 78 characters.

You can use all characters and digits. If the text contains a comma, you must enclose the text in single quotation marks. Leading blanks are ignored.

The HELPMESSAGE attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Example: Displaying a Help Message to List Valid Values for a Column

The following example shows how to display a help message when the DEPARTMENT value is other than MIS, PRODUCTION, or SALES. DEPARTMENT has an ACCEPT attribute which tests values entered for it.

```
FIELDNAME=DEPARTMENT,ALIAS=DPT,FORMAT=A10,
   ACCEPT=MIS PRODUCTION SALES,
   HELPMESSAGE='DEPARTMENT MUST BE MIS, PRODUCTION, OR SALES',$
```

If you enter a value other than MIS, PRODUCTION, or SALES for DEPARTMENT on a CRTFORM, both the default message and help message display on the screen:

```
DATA VALUE IS NOT AMONG ACCEPTABLE VALUES FOR DEPARTMENT
DEPARTMENT MUST BE MIS, PRODUCTION, OR SALES
```

Example: Displaying a Help Message When a Format Error Occurs

The following example shows how to display a help message when a format error occurs in HIRE_DATE. Note that the format for HIRE_DATE is integer.

```
FIELDNAME=HIRE_DATE,ALIAS=HDT,FORMAT=I6YMD,
HELPMESSAGE=THE FORMAT FOR HIRE_DATE IS I6YMD,$
```

If you enter alphanumeric characters for HIRE_DATE on a CRTFORM, both the default message and help message display on the screen:

```
FORMAT ERROR IN VALUE ENTERED FOR FIELD HIRE_DATE
THE FORMAT FOR HIRE_DATE IS I6YMD
```

CHAPTER 2

Using the Adapter for Adabas

Topics:

The Adapter for Adabas allows applications to access Adabas data sources. The adapter converts application requests into native Adabas statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the Adabas Environment

How to:

Prepare the Adabas Environment on Windows

Prepare the Adabas Environment on UNIX

Prepare the Adabas Environment on OpenVMS

Example:

Specifying the Adapter for Adabas on OpenVMS

Adabas environment variables must be set before you configure the server. Refer to the Software AG documentation for more information about required environment variables.

Procedure: How to Prepare the Adabas Environment on Windows

On Windows, the Adabas environment is set up during the installation of Adabas.

Procedure: How to Prepare the Adabas Environment on UNIX

1. Specify the Adabas database directories to access using the UNIX environment variable $ADADIR. For example:

```
ADADIR=/rdbms/sag/ada
export ADADIR
```

2. Specify the Adabas driver using the UNIX environment variable $ADALNK. For example:

```
ADALNK=/rdbms/sag/ada/v31135/adalnk.so
export ADALNK
```

3. Specify the version of the Adabas product using the UNIX environment variable $ADAVERS. For example:

```
ADAVERS=v31135
export ADAVERS
```

Procedure: How to Prepare the Adabas Environment on OpenVMS

When a site creates an Adabas ID, the software automatically generates a DCL setup script so users (and products such as iWay) can properly invoke the environment.

The specification for the location of the Adabas setup file is

```
$@SAG$ROOT:[adabas_root]LOGIN.COM
```

where:

`SAG$ROOT`

Is the disk on which Adabas is installed.

`adabas_root`

Is the root directory of the Adabas installation.

Example: Specifying the Adapter for Adabas on OpenVMS

```
$@SAG$ROOT:[Adabas]LOGIN.COM
```

The logicals ADABAS$MAIN, ADALNK, and ADABAS$VERSION will be used to communicate with Adabas Nucleus.

Configuring the Adapter for Adabas

In this section:

Declaring Connection Attributes

Overriding Default Passwords in Specific Files

How to:

Configure the Adapter From the Web Console Using Generic Parameters

Configure the Adapter From the Web Console Using OS/390 and z/OS Parameters

Reference:

Declaring Connection Attributes From the Web Console

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Adapter From the Web Console Using Generic Parameters

The following Adabas parameters are applicable to all platforms and must be entered into the input fields on the Add Adapter screen.

1. Specify the Adabas database number to access. For example:

   001

2. Specify the number of the Adabas Employee or another Demo File. For example:

   011

3. Select the level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default.

   If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure* after the parameters are entered. Configuration results will appear on the screen. The SET CONNECTION_ATTRIBUTES command is inserted into the edasprof.prf file. For example,

   ```
   ENGINE ADBSINX SET CONNECTION_ATTRIBUTES ;1:ADAEMPL=11
   ```

   Tip: Once the adapter is configured, the connection name appears in the navigation pane under *Configured*. If you left-click either the adapter name or the connection name, an appropriate set of options becomes available.

Procedure: How to Configure the Adapter From the Web Console Using OS/390 and z/OS Parameters

The following Adabas parameters are applicable to OS/390 and z/OS platforms only, and must be entered into the input fields on the Add Adapter screen.

1. Specify the Adabas router SVC number. For example:

   240

2. Specify the type of the device where the Adabas Associator is located. For example:

   3390

3. Specify the name of the Adabas source library that contains member ADALNK. For example:

   ADABAS.V713.SRCE

4. Specify the name of the Adabas Associator Data Set. For example:

   ADABAS.V713.EXAMPLE.DB001.ASSOR1

5. Click *Configure* after the parameters are entered. Configuration results will appear on the screen. The SET CONNECTION_ATTRIBUTES command is inserted into the edasprof.prf file. For example,

   ```
   ENGINE ADBSINX SET CONNECTION_ATTRIBUTES
   ;3:"ADAEMPL=1,ADASVC=240,ADADEVICE=3390,
   ADAASSO=ADABAS.V713.EXAMPLE.DB001.ASSOR1"
   ```

   Tip: Once the adapter is configured, the connection name appears in the navigation pane under *Configured*. If you left-click either the adapter name or the connection name, an appropriate set of options becomes available.

Reference: Declaring Connection Attributes From the Web Console

Attribute	Description
Default DB number	Number of the database to access. (Required for all platforms.)
SAG Employee/Demo file number	Number of the Adabas Employee or another Demo File. (Required for all platforms.)
SVC number	SVC number of the Adabas router. (Required only for OS/390 and z/OS.)
Device type	Type of device where the Adabas Associator is located. (Required only for OS/390 and z/OS.)
ADABAS Source Library name	Name of the Adabas source library that contains member ADALNK. (Required only for OS/390 and z/OS.)
Associator data set name	Name of the Adabas Associator data set. (Required only for OS/390 and z/OS.)

Declaring Connection Attributes

No user ID and password are needed to establish a connection to Adabas. This means that Trusted Mode is the standard for Adabas. To secure a connection, use the SET PASSWORD feature.

Overriding Default Passwords in Specific Files

How to:

Use the SET PASSWORD Command

Example:

Using the SET PASSWORD Command to Specify Files in a Database

You can set passwords for Adabas files in the server profile using the SET PASSWORD command. You can set default passwords for all files and/or databases. Specific passwords can be set for specific files which will override the default. The SET command overrides the password coded in the Access File. The SET PASSWORD command remains valid throughout the user's session.

Syntax: How to Use the SET PASSWORD Command

```
ENGINE ADBSINX SET PASSWORD password FILENO ALL DBNO {ALL|dbno}
ENGINE ADBSINX SET PASSWORD password FILENO fileno DBNO dbno
ENGINE ADBSINX SET PASSWORD OFF
ENGINE ADBSINX SET PASSWORD DEFAULT
```

where:

`ADBSINX`

Indicates the Adapter for Adabas.

`password`

Is the password, which can be from one to eight characters in length.

`FILENO`

Specifies the file number for which the password is set.

`DBNO`

Specifies the database or databases for which the password is set.

`ALL`

Indicates all files and/or databases used with the FILENO and DBNO parameters. If you want to use ALL for both FILENO and DBNO, issue this command before any other subsequent password commands. ALL overrides any prior settings.

`dbno`

Is any valid numeric database value between 0 and 255 (65,535 is the maximum value on a mainframe platform). The DBNO parameter indicates the actual database number.

`fileno`

Provides a file number, a list of file numbers, and/or a range of file numbers used with the FILENO parameter. Numbers and ranges can be combined by separating items with commas. Valid file numbers range between 0 and 255 (5000 is the maximum value on a mainframe platform).

`OFF`

Clears all previous ADBSINX SET PASSWORD commands. The Adapter for Adabas does not use any passwords specified in the Access File. This command lets you access only those files that have no password security.

`DEFAULT`

Clears all previous ADBSINX SET PASSWORD commands and causes the Adapter for Adabas to use the password in the Access File.

To set a default password:

- Issue the SET PASSWORD command using the ALL keyword for FILENO and/or DBNO.
- Issue specific password requests by specifying particular file and/or database numbers (or ranges) in a subsequent SET PASSWORD command.

Note that subsequent requests are appended to previous requests. Changes are cumulative. The passwords can be issued from a FOCEXEC, which may be encrypted.

Example: Using the SET PASSWORD Command to Specify Files in a Database

To set the password to BRUCE for specific files in database number 123 on OS/390 and z/OS, issue the following command:

```
ENGINE ADBSINX SET PASSWORD BRUCE FILENO 1,3-5,23,89-93 DBNO 123
```

To clear all previous ADBSINX SET PASSWORD commands on OS/390 and z/OS, causing the adapter for Adabas to use the password in the Access File, issue the following command:

```
ENGINE ADBSINX SET PASSWORD DEFAULT
```

Adabas Overview

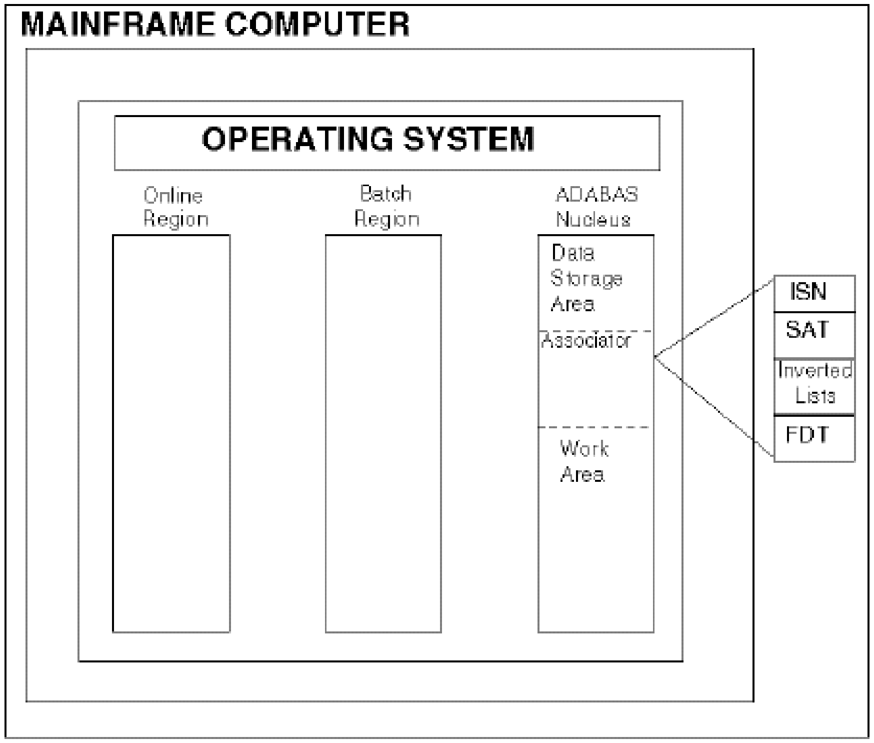
In this section:

This topic provides an overview of Adabas and Adabas files, and includes a discussion of server concepts.

Adabas is a field-oriented DBMS. Data retrievals and updates are performed on a field-by-field basis.

Since Adabas data retrieval occurs at the field level rather than at the record level, your applications may be designed without consideration for the physical organization and maintenance of the record. Data is accessed in a variety of ways (in physical sequence, in logical sequence, or by Internal Sequence Numbers), thereby enabling you to tailor data access to address your specific needs.

The Adabas DBMS consists of several components. The following is an illustration of the Adabas environment.



The following are the components of the Adabas environment:

Component	**Description**
Operating System	Set of programs (software) that control the operation of the computer (hardware).
Online Region	Part of the Dynamic Area of the computer. It is the section in which multiple jobs execute simultaneously. The Online Region is also known as the Foreground Region (applies only to TSO).
Batch Region	Part of the Dynamic Area of the computer. Jobs become a series of commands that are grouped together and processed in batches (applies only to batch jobs).
Data Storage Area	Contains the actual data. The data is stored in compressed form.
Associator	Stores relationships about the data. It contains the following components: • Internal Sequence Number (ISN) list, also called the Address Converter, used to determine the physical location of data. • Space Allocation Tables (SATs) to control storage space. • Inverted lists for the defined descriptors. Inverted lists and descriptors are discussed in *Inverted Lists: The Adabas Key Structure* on page 2-11. • Field Definition Tables (FDTs), which contain detailed data structure information for each field and descriptor.
Work Area	Is a "scratch pad" used by Adabas to build ISN lists and to sort and store records that your program requires.

The Multi-Programming Module (MPM), the cornerstone of the Adabas Nucleus, contains the logic that enables multiple programs, both batch and online, to access Adabas databases simultaneously.

Each database has its own data storage area, associator, and work area, in addition to its own unique database number.

Adabas Files

When accessing Adabas, the server uses logical files as the unit of data retrieval. An Adabas file is identified by a file number. The number is unique within each database.

All file and field documentation for the Adabas database can be stored in the Predict dictionary. The information about the data structure includes processing uses, file ownership, and field descriptions.

Inverted Lists: The Adabas Key Structure

An Adabas file is defined with a set of indexes called descriptors. The Adabas term for these values, which reside in the File Associator Table, is inverted lists. Inverted lists contain the values of descriptors, a count of the total number of records in which each value appears, and the Internal Sequence Numbers (ISNs), in ascending order, associated with each occurrence.

Adabas utilities create and maintain an inverted list for each descriptor identified. These lists store data in an ascending sequence by the value of the key. A single file may have up to 256 inverted lists associated with it (the number of inverted lists is unlimited for a mainframe platform).

The descriptors are identified to speed data location and to retrieve specific, frequently required data from Adabas files. Of the available types of inverted lists, the Adapter for Adabas supports the following:

Inverted List	**Description**
Descriptors	Key values associated with a single field (also called elementary field descriptors).
Subdescriptors	Key values associated with part of a single field.
Superdescriptors	Key values associated with all or part of two to twenty fields.
Phonetic descriptors	Alphabetic values associated with the first twenty bytes of the field value.
Hyperdescriptors	A value generated, based on a user-supplied algorithm.

In the following text, descriptor refers to all five types of descriptors interchangeably, unless otherwise indicated.

Two or more files may be related in these ways:

- The database administrator may couple several files based on a common descriptor key in each file.
- An application program may logically relate files if there are fields in one file that can be used to access a key in another.

Field Definition Tables

Reference:

Standard Formats for Adabas Fields

Field description information for Adabas files is maintained in the FDT, which is part of the Adabas associator. Adabas uses this information when accessing files. The FDT includes the following information:

Field Information	**Description**
Field Level Indicator	Denotes a hierarchical relationship between fields. In an Adabas record description the levels are 1, 2, 3, and so on.
Adabas Field Name	Two-character name used by Adabas to identify the field. This name is unique to the file. The first character must be alphabetic, and the second character can be alphabetic or numeric. Field names E0 through E9 are reserved for internal Adabas use.
Standard Length	Length of the field in bytes.
Standard Format	Format of the field. Refer to the table in *Standard Formats for Adabas Fields* on page 2-13 for the acceptable formats and their meanings.
Field Properties	Indicates whether the field value is null suppressed (NU), the field is of fixed or variable length (FI), or if the field (which exists on a mainframe platform) is long alphanumeric (LA).
Descriptor Status	Indicates if the field is a descriptor, subdescriptor, superdescriptor, phonetic descriptor, hyperdescriptor, subfield, or superfield.
Field Type	Indicates if a field is a group (GR), multi-value (MU), or periodic group (PE). Multi-value fields and periodic groups are examples of multiply occurring fields.

Typically, an Adabas FDT contains some, but not all, of the field characteristics discussed above. To view the FDT, Software AG provides the ADAREP report. See your Software AG documentation for more information on how to run the ADAREP report.

The external field name is in the Predict dictionary.

External Field Name	32-byte name used in a Predict to identify the field. This name is unique to the file.

Reference: Standard Formats for Adabas Fields

The following table shows the acceptable standard formats for Adabas fields and their meanings:

Format	**Description**
`A`	Alphanumeric data field with a maximum of 253 bytes (126 for descriptor fields).
`U`	Zoned decimal data field with a maximum of 29 bytes (signed, unpacked data).
`P`	Signed packed decimal data field with a maximum of 15 bytes.
`B`	Unsigned binary data field with a maximum of 126 bytes.
`F`	Single-precision, floating point data field that is always four bytes long.
`G`	Decimal, double-precision integer field with an eight-byte maximum.

In the following example, the column on the left illustrates a partial Adabas FDT. The bold numbers on the left refer to the numbered annotations that follow:

```
   PAY-FILE

   FIELD DESCRIPTION (from FDT)          FIELD NAME (from DDM)

1. 01, PS, 08, P, NU, DE                 SSN
2. 01, PW, 04, P                         HOURLY_WAGE
3. 01, MI, PE                            MONTHLY_INFO
   02, MH, 04, P                         MONTHLY_HOURS
   02, MW, 04, P                         MONTHLY_WAGES
   02, MT, 04, P                         MONTHLY_TAX
4. 01, PT, 04, P, DE, MU                 TAX
   01, PC, 08, P, DE, MU                 CHILD_SSN
```

1. The first field, SSN, is an elementary-level field. Its internal name is PS, it is eight bytes long, and it is in packed decimal data format. The field is null suppressed, as indicated by the NU, and it is a descriptor (DE).
2. HOURLY_WAGE is also an elementary-level field. It is called PW internally, is four bytes long, and is also in packed decimal data format.
3. MONTHLY_HOURS, MONTHLY_WAGES, and MONTHLY_TAX are all subordinate fields of the MONTHLY_INFO periodic group (PE). Internally, they are called MH, MW, and MT, respectively. Each is four bytes long and is in packed decimal data format.
4. TAX and CHILD_SSN are elementary-level fields. TAX is called PT internally, is four bytes long, is in packed decimal data format, and is a multi-value (MU) field with a descriptor (DE) associated with it. CHILD_SSN is called PC internally, is eight bytes long, is in packed decimal format, and is also a multi-value field with a descriptor associated with it.

For more information about Adabas periodic groups (PE) and multi-value (MU) fields, see *Mapping Adabas Files With Variable-length Records and Repeating Fields* on page 2-68.

Managing Data Storage With Null-Suppression

Null-suppression is one of the methods Adabas employs to manage data storage. Only the fields within a record that contain data are stored. Fields containing blanks for alphabetic characters or zeros for numeric data are not stored. They are represented by a one-byte "empty field" character. When a procedure accesses a record containing a null-suppressed field, Adabas expands the field to its full length and includes the null values of blanks or zeros.

When the null-suppressed field is a descriptor or part of a superdescriptor, no entry is made for the record containing that field in the inverted list. It would not be productive to have the inverted list direct you to records in which the descriptor has no data.

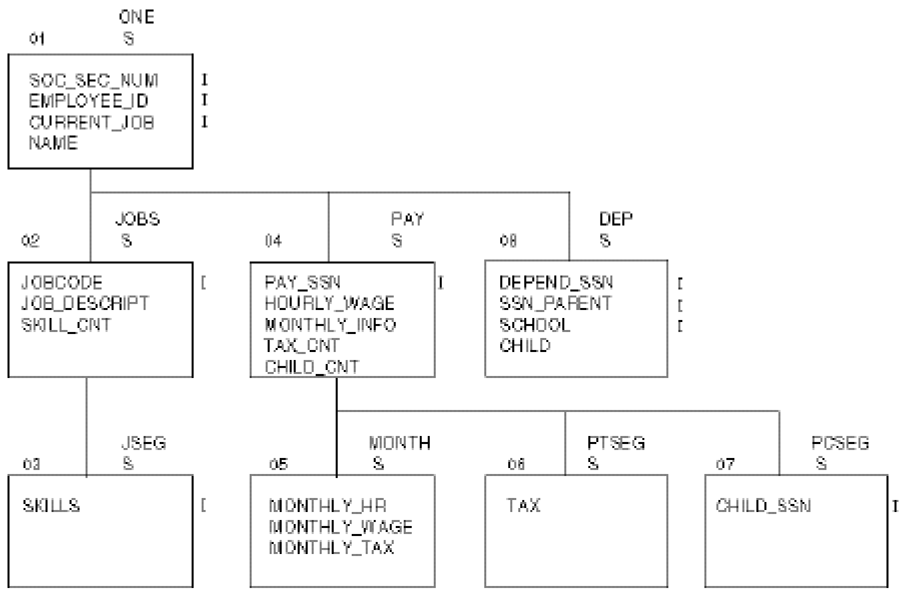
A field with null-suppression appears with the NU attribute in the Adabas FDT.

Note that for the iWay server to support NULL (MISSING = ON) for an Adabas field, you must define the field in the Adabas FDT with the NC attribute.

Server File Structure

The server uses a non-procedural language to create reports, graphs, and extract files. It also enables you to access data sources without knowing the details of the file structure or access method.

The server treats any data source as either a single-path or multi-path hierarchy. Graphically, information is laid out using an inverted tree structure, as in the following sample STAFF file. In the example, the letter I to the right of a field indicates an indexed field.



The most general information appears at the top, and the more specific information appears under it. Each box in the structure is referred to as a segment. A database can consist of one or more logically related segments.

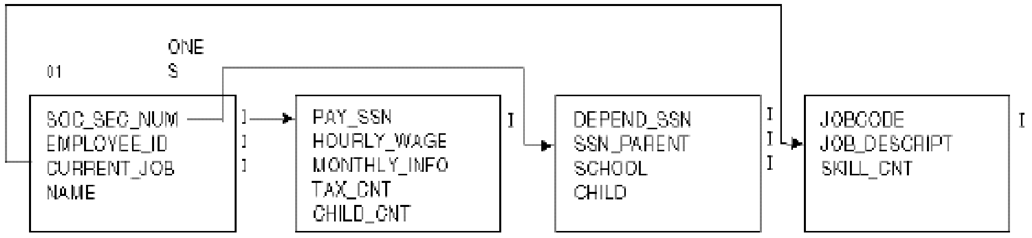
When logically related information is retrieved using this database structure, multiple occurrences of each segment are created. Each occurrence of a segment is called a segment instance. Each Adabas database is equivalent to a collection of logically related segment instances.

The retrieval sequence of the segments is determined by the view of the structure, that is, the order of the segments from top to bottom and left to right. The segment at the top is the parent, or root, segment. The segments under the parent are the child, or descendant, segments.

Generally, parent and child segments have a one-to-many relationship; that is, a single parent has multiple occurrences of one child segment. However, the server also handles one parent-one instance only of a child segment.

Adabas uses specific concepts and techniques for organizing data. It holds data in logically distinct files that are interrelated using fields that share common formats and values called descriptors.

Within a single Adabas file, records that contain multiple occurrences of a field can vary in length and format. The following example illustrates an Adabas structure containing four separate files that are linked by common fields:



Many Adabas structures are more complex than the preceding example.

Adabas Descriptors

Descriptors are fields, partial fields, or groups of complete and partial fields used by Adabas to select records in a file. Adabas descriptors correspond to indexed fields.

There are five types of descriptors supported by the Adapter for Adabas:

- Descriptors (DE), which have a value associated with a single field.
- Superdescriptors (SPR), which have a key value associated with all or part of two to twenty fields (see your Software AG documentation for current limitations on superdescriptors).
- Subdescriptors (NOP), which have a key value associated with part of a single field.
- Phonetic Descriptors (PDS), which have a similar phonetic value associated with a single field. The phonetic value of a descriptor is based on the first twenty bytes of the field value. Only alphabetic values are considered: numeric values, special characters, and blanks are ignored. Lowercase and uppercase alphanumeric characters are internally identical.
- Hyperdescriptors (HDS), which have a value generated, based on a user-supplied algorithm.

All five types of descriptors are collectively referred to as "descriptors."

Consider the Adabas FDT for the STAFF file:

```
                          STAFF FILE FDT
FIELD DESCRIPTION (from FDT)             FIELD NAME (from Predict)
   01, ES, 08, P, NU, DE                 SOC_SEC_NUM
   01, EI, 06, A, NU, DE                 EMPLOYEE_ID
   01, EJ, 03, A, DE                     CURRENT_JOB
   01, EN, 24, A                         NAME
```

Notice that SOC_SEC_NUM, EMPLOYEE_ID, and CURRENT_JOB are labeled as descriptors. Any one of these three fields can be used to search the STAFF file.

Adabas descriptors, superdescriptors, and subdescriptors must be declared in Master and Access Files in certain ways. For more information, see *Mapping Adabas Descriptors* on page 2-59.

Adabas Files With Fixed-length Records

Unless an Adabas file has multi-value fields or periodic groups, each field occurs once in each record. When describing an Adabas file, you do not need to describe all the fields in your Master File; just the fields you use. Note that if you choose a periodic group, all fields in the periodic group must be defined. A single simple Adabas file maps to a single segment on the server, and each field you use in the Adabas file becomes a field in the segment.

Adabas RECORD		**SEGMENT**	
DB	**Name (in Predict)**	**Field Name**	**Alias**
ES	SOC_SEC_NUM	SOC_SEC_NUM	ES
EI	EMPLOYEE_ID	EMPLOYEE_ID	EI
EJ	CURRENT_JOB	CURRENT_JOB	EJ
EN	NAME	NAME	EN

Adabas Files With Variable-length Records and Repeating Fields

There are two types of Adabas repeating fields:

- Multi-value (MU) fields.
- Periodic (PE) groups.

An MU field is a single field that occurs a variable number of times in a record. It appears as type MU in an Adabas record description. Adabas supports up to 191 occurrences of MU fields per record.

A PE group is a group of contiguous fields that occur a variable number of times in a single record. It appears as type PE in an Adabas record description. Adabas supports up to 191 occurrences of PE fields per record. These component fields are regular fields or MU fields.

Note: The Adapter for Adabas supports the maximum number of occurrences of MU fields and PE groups allowed by Software AG.

When an Adabas file contains MU fields or PE groups, the repeating information occurs a variable number of times. The physical record length varies depending on how many times the repeating information actually appears.

More than one segment is needed to accurately describe an Adabas file with repeating data. The number of times the fields repeat is expressed in a counter field. For more information, see *Mapping Adabas Files With Variable-length Records and Repeating Fields* on page 2-68.

Managing Adabas Metadata

In this section:

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Adabas data types.

For the Adapter for Adabas to access Adabas files, you must describe each file you use in a Master and Access File. The logical description of an Adabas file is stored in a Master File, which describes the field layout.

Creating Synonyms

Synonyms define unique names (or aliases) for each Adabas table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File, which represent the server's metadata.

The CREATE SYNONYM command automatically generates Master and Access Files for Adabas files based on information stored in the Field Definition Table (FDT) in Adabas and the Predict dictionary (optionally). The command requires Adabas Release 5.0 or higher and Predict Release 3.1.4 or higher.

The syntax is identical on UNIX, Windows, OS/390 and z/OS, and OpenVMS platforms.

Note: You can add the following parameters to the CREATE SYNONYM syntax:

- ISN provides support for retrieval of Adabas internal sequence numbers.
- GFBID allows the use of Adabas Global Format buffers for requests that utilize the same field lists.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The Select Synonym Candidate for Adabas pane (Step 1 of 2) opens.

4. Enter values for the following parameters:

Parameter	**Description**
`Default DB number`	Number of the database to access. (Required for all platforms.)
`Associator Data Set name`	Name of the Adabas Associator data set. (Required only for OS/390 and z/OS.)

5. Click the Use Predict Metadata check box to if you want to use Predict™ software that has been installed for use with Adabas.

6. Click *Select Tables*. The Create Synonym for Adabas pane (Step 2 of 2) opens.

7. Provide values for the following superdescriptior processing parameters:

Parameter	**Description**

`Mode`	`Extended`
	Describes superdescriptors as groups in the Master File. Neither the groups nor their components (fields) have field names specified in the Master File. Aliases are populated using the Adabas FDT (2-byte name).
	The Adapter for Adabas uses new logic to process the selection criteria of requests involving that synonym. Screening conditions within the request are analyzed and the superdescriptor that covers the highest order component fields required by the selection criteria are used. For more information, see *NEW Synonym Setting for Superdescriptors* on page 2-79.
	Note that you must specify CALLTYPE=RL in the Access File in order to take advantage of superdescriptor-based access.
	`Standard`
	Incorporates the usual CREATE SYNONYM behavior, which does not trigger the new logic for requests using superdescriptors. Standard is the default value.

`ISN field`	`Yes`
	Specifies that a field generated in the Master File is to be used to specify Adabas Internal Sequence Numbers (ISN).
	`No`
	Indicates that Adabas Internal Sequence Numbers (ISN) are not to be included in the Master File.

`GSBID field`	`Yes`
	Specifies that a field generated in the Master File is to be used to specify Adabas Global Format Buffer ID values.
	`No`
	Indicates that Global Format Buffer ID values are not to be included in the Master File.

8. Enter the following additional parameters:

Parameter	**Description**
Select Application	Select an application directory. The default value is baseapp.
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

9. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

10. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

11. Click *Create Synonym*. The result of the Create Synonym process is a Master File, which is created and added under the specified application directory.

12. A status window displays the message:

```
All Synonyms Created Successfully
```

From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synoynm
FOR [FILE=]file-number[/predict-file-number[(predict-filename)]]
DBMS ADBSINX
DATABASE dbid[/predict-dbid]
PARMS '<parameter_list>'
[CHECKNAMES][UNIQUENAMES]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is the alias for the data source (maximum 64 characters).

file-number

Is the number of the Adabas file used as the base for the Master and Access Files. The range is from 1 to 255 (5000 is the maximum value on a mainframe platform).

predict-file-number

Is the number of the Adabas file where the Predict dictionary data is located. The range is from 1 to 255 (5000 is the maximum value on a mainframe platform).

predict-filename

Is the name of the view in the Predict dictionary that corresponds to the Adabas file being used. Use this name when it differs from *synonym*. Maximum length is 32 bytes.

ADBSINX

Is the SUFFIX that indicates the Adapter for Adabas.

dbid

Is the Database ID (DBNO) for the Adabas DBMS to be accessed. The range is from 1 to 255 (65535 is the maximum value on a mainframe platform).

predict-dbid

Is the Database ID (DBNO) for the Adabas DBMS that contains the Predict dictionary data. The range is from 1 to 255 (65535 is the maximum value on a mainframe platform).

parameter_list

Are the parameter values delimited by space or comma. Possible values are:

GFBID specifies that a field generated in the Master File is to be used to specify Adabas Global Format Buffer ID values.

ISN specifies that a field generated in the Master File is to be used to specify Adabas Internal Sequence Numbers (ISN).

NEW or STD is the processing mode for Adabas superdescriptor fields. For more information, see *NEW Synonym Setting for Superdescriptors* on page 2-79.

CHECKNAMES

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

`UNIQUENAMES`

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

Note: These values override values previously set by SET commands. For more information, see *Customizing the Adabas Environment* on page 2-78.

Example: Creating Synonyms for Adabas With and Without Predict Existences

- **When the Predict file exists:** Issue CREATE SYNONYM with the name SAMPLE for file 2 from Database 1. Predict dictionary is located in file 12 from database 2. The name of the view from the Predict dictionary is VEHICLES:

```
CREATE SYNONYM SAMPLE FOR FILE=2/12(VEHICLES)
DBMS ADBSINX DATABASE 1/2
END
```

- **With the Predict file and the same file name:** Issue CREATE SYNONYM with the name SAMPLE for file 2 from Database 1. The name of the view from the Predict dictionary is the same as the Master and Access Files:

```
CREATE SYNONYM SAMPLE FOR FILE=2/11
DBMS ADBSINX DATABASE 1/1
END
```

- **Without the Predict file:** Issue CREATE SYNONYM as follows:

```
CREATE SYNONYM FILE12 FOR FILE=12
DBMS ADBSINX DATABASE 1
END
```

Below are samples of Master and Access Files generated in this example.

Generated Master File: file12.mas

```
FILE=FILE12 ,SUFFIX=ADBSINX ,$

SEGNAME=S01 ,SEGTYPE=S0 ,$
FIELD=AA_FIELD ,ALIAS=AA ,A15 ,A15 ,INDEX=I,$
FIELD=AB_FIELD ,ALIAS=AB ,I9  ,I4  ,  ,$
FIELD=AC_FIELD ,ALIAS=AC ,A8  ,A8  ,INDEX=I,$
GROUP=CD_GROUP ,ALIAS=CD ,A50 ,A50 ,  ,$
FIELD=AD_FIELD ,ALIAS=AD ,A20 ,A20 ,INDEX=I,$
FIELD=AE_FIELD ,ALIAS=AE ,A20 ,A20 ,  ,$
FIELD=AF_FIELD ,ALIAS=AF ,A10 ,A10 ,INDEX=I,$
FIELD=AG_FIELD ,ALIAS=AG ,P2  ,Z2  ,  ,$
FIELD=AH_FIELD ,ALIAS=AH ,A1  ,A1  ,INDEX=I,$
FIELD=AI_FIELD ,ALIAS=AI ,A1  ,A1  ,  ,$
FIELD=AJ_FIELD ,ALIAS=AJ ,P6  ,Z6  ,  ,$
```

```
$GRMU=AK_GROUP ,ALIAS=AK ,A7  ,A4  , ,$
FIELD=AL_FIELD ,ALIAS=AL ,A3  ,A3  , ,$
FIELD=AM0101_CNT ,ALIAS=AMC   ,I4  ,I1 , ,$
$ $$$$$$$$$$$$$$$$$$$$$$ Superdescriptor $$$$$$$$$$$$$$$$$$$$$$$$$$ $
FIELD=AN_FIELD ,ALIAS=AN ,A4  ,A4  ,INDEX=I,$
$ FIELD=AJ_FIELD_S01 ,ALIAS=AJ,P6  ,Z2 , ,$
$ FIELD=AJ_FIELD_S01 ,ALIAS=AJ,P6  ,Z2 , ,$
$ $$$$$$$$$$$$$$$$$$$$$$ Superdescriptor $$$$$$$$$$$$$$$$$$$$$$$$$$ $
GROUP=AO_GROUP ,ALIAS=AO ,A28 ,A22 ,INDEX=I,$
FIELD=AG_FIELD_S02 ,ALIAS=AG  ,P2  ,Z2 , ,$
FIELD=AD_FIELD_S02 ,ALIAS=AD  ,A20 ,A20 ,INDEX=I,$

SEGNAME=AM0101 ,SEGTYPE=S0, PARENT=S01 ,OCCURS=AMC,$
FIELD=AM_FIELD    ,ALIAS=AM   ,P7  ,P4  , ,$
FIELD=AM0101_OCC  ,ALIAS=ORDER,I4  ,I1  , ,$
```

Generated Access File: file12.acx

```
RELEASE=6  ,OPEN=YES                                                  ,$
SEGNAM=S01 ,ACCESS=ADBS ,FILENO=12   ,DBNO=1     ,WRITE=YES           ,
            UNQKEYNAME=AA_FIELD                                       ,
            CALLTYPE=FIND                                             ,$
$           CALLTYPE=RL  ,SEQFIELD=AA_FIELD                           ,$
$FIELD=AA_FIELD                              ,TYPE=DSC                ,$
$FIELD=AC_FIELD                              ,TYPE=DSC                ,$
$FIELD=AD_FIELD                              ,TYPE=DSC                ,$
$FIELD=AF_FIELD                              ,TYPE=DSC                ,$
$FIELD=AH_FIELD                              ,TYPE=DSC                ,$
 FIELD=AN_FIELD                              ,TYPE=NOP                ,$
 FIELD=AO_GROUP                              ,TYPE=SPR                ,$
  FIELD=AG_FIELD_S02                         ,TYPE=   ,NU=YES         ,$
  FIELD=AD_FIELD_S02                         ,TYPE=DSC,NU=YES         ,$
SEGNAM=AM0101   ,ACCESS=MU   ,FILENO=12     ,DBNO=1   ,WRITE=YES      ,$
```

Example: Creating a Synonym Using a Parameter List

Issue the following command to create Master and Access Files with ISN support and to process a superdescriptor in the NEW mode:

```
CREATE SYNONYM SAMPLE FOR 12
DBMS ADBSINX DATABASE 1
PARMS 'ISN, NEW'
END
```

Note:

- Only one Adabas file description can be generated in the Master File. If multiple Adabas files must be described in one Master File, a description for each Adabas file must be created and combined in one Master File manually.

- The CREATE SYNONYM command describes Adabas files within the constraints of the Adapter for Adabas. For more information, see *Platform-Specific Functionality* on page 2-77. Superdescriptors, subdescriptors, periodic elements (PE groups), and multi-value fields (MU) are included. Comments are used to describe unsupported fields in the generated Master File.
- The CREATE SYNONYM command allows you to customize the output using data from the Predict dictionary. You can:
  - Replace generic field names constructed based on Adabas field names (xx_FIELD) with the names obtained from the Predict dictionary.
  - Replace the USAGE format and length generated from an Adabas FDT (Field Definition Table) with the format and length from the Predict dictionary.
  - Include the NATURAL column heading stored in the Predict dictionary in the Master File for use as the default column heading in server reports.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '| ', '&', ' (', ') ', ' <', '> ', '' ', ' =', '''

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Reference: Master File Field Attributes

The following field attributes describe Adabas data segments.

Field Attribute	Description
`FILE`	Master File name. It may or may not match the file name in the Adabas DBMS.
`SUFFIX`	Always ADBSINX.
`SEGNAME`	Segment names in the description generated by CREATE SYNONYM. They follow a logical format to provide uniqueness within the file.
`FIELD`	Field name from the Predict dictionary (if used) or generated automatically (xx_FIELD).
`GROUP`	Identifies fields described as simple groups or PE groups. Is the field name from the Predict dictionary (if used) or generated automatically (xx_GROUP).
`ALIAS`	Actual Adabas short name. It can be detailed for counter fields without standard length. For order fields, it has the value ORDER.
`USAGE`	USAGE format and length of the field. This attribute determines how the value is displayed in reports. Values are determined based on ACTUAL format and length and then may be detailed by value from the Predict dictionary (if used). For fields without standard length, it has the maximum value for the format.
`ACTUAL`	Field standard format and length as stored in the Adabas Field Definition Table (FDT) for a given file. For fields without standard length, it has the maximum value for the format. For fields with format A and option LA (Long Alphanumeric, used only for OS/390 and z/OS) the ACTUAL length must have a value greater than 253. Value 500 is assumed.
`INDEX=I`	Indicates the field is a superdescriptor, subdescriptor, or simple descriptor.
`TITLE`	Column heading used in reports. This attribute is included only if the Predict dictionary is used and the field has a column heading value in the Predict dictionary.
`GROUP`	Identifies fields described as simple groups or PE groups.
`$GRMU`	Group that contains a PE group or an MU (multi-value) field.

Field Attribute	Description
`$2LONG`	Group field whose total length exceeds the maximum of 4096 characters supported by the server.
`$PEMU`	PE group that contains an MU (multi-value) field or is a group field that contains a PE group or an MU field. In both cases, the field cannot be used to retrieve data using the adapter.
`$FIELD`	Field that belongs to a commented group or superdescriptor composed of partial fields (NOP).

Reference: Access File Attributes

The following Access File attributes describe Adabas data segments:

Access File Attribute	Description
`RELEASE`	Adabas release. If RELEASE=5 or less, then FILENO can be 1–255 and DBNO can be 0–255. If RELEASE is 6 or greater, then the adapter supports two-byte FILENO (1–5000) and DBNO (1–65535).
`OPEN`	Determines whether the adapter should issue Adabas OPEN and CLOSE calls for each report request.
`SEGNAM`	Segment name as described in the Master File.
`ACCESS`	Access method for the segment. ADBS indicates segments that contain non-repeating data. PE indicates segments that describe periodic groups. MU indicates segments that describe multi-value fields.
`DBNO`	Adabas database number. On UNIX, Windows, and OpenVMS, this attribute must be included in the Access File. DBNO depends on RELEASE. If RELEASE is 5 or less, DBNO can be 0–255; if RELEASE is 6 or greater, DBNO can be 1–65535.
`FILENO`	Adabas file number. If RELEASE is 5 or less, FILENO can be 1–255; if RELEASE is 6 or greater, FILENO can be 1–5000.
`UNQKEYNAME`	Field name that will be used as a unique key during file updating. The first unique indexed field from the root segment is used. If that does not exist, then a value from SEQFIELD is used.

Access File Attribute	**Description**
`CALLTYPE`	FIND. Creates an additional commented line with CALLTYPE=RL if the Predict file view contains SEQFIELD data or the root segment contains an indexed field. The line with the preferable value of CALLTYPE can be chosen depending on the method used to retrieve the data from the database.
`SEQFIELD`	SEQFIELD name taken from the Predict file view, if used, or the first unique indexed field, or the first indexed field from Root segment. If CALLTYPE=FIND, then no value is entered for SEQFIELD in the Access File.
`FIELD`	Describes a descriptor, superdescriptor, subdescriptor, phonetic descriptor, hyperdescriptor, or a component of a superdescriptor.
`TYPE`	Type of descriptor for FIELD. Values are SPR for superdescriptors, NOP for subdescriptors or superdescriptors composed of partial fields, PDS for phonetic descriptors, HDS for hyperdescriptors, or DSC for a simple descriptor.
`NU`	Indicates if the field uses null suppression. Values are YES or NO.
`PASS`	Adabas password for the file. This attribute can be added manually.
`KEYFLD`	Only for child segments. This is the common field in the parent segment, which is used to relate the two files.
`IXFLD`	Only for child segments. This is the common field in the child segment, which is used to relate the two files.

Reference: Comments in Master and Access Files

The generated Master File and Access File contain several commented entries. They are provided for the client's convenience and are not used by the server. Comments are provided for the:

- CREATE SYNONYM time stamp and creator's User ID (included in both the Master and Access Files).
- Predict file name, file number, and DBID, if Predict is used.
- Headers before special fields: super/sub/hyper/phonetic descriptors or fields without standard length.

- Fields not supported by the adapter.
- Component fields of superdescriptors that are composed of partial fields (described as NOP type).

Syntax: How to Use the ISN and GFBID Parameters With CREATE SYNONYM

```
CREATE SYNONYM app/synonym
FOR FILE=file-number[/predict-file-number[(predict-filename)]]
DBMS ADBSINX
DATABASE dbid[/predict-dbid][PARMS|GFBID|ISN]
END
```

where:

`GFBID`

Specifies that a field generated in the Master File is to be used to specify Adabas Global Format Buffer ID values.

`ISN`

Specifies that a field generated in the Master File is to be used to specify Adabas Internal Sequence Numbers (ISN).

For complete CREATE SYNONYM syntax, see *Create a Synonym Manually* on page 2-23.

Reference: Usage of Master File Fields With ISN Support

The Adapter for Adabas can employ a new data retrieval strategy through Read Logical by ISN (L1) calls. It can be used to determine the Internal Sequence Number (ISN) of a record that was read or that will be inserted into an Adabas file.

ISN-based access is applicable only if an ISN field is described in the Master File. This field has a field name that is user-defined and an ALIAS of ISN. The Usage format is I and the Actual format must be I4. This field can be defined only in segments that contain non-repeating data (that is, using an access method defined in the Access File as ADBS):

```
FIELD=ISN_FIELD, ALIAS=ISN, I10, I4, $
```

Equality tests on the ISN field can be used to retrieve a single record when the:

- Report request contains an equality operator in the selection test on an ISN list.

  or

- ISN field is used as the cross-referenced field in the Join to an Adabas file.

Adabas returns the Response Code 113 if the record with the ISN defined in the test is not present in the Address Converter for the file. The adapter returns the following message: "Record is not found".

The Adapter for Adabas uses Read Logical by ISN (L1) calls to retrieve all records for the entry segment when the:

- Report request does not contain optimizable selection criteria on an inverted list.
- Access File contains a SEQFIELD value for the segment whose value is equal to the ISN field name.

For example, the Access File ADATEST contains:

```
SEGNAM=S01, ACCESS=ADBS , ... ,SEQFIELD=ISN_FIELD ,$
```

Adabas returns each record in ascending order by the value of the ISN.

Example: Equality Test on ISN Field

```
SELECT AA_FIELD, AJ_FIELD FROM ADATEST
WHERE ISN_FIELD = 1100;
```

Note: Multifetch option will be suppressed for this call.

Read Logical by ISN (L1) calls to retrieve a subset of records for the entry segment when the:

- Report request contains the GE/GT relational operator in the selection test on an ISN list.
- Report request does not contain any selection tests on an inverted list.
- Access File contains a SEQFIELD value for that segment.

Example: Inequality Test on ISN Field

Access File ADATEST contains:

```
SEGNAM=S01, ACCESS=ADBS, ... ,SEQFIELD=AA_FIELD ,$
```

Read Logical by ISN (L1) is used:

```
SELECT AA_FIELD, AE_FIELD FROM ADATEST
WHERE ISN_FIELD > 1100;
```

Reference: ISN for Insert

After issuing an Insert request on a file with an ISN field in the Master File, the resulting ISN generated by Adabas is displayed by the message FOC4592:

```
(FOC4592) RECORD IS INSERTED WITH ISN : nnnnn
```

Example: Issuing an Insert Request

```
SQL
 INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD)
VALUES ('11111111', 'TAMPA');
END

ADBTEST ADBSINX ON 09/20/2002 AT 15.22.16
(FOC4592) RECORD IS INSERTED WITH ISN : 11
(FOC4566) RECORDS AFFECTED DURING CURRENT REQUEST : 1/INSERT

TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 1 UPDATED = 0 DELETED = 0
```

You can assign a value for the ISN field if the Insert request contains the ISN field and the assigned value is not 0.

The adapter will issue an Adabas N2 Direct Call to assign this ISN value to the inserted record. Adabas returns Response Code 113 if this value was already assigned to another record in the file or if it is larger than the MAXISN in effect for the file.

Reference: Master File Fields and GFBID Support

The Adapter for Adabas can optimize the performance of queries that use the same Adabas field lists repeatedly. Field lists are generated in Adabas format buffers, and can be retained.

Global Format Buffer ID (GFBID) support is applicable only if a GFBID field is present in the Master File. This field has a user-defined field name, and an ALIAS of GFBID. This field is used to determine the Global Format Buffer ID that will be defined in read requests with identical field lists for the same database. The GFBID field has a USAGE format of A8 and an ACTUAL format of A8. This field can be defined only in segments that contain non-repeating data (that is, the access method defined in the Access File is ADBS).

Example: Using GFBID Syntax in a Master File

```
FIELD=GID_FIELD, ALIAS=GFBID, A8, A8, $
```

Note:

- If the field list is changed but the same GFBID is used in a request, then incorrect results may be displayed. In some cases, a message about the possible mismatch between the Field Definition Table (FDT) and Master File will be issued.
- If two requests have the same list of selected fields but different fields are used in selection criteria, then the requests must use different GFBID values.

The GFBID field can be defined in the Master File manually or using the Create Synonym facility with option PARMS GFBID.

Example: Using the GFBID Parameter in the CREATE SYNONYM Command

```
CREATE SYNONYM ADATEST FOR 11 DBMS ADBSINX DATABASE 3 PARMS 'GFBID'
END
```

The value of GFBID can have up to 8 bytes and must start with a digit or uppercase character.

If GFBID is used in a request against a file that contains simple PE/MU segments, it will be applied to Adabas calls to get the field values from these segments. The GFBID value will be adjusted for each non-ADBS segment; that is, a segment number is placed in the last (8th) byte of the GFBID field.

Calls to a PE (periodic element) with MU (multi-value) child or an MU with PE parent segment will not use GFBID values.

Note: To make the GFBID value unique within Adabas, do not use 8 byte GFBID values for these types of calls.

- If a GFBID field is defined in the Master File, it can be used in a request as part of the selection test. The adapter will take the GFBID value from the selection criteria, deactivate the field, and remove it from the match array. The GFBID value is placed in the ADD5 field of the Adabas Control Block for the request.
- If the GFBID is applied to an Adabas call, then field name ADD5 appears in the adapter trace. GFBID values that were issued in all requests to the database are accumulated in the Adabas internal queue. These values can be removed from the queue by issuing the following adapter SET command:

```
 ENGINE ADBSINX SET GFBID_OFF ALL/<value> DBID <number>
```

When a single <value> is in the SET command, all values issued for child segments are removed as well. If <value> contains blanks, then value must be enclosed in single quotation marks.

ALL clears all GFBID values.

Example: Using GFBID Syntax

Master File ADATEST contains:

```
FIELD=GID_FIELD, ALIAS=GFBID, A8, A8, $
```

Request 1. Global ID = ABC1:

```
SQL
SELECT AA_FIELD, AJ_FIELD, ISN_FIELD, AQ0201_OCC, AR_FIELD, AS_FIELD,
AT_FIELD
FROM EMPLOYEES
WHERE ISN_FIELD > 1100 AND GID_FIELD = 'ABC1';
END
```

Result:

```
PAGE 1

 AA_FIELD AJ_FIELD      ISN_FIELD AQ0201_OCC AR_FIELD AS_FIELD AT_FIELD
 -------- --------      --------- ---------- -------- -------- --------
 30034517 DERBY         1105      1          UKL      6500     750
 30034517 DERBY         1105      2          UKL      6800     1240
 30034517 DERBY         1105      3          UKL      7100     1345
 30034517 DERBY         1105      4          UKL      7450     1450
 30034517 DERBY         1105      5          UKL      7800     1700
 50005600 VIENNE        1106      1          FRA      165810   10000
 50005300 PARIS         1107      1          FRA      166900   5400

 NUMBER OF RECORDS IN TABLE= 7 LINES= 7
```

Request 2. Global ID = ABC1, but field's list is changed:

```
SQL
SELECT AA_FIELD, ISN_FIELD, AQ0201_OCC, AR_FIELD, AS_FIELD, AT_FIELD
FROM EMPLOYEES
WHERE ISN_FIELD > 1100 AND GID_FIELD = 'ABC1';
END
```

Result:

```
(FOC4567) POSSIBLE MISMATCH BETWEEN FDT AND MFD : file=19879548
(FOC4591) ERROR RETURN ON READ BY ISN : S01 /999:ffffffcd001d00

 NUMBER OF RECORDS IN TABLE= 0 LINES= 0
```

Request 3. Remove value 'ABC1' from Adabas:

```
ENGINE ADBSINX SET GFBID_OFF ABC1 DBID 3
```

Request 4. Repeat request 2.

```
SQL
SELECT AA_FIELD, ISN_FIELD, AQ0201_OCC, AR_FIELD, AS_FIELD, AT_FIELD
FROM EMPLOYEES
WHERE ISN_FIELD > 1100 AND GID_FIELD = 'ABC1';
END
```

Result:

```
 PAGE 1

 AA_FIELD ISN_FIELD AQ0201_OCC AR_FIELD AS_FIELD AT_FIELD
 -------- --------- ---------- -------- -------- --------
 30034517 1105      1          UKL      6500     750
 30034517 1105      2          UKL      6800     1240
 30034517 1105      3          UKL      7100     1345
 30034517 1105      4          UKL      7450     1450
 30034517 1105      5          UKL      7800     1700
 50005600 1106      1          FRA      165810   10000
 50005300 1107      1          FRA      166900   5400

 NUMBER OF RECORDS IN TABLE= 7 LINES= 7
```

Request 5. Remove all GFBID's values for Database 3 from Adabas:

```
ENGINE ADBSINX SET GFBID_OFF ALL DBID 3
```

Overview of Master and Access Files

The server processes a report request with the following steps:

1. It locates the Master File for the Adabas file. The file name in the report request is the same as the OS/390 and z/OS PDS member name.
2. It detects the SUFFIX attribute, ADBSINX, in the Master File. Since this value indicates that the data resides in an Adabas file, it passes control to the adapter.
3. The adapter locates the corresponding Access File and uses the information contained in both the Master and Access Files to generate the Adabas direct calls required by the report request. It passes these direct calls to the Adabas DBMS.
4. The adapter retrieves the data generated by the Adabas DBMS and returns control to the server. For some operations, the server may perform additional processing on the returned data.

Master Files for Adabas

In this section:

File Attributes in Master Files

Segment Attributes in Master Files

Field Attributes in Master Files

Standard Master File attributes are generally used to describe Adabas files, but there are concepts in Adabas file processing that require special terminology at the field and segment level. On OS/390 and z/OS, the Master File is a member of a PDS with DDNAME MASTER.

The server permits a great deal of flexibility in its Master Files. For the Adapter for Adabas, you need to describe only the Adabas fields actually used in reporting applications, omitting any unnecessary Adabas fields. (Note the exception that if a PE group is included in the Master File, all fields of the PE group must be included. See *Field Attributes in Master Files* on page 2-43 and *Describing Multi-value Fields Within Periodic Groups* on page 2-74 for additional information.) Additionally, describing the same Adabas file in multiple ways enables you to access that same file in its different configurations within one procedure.

Master Files have three parts: file declaration, segment declaration, and field declaration, each of which is explained in these topics. Concepts that are specific to Adabas files, for which the server requires unique syntax in the Master File, are also explained.

The following is an example of the Master File for VEHICLES.

```
$$$ CREATED ON 12/10/03 AT 10.17.27 BY PMSMJB
FILENAME=ADACAR,SUFFIX=ADBSINX,$

$ ADABAS FILE = VEHICLES-FILE                     DICTIONARY = 6
SEGNAME=S01     ,SEGTYPE=S,$
 FIELD=REG_NUM                    ,ALIAS=AA     ,A15   ,A15   ,INDEX=I,$
 FIELD=CHASSIS_NUM                ,ALIAS=AB     ,I9    ,I4    ,$
 FIELD=PERSONNEL_ID               ,ALIAS=AC     ,A8    ,A8    ,INDEX=I,$
 GROUP=CAR_DETAILS                ,ALIAS=CD     ,A50   ,A50   ,$
  FIELD=MAKE                      ,ALIAS=AD     ,A20   ,A20   ,INDEX=I,$
  FIELD=MODEL                     ,ALIAS=AE     ,A20   ,A20   ,$
  FIELD=COLOR                     ,ALIAS=AF     ,A10   ,A10   ,INDEX=I,$
 FIELD=YEAR                       ,ALIAS=AG     ,P2    ,Z2    ,$
 FIELD=CLASS                      ,ALIAS=AH     ,A1    ,A1    ,INDEX=I,$
 FIELD=LEASE_PUR                  ,ALIAS=AI     ,A1    ,A1    ,$
 FIELD=DATE_ACQ                   ,ALIAS=AJ     ,P6    ,Z6    ,$
$GRMU=CAR_MAINTENANCE             ,ALIAS=AK     ,A11   ,A7    ,$
  FIELD=CURR_CODE                 ,ALIAS=AL     ,A3    ,A3    ,$
  FIELD=MAINT_COST_CNT            ,ALIAS=AMC    ,I4    ,I2    ,$
 FIELD=DAT_ACQ_DESC               ,ALIAS=AN     ,A4    ,A4    ,INDEX=I,$
 GROUP=MODEL_YEAR_MAKE            ,ALIAS=AO     ,A28   ,A22   ,INDEX=I,$
  FIELD=YEAR_S02                  ,ALIAS=AG     ,P2    ,Z2    ,$
  FIELD=MAKE_S02                  ,ALIAS=AD     ,A20   ,A20   ,INDEX=I,$

SEGNAME=AM0101  ,SEGTYPE=S,PARENT=S01  ,OCCURS=AMC,$  MAX= 60
 FIELD=MAINT_COST                 ,ALIAS=AM     ,P7    ,P4    ,$
 FIELD=AM0101_OCC                 ,ALIAS=ORDER  ,I4    ,I2    ,$
```

File Attributes in Master Files

Master Files begin with a file declaration that names the file and describes the type of data source. The file declaration has two attributes, FILENAME and SUFFIX.

The syntax is

```
FILE[NAME]=filename, SUFFIX=ADBSINX  [,$]
```

where:

filename

Is a one- to eight-character name that complies with server file naming conventions.

ADBSINX

Is the value that specifies the Adapter for Adabas.

FILENAME

The FILENAME (or FILE) is any valid name from one to eight characters long.

In the OS/390 and z/OS environment, it is a member of a partitioned data set with the DDNAME MASTER.

SUFFIX

The value for the SUFFIX attribute is always ADBSINX, which is the name of the Adapter for Adabas program that the server loads to read an Adabas database.

Example: File Declaration

The following example is a typical file declaration:

```
FILENAME=ADACAR, SUFFIX=ADBSINX,$
```

Segment Attributes in Master Files

In this section:

SEGNAME

SEGTYPE

PARENT

OCCURS

How to:

Name a Root Segment in the Master File

Name a Segment for a PE Group and MU Field

Example:

Naming a Segment for a PE Group and MU Field

Segment Declaration

Each segment declaration describes one Adabas record or a subset of an Adabas record (called a logical record type). Each logical record type described in a Master File requires a segment declaration that consists of at least two attributes, SEGNAME and SEGTYPE.

The syntax is

```
SEGNAME=segname, SEGTYPE=segtype [,PARENT=parent]
                                 [,OCCURS=occursname][,$]
```

where:

segname

Is a unique one- to eight-character name that identifies the segment.

segtype

Identifies the characteristics of the segment. Possible values are:

`S` which indicates multiple instances of a descendant segment are related to a given parent.

`SO` which indicates the Write Adapter for Adabas (for the Read Adapter for Adabas, SEGTYPE=S is sufficient).

`U` which indicates a single instance of data for a descendant segment is related to its parent.

`KL` which indicates multiple instances of a cross-referenced descendant segment are related to a given parent.

`KLU` which indicates a single instance of a cross-referenced descendant segment is related to its parent.

parent

Is the name of the parent segment from the Adabas database.

occursname

Indicates a data field in the parent segment that contains the number of occurrences of the descendant segment. Its value is derived from the two-character internal Adabas name (ALIAS field on the server) appended with the letter C.

SEGNAME

The SEGNAME (or SEGMENT) attribute is the name of a group of data fields that have a one-to-one relationship to each other in the Adabas file.

You may describe an Adabas file in the server's hierarchical design to take advantage of multi-value fields and periodic groups. You may have multiple Adabas records described in the same server view. Each record is described as a segment with a unique SEGNAME.

Syntax: How to Name a Root Segment in the Master File

For the root segment of the Adabas file, the segment name is

```
Snn
```

where:

nn

Is a two-digit number indicating the order in which the file was selected.

The first file (selected as the root from the File Selection Menu) has SEGNAME=S01. Subsequent files used in the same description (selected as children from the File Selection Menu) have SEGNAME=S02, SEGNAME=S03, and so on.

Syntax: How to Name a Segment for a PE Group and MU Field

The segment names for PE (periodic element) groups and MU (multi-value) fields have the format

aammnn

where:

aa

Is the Adabas field name.

mm

Is a two-digit number that indicates the order in which the PE group or MU field appears in the PREDICT description of the file.

nn

Is the order number used for the root segment.

Example: Naming a Segment for a PE Group and MU Field

Using the syntax aammnn, SEGNAME=BE0201 describes the segment for the field BE, which is the second (02) PE group or MU field described in the first segment (01).

`SEGTYPE`

Is the root segment and cross-reference segment type. The SEGTYPE is S0 for the root file and children with a non-unique IXFLD. The SEGTYPE is U for a child with a unique IXFLD. The SEGTYPE for all PE and MU fields is S0.

`PARENT`

Is the value of SEGNAME of the parent record.

`OCCURS`

Indicates the number of occurrences of a PE group or MU field. This attribute contains the Adabas field name of the PE group or MU field with the suffix C, which is the ALIAS of the counter field in the parent segment.

SEGTYPE

The SEGTYPE attribute identifies the basic characteristics of related or cross-referenced segments. Cross-referencing is a method of accessing information from two or more different files or segments to use in a single report request.

PARENT

Any segment except the root is a descendant, or child, of another segment. The PARENT attribute supplies the name of the segment which is the logical parent or owner of the current segment. If no PARENT attribute appears, the default is the immediately preceding segment; however, it is highly recommended to include the parent. The PARENT name is the one- to eight-character SEGNAME of a previous segment.

OCCURS

For more information on the segment attribute OCCURS, which applies to repeating fields and groups, see *Describing Multi-value Fields and Periodic Groups With the OCCURS Attribute* on page 2-71.

Example: Segment Declaration

The following two examples are typical segment declarations:

```
SEGNAME=S01          ,SEGTYPE=S     ,$
SEGNAME=AM0101       ,SEGTYPE=S     ,PARENT=S01     ,OCCURS=AMC    ,$
```

Field Attributes in Master Files

In this section:

FIELDNAME

ALIAS

Using the ALIAS to Reformat Adabas Fields

USAGE

ACTUAL

INDEX

GROUP

Example:

Field Declaration

Field declarations describe the fields in each file. For the simplest file structures (single-segment, fixed-length files), you need to describe only the Adabas fields you actually use for reporting purposes.

Put the fields in any order within their segment, except in the case of fields within periodic groups (PE). The PE group must have all fields defined and in the proper order.

The syntax is

```
FIELD[NAME]=fieldname, ALIAS=alias, [USAGE=]display, [ACTUAL=]format,$
```

where:

fieldname

Is the 1- to 66-character name that is unique to the Master File.

alias

Is the two-character internal Adabas field name.

display

Is the server display format for the field.

format

Is the server definition of the Adabas field format and length.

Note: Field declarations are always terminated with ,$.

There are additional attributes that apply to superdescriptor, cross-referenced, repeating, and other types of fields.

FIELDNAME

The field name appears as a column heading when you include the field in a report request. You can change the default by using AS or NOPRINT in the report or by using TITLE in the Master File.

Field names are unique names of up to 66 characters. The Master File field name does not need to be the same as the Adabas field name. Field names can include any alphanumeric characters, but the first character must be a letter from A to Z. Avoid embedded blanks and special characters.

Since all references to data on the server are through field names or aliases, the field names should be unique within a Master File. This requirement is true even when the same Adabas record is included as two differently structured segments in the same Master File. You must specify different field names in the two segments or the server uses the first one in the Master File unless you qualify which field you want, for example, file.field, segment.field.

ALIAS

The ALIAS for a single Adabas field *must* contain the two-character internal Adabas field name. The adapter uses it to generate direct calls to the Adabas DBMS. The format rules are as follows:

- The ALIAS equals the two-character internal Adabas field name, for example, AM.
- The letter C is appended to the two-character internal Adabas field name if the field is the counter field for a PE group or MU field, for example, AMC.
- The ALIAS is the two-character internal Adabas field name appended with the digit 1 if an application calls for only the first occurrence of a repeating field, for example, AM1.
- The ALIAS can be used to reformat the length of the Adabas field. See *Using the ALIAS to Reformat Adabas Fields* on page 2-45 for examples using the reformatting option.

Note: Because of the requirement that the ALIAS in a Master File be the same as the two-character internal Adabas field name, your Master File may include duplicate ALIAS values. In that case, the field name is used to differentiate between the segments.

Using the ALIAS to Reformat Adabas Fields

The ALIAS field allows reformatting of the field for a different view. Several Adabas data formats have lengths that exceed the supported maximum length of data formats on the server. You can use the ALIAS to convert this data to an acceptable ACTUAL format and length.

When a field is specified in a retrieval request, the value in the ALIAS is passed to Adabas in the Format buffer. Adabas uses the ALIAS to process the field and return the data to the server.

To use the reformatting option, define the ALIAS using length and format values that indicate how you want to reformat the field.

The syntax is

```
ALIAS='xx,l,f'
```

where:

xx

Is the two-character internal Adabas field name.

l

Is the new length of the field value in number of bytes.

f

Is the new Adabas format of the field value.

The entire value must be enclosed in single quotation marks.

For example, if you want to use only the first three characters in a six-byte field, use the following syntax in your Master File:

```
FIELD=L_DBNR, ALIAS='AU,3,P', USAGE=P3, ACTUAL=P3   ,$
```

In this case, the Adapter for Adabas will read only the first three bytes (L_DBNR is a six-byte field with the Adabas description: 01, AU, 6, U).

In the next example, to convert an Adabas binary field format of ten bytes (Adabas description: 01, BB, 10, B) to an alphanumeric format large enough to accommodate the Adabas value, you could use the following syntax in your Master File:

```
FIELD=TESTFLD, ALIAS='BB,10,A', USAGE=A20, ACTUAL=A10   ,$
```

Notice that you must also change the USAGE and ACTUAL values to reflect the changes in the format. Refer to the chart in *ACTUAL* on page 2-47 for the acceptable USAGE and ACTUAL values along with the corresponding Adabas codes.

Note: Groups cannot be reformatted. This formatting is valid only for elementary fields.

USAGE

The USAGE attribute enables you to describe how you want fields displayed on the screen, printed in reports, or used in calculations. This attribute includes the data field type, length, and any applicable edit options when the field values are printed. FORMAT is a synonym for USAGE.

The syntax is

```
USAGE=tllleeeee
```

where:

t

Is the field type.

lll

Is the number of positions needed to display the field.

eeeee

Are the edit options.

The value that you specify for field length is the number of alphanumeric characters or numeric digits that the server allocates for the field in any display or report. The specified value excludes the editing characters, such as comma, $, and so on. Edit options control how a field value is printed.

ACTUAL

The ACTUAL attribute describes the length and type of the Adabas field in storage. It is used to describe the format of fields from external data files only, and must be included in the Master Files needed for the Adapter for Adabas. The source of this information is your existing Adabas Field Definition Table (FDT).

The syntax is

```
ACTUAL=tlll
```

where:

t

Is the server field type of the Adabas field.

lll

Is the storage length of the Adabas field.

The server permits the following conversions from ACTUAL format to USAGE (display) format:

Adabas Format Type	**Server ACTUAL Type**	**Server ACTUAL Length***	**Server USAGE Type**	**Server USAGE Length***	**Description**
A	A	1-256	A	1-256	Alphanumeric
G	F D	4 8	F D	1-9 1-15	Float Single-Precision Float Double-Precision
F	I	2, 4	I	1-9	Integer
B 1-4 5-6 7-126	 I P A	 1-4 7-8 9-126	 I P A	 1-9 12-15 14-252	 Integer Values Packed Decimal Alpha
P	P	1-15	P	1-33.*n*	Packed Decimal
U	Z	1-29	A or P	1-33.*n*	Zoned

* Lengths are measured as follows:

- ACTUAL is in number of bytes.
- USAGE allows space for all possible digits, a minus sign for negative numbers, and a decimal point in numbers with decimal digits.

The following table shows the codes for the types of data the server reads:

ACTUAL Type	Description
`A`*n*	Alphanumeric characters A to Z, 0 to 9, and the special characters in the EBCDIC display mode, where *n* = 1 to 256 bytes.
`D8`	Double-precision, floating-point numbers, stored internally in eight bytes.
`F4`	Single-precision, floating-point numbers, stored internally in four bytes.
`I`*n*	Binary integers, where *n* = 1 to 4.
`P`*n*	Packed decimal data, where *n* = 1 to 15 bytes.
`Z`*n*	Zoned decimal data, where *n* = 1 to 30 bytes. Represent the field as *Zm.n*, where *m* is the total number of digits, and *n* is the number of decimal places (for example, Z6.1 means a six-digit number with one decimal place).

INDEX

For more information on the field attribute INDEX, which applies to descriptors, see *Specifying INDEX=I* on page 2-60.

GROUP

For more information on the field attribute GROUP, which applies to superdescriptors, see *Specifying Superdescriptors Using the GROUP Attribute* on page 2-61.

Example: Field Declaration

The following is a typical field declaration:

```
FIELD=MODEL ,ALIAS=AE ,USAGE=A20 ,ACTUAL=A20 ,$
```

Access Files for Adabas

In this section:

Release Declaration

Segment Attributes in Access Files

An Access File describes the physical attributes of the Adabas file, such as the database number, file number, descriptors, and security. See *Overview of Master and Access Files* on page 2-36 for more information about Master and Access Files.

Access Files store database access information used to translate report requests into the required Adabas direct calls.

In the OS/390 and z/OS environment, the Access File is located in a PDS allocated to the DDNAME ACCESS. Each member is a separate Access File.

An Access File consists of 80-character records in comma-delimited format. A record in an Access File contains a list of attributes and values, separated by commas and terminated with a comma and dollar sign (,$). This list is free-form and spans several lines if necessary. You can specify attributes in any order.

The server reads blank lines, and lines starting with a dollar sign in column one, as comments.

Every Access File contains a release declaration and at least one segment declaration. Release declarations and segment declarations each have their own set of attributes. These attributes are discussed in the following topics.

The following is a sample Access File:

```
$$$ FILENAME=EMPFILE1,SUFFIX=ADBSINX,$
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = EMPLOYEES                          DICTIONARY =
SEGNAM=S01    ,ACCESS=ADBS,FILENO=001,DBNO=1,CALLTYPE=RL,
              SEQFIELD=PERSONNEL_ID,$
FIELD=DEPT_PERSON                         ,TYPE=SPR,$
FIELD=DEPT                                ,TYPE=DSC,NU=YES,$
FIELD=NAME                                ,TYPE=   ,NU=NO,$
 FIELD=LEAVE_LEFT                         ,TYPE=SPR,$
 FIELD=LEAVE_DUE                          ,TYPE=   ,NU=YES,$
 FIELD=LEAVE_TAKEN                        ,TYPE=   ,NU=YES,$
FIELD=DEPARTMENT                          ,TYPE=NOP,NU=NO,$
SEGNAM=AI0101,ACCESS=MU  ,FILENO=001,$ ADDRESS_LINE
SEGNAM=AQ0201,ACCESS=PE  ,FILENO=001,DBNO=1,$ INCOME
SEGNAM=AT0301,ACCESS=MU  ,FILENO=001,DBNO=1,$ BONUS
SEGNAM=AW0401,ACCESS=PE  ,FILENO=001,$ LEAVE_BOOKED
SEGNAM=AZ0501,ACCESS=MU  ,FILENO=001,$ LANG
```

Release Declaration

In this section:

RELEASE

OPEN

The release declaration must be the first uncommented line of the Access File. It identifies the release of Adabas and indicates whether the Adapter for Adabas issues Adabas OPEN and CLOSE calls for each report request.

The syntax is

```
RELEASE=relnum [,OPEN={YES|NO}] ,$
```

where:

relnum

Is the Adabas release number.

OPEN

Specifies whether the adapter issues Adabas OPEN and CLOSE calls to Adabas in each report request. Possible values are:

YES which indicates that Adabas OPEN and CLOSE calls are issued. YES is the default value.

NO which indicates that Adabas OPEN and CLOSE calls are not issued.

RELEASE

The first declaration in the Access File contains the Adabas release number with the attribute RELEASE. Specify the full release number. The value is for documentation only.

The following example illustrates the use of the RELEASE attribute to specify the Adabas release number:

```
RELEASE=6 ,$
```

OPEN

The OPEN attribute specifies whether the Adapter for Adabas issues Adabas OPEN and CLOSE calls to Adabas in each report request. Specifying YES or NO achieves the following results:

Value	**Result**
`YES`	The adapter issues an OPEN call to Adabas at the start of a call set initiated by a retrieval request. An Adabas CLOSE is issued at the end of retrieval for the request. YES is the default value. `RELEASE=6 , OPEN=YES ,$`
`NO`	Adabas OPEN and CLOSE calls are not issued for any retrieval request. `RELEASE=4.1, OPEN=NO ,$`

Accept the default value (YES) unless you are using an Adabas release lower than 5.0.

Segment Attributes in Access Files

In this section:

Segment declarations contain detailed information about each segment of an Adabas file. The segment declaration attributes specify the segment name, the file and database numbers, and the Adabas password, as well as retrieval options.

The syntax is

```
SEGNAM=segname, ACCESS=access, FILENO=file_number [,DBNO=database_number]
[,CALLTYPE={FIND|RL}] [,USE={FIND|ANY}] [,SEQFIELD=seqfield]
[,PASS=password]
[,FETCH={ON|OFF}] [,FETCHSIZE={n|MAX}] [,UNQKEYNAME=unique_field]
[,WRITE=YES|NO],$
```

where:

segname

Is the SEGNAME value from the Master File.

access

Specifies the segment type that the adapter uses for each segment. Possible values are:

ADBS which is for a segment that contains only non-repeating fields. ADBS is the default value.

PE which is for a segment that is a periodic group in the same file as the parent segment.

MU which is for a segment that is a multi-value field in the same file as the parent segment.

file_number

Is the number that identifies an Adabas file.

database_number

Is the number that identifies an Adabas database.

CALLTYPE

Indicates the type of data retrieval call constructed by the adapter. Possible values are:

FIND in which a FIND call is issued when there are one or more WHERE or IF statements in a retrieval request detected against fields defined with INDEX=I. FIND is the default value.

RL in which a Read Logical (RL) call is issued when there are one or more WHERE or IF statements against fields defined with INDEX=I.

Note: See your database administrator for the most efficient CALLTYPE to be used at your site.

USE

Indicates a method for handling unreadable sub or suprerdescriptor fields that are derived from PE or MU access types and defined in the root segment. Possible values are:

FIND indicates that the S1 command will be issued against such a field even if CALLTYPE is RL.

ANY indicates that fields are not of the sub/superdescriptor types. ANY, or the absence of a USE= entry, are the default values.

seqfield

Provides a default index which controls Read Logical (RL) retrieval when there is no IF or WHERE selection test.

password

Is the Adabas password for the file.

`FETCH`

Indicates whether to use the Fetch feature. Possible values are:

`ON` which sets the Fetch feature on for the user session. ON is the default value.

`OFF` which sets the Fetch feature off for the user session.

If you include this attribute, you must also include FETCHSIZE (see below).

`FETCHSIZE`

Sets the number of records per buffer. Possible values are:

`n` which is the number of records per buffer (1 to 999). 10 is the default value.

`MAX` which is automatically calculated by the adapter to allow the maximum number of records that fit in a 32K buffer.

`UNQKEYNAME`

Is the attribute that indicates the unique key. It does not necessarily define the key described in the ADABAS FDT with the UQ option.

`WRITE`

Is the attribute that indicates the WRITE capability. NO is the default value.

Note: FETCH and FETCHSIZE are applicable only to segments described as ACCESS=ADBS. FETCH and FETCHSIZE can be set dynamically to override the Access File settings.

Special attributes that apply to embedded JOINs are explained in *Implementing Embedded JOINs: KEYFLD and IXFLD* on page 2-65. Attributes that apply to superdescriptors and subdescriptors are explained in *Customizing the Adabas Environment* on page 2-78.

SEGNAM

The SEGNAM attribute specifies the name of the segment being described. It is the same name as the SEGMENT value in the Master File.

ACCESS

The ACCESS attribute specifies the segment type that the Adapter for Adabas uses for each segment. The values are:

- ADBS, which specifies a segment that contains only non-repeating fields. ADBS is the default value.
- PE, which specifies a segment that is a periodic group in the same file as the parent segment.
- MU, which specifies a segment that is a multi-value field in the same file as the parent segment.

All three access values use the attributes FILENO and DBNO, as shown in this partial Access File for EMPLOYEES with the specification of PE and MU fields.

```
$$$ FILENAME=EMPFILE1,SUFFIX=ADBSINX,$
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = EMPLOYEES                         DICTIONARY =
SEGNAM=S01    ,ACCESS=ADBS,FILENO=001,DBNO=1,CALLTYPE=RL,
               SEQFIELD=PERSONNEL_ID,$
SEGNAM=AQ0201,ACCESS=PE   ,FILENO=001,DBNO=1,$ INCOME
SEGNAM=AT0301,ACCESS=MU   ,FILENO=001,DBNO=1,$ BONUS
SEGNAM=AW0401,ACCESS=PE   ,FILENO=001,DBNO=1,$ LEAVE_BOOKED
```

ACCESS=ADBS

ACCESS=ADBS specifies the main segment of the file. This segment contains only non-repeating fields.

FILENO

The FILENO attribute specifies the Adabas file number. You need this file number to identify the Adabas file(s) you wish to access. The valid values are 1 to 255 (5000 is the maximum value for a mainframe platform). The FILENO attribute has to be defined for segments with ACCESS=ADBS. If the FILENO attribute is omitted for segments with ACCESS=MU/PE, then it will be taken from the corresponding parent segment.

Suppose you are accessing two segments: the first is in the EMPLOYEE file, and the second is in the INSURANCE file. To describe this situation in the Access File, use the SEGNAM and FILENO attributes as illustrated in DBNO.

DBNO

The DBNO attribute specifies the Adabas database number. It is used when you define files from multiple databases in one Master File. If it is not provided, the DBNO will be read from the DDCARD (on a mainframe platform only). If the DDCARD is not allocated, the adapter uses the default value, DBNO=0.

If the DBNO attribute is omitted for a segment with ACCESS=PE/MU, then it will be taken from the corresponding parent segment. The DBNO attribute has to be defined for segments with ACCESS=ADBS on non-mainframe platforms.

The following example illustrates the use of both the FILENO and DBNO attributes to account for the files being in different databases:

```
SEGNAM=ONE    ,FILENO=1   ,DBNO=1 ,... ,$
SEGNAM=PAY    ,FILENO=2   ,DBNO=3 ,... ,$
```

CALLTYPE

The CALLTYPE attribute affects how the Adapter for Adabas processes WHERE and IF statements on descriptors in report requests and how it retrieves cross-referenced file data from a descendant segment. With CALLTYPE, you optionally specify the type of access to use in order to process the inverted lists for any given segment. Ask your Adabas database administrator to advise you when you are selecting a CALLTYPE.

The possible values are:

Value	Description
FIND	An Adabas FIND call is issued when there are one or more WHERE or IF statements against fields defined to the server with INDEX=I. The search is on specific values of the field. The Adapter for Adabas can generate Adabas FIND (S1) calls even when non-descriptor fields are used as search criteria. This can occur whenever CALLTYPE=FIND is specified in the Access File, and if you include an IF or WHERE test when referencing a non-descriptor field in your TABLE or TABLEF request. To turn off this feature, see the topic in *Adapter Navigation* on page 2-105 that discusses the optimization of the FIND call using non-descriptor fields.

Value	Description
RL	An Adabas Read Logical call is issued when there are one or more WHERE or IF statements against fields defined to the server with INDEX=I. **Note:** If you have selected a field defined with TYPE=NOP (for example, subdescriptors), a FIND call is issued even if RL has been specified in cases when: • SET NOPACCESS FIND was performed. • The server is running in OpenVMS environment. • The same Adabas field is defined in the descendant PE/MU segment. Switching from RL to FIND mode is also performed if a field is defined: • With TYPE=SPR and the server is running in an OpenVMS environment. • With TYPE=PDS (phonetic descriptor) or TYPE=HDS (hyperdescriptor). • In a PE segment (is a part of periodic group). The Adabas Interface uses Read Logical by ISN (L1) calls to retrieve all records for the entry segment when the: • Report request does not contain an optimizable selection test on an inverted list or an ISN list. • Access File contains a SEQFIELD value for that segment and this value is equal to the ISN field name. Adabas returns each record corresponding to an entry in the Address Converter. The records are in ascending value of the ISN. The first IF statement in a request that refers to a field with INDEX=I determines the inverted list used for the Read Logical access for the root of the subtree. Refer to *Referencing Subtrees and Record Retrieval* on page 2-88 for information on subtrees. When retrieving descendant segments, the Adapter for Adabas issues a Read Logical call using the descriptor pointed to by IXFLD. For more information on IXFLD, refer to *Implementing Embedded JOINs: KEYFLD and IXFLD* on page 2-65. For CALLTYPE=RL, all other selection tests on descriptor fields specified in your report request are applied after the record is returned.

USE=FIND

The Adapter for Adabas cannot read sub/superdescriptors that are derived from PE/MU access types and defined in a root segment. These types of fields may be used only in an Adabas FIND command that produces a "trusted" answer set. However, you can add the description parameter USE=FIND in the Access File for sub and superdescriptor fields that cannot be read by the adapter.

The following processing rules apply for fields that are defined with this description or that are automatically recognized as a fields derived from PE/MU:

- The Adabas command S1 is issued even if CALLTYPE is RL in the Access File.
- If the field is used in a PRINT/WRITE list, the adapter issues the error message:

  ```
  (FOC4582) THIS NOP/SPR FIELD CAN BE USED IN FIND CALLS ONLY.
  ```

- If the field is used in a complex WHERE condition and can't be fully converted to the generated FIND call, the adapter issues the error message:

  ```
  (FOC4493) THIS NOP/SPR FIELD CANNOT BE USED IN COMPLEX FIND
  ```

The CREATE SYNONYM command always places sub/superdescriptors derived from PE/MU in a root segment with TYPE=NOP in the Access File. Moreover, the sub descriptors have the description USE=UFIND in the Access File.

SEQFIELD

The SEQFIELD attribute specifies the field you want to use as the sequencing value:

```
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = EMPLOYEES                         DICTIONARY =
SEGNAM=S01,ACCESS=ADBS,FILENO=001,DBNO=1,CALLTYPE=RL,
           SEQFIELD=PERSONNEL_ID,$
```

PASS

The PASS attribute specifies the Adabas password for the file. It is required if the file is password protected.

You must specify the password for each password-protected file you use. For example:

```
SEGNAM=ONE   ,FILENO=1   ,DBNO=1   ,PASS=ONEXX ,...,$
SEGNAM=PAY   ,FILENO=2   ,DBNO=3   ,PASS=USER2 ,...,$
```

FETCH

The FETCH attribute indicates whether to use the Fetch feature for segments described as ACCESS=ADBS in the Access File. Valid values are ON (default) or OFF.

FETCHSIZE

The FETCHSIZE attribute sets the number of records per buffer. You can supply a specific number or have the Adapter for Adabas automatically calculate the maximum number of records that fit in a 32K buffer.

The following is an example of an Access File that enables the Fetch feature and sets the number of records per buffer to 15.

```
$$$ FILENAME=EMPFILE1,SUFFIX=ADBSINX,$
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = EMPLOYEES                         DICTIONARY =
SEGNAM=S01,ACCESS=ADBS,FILENO=001,CALLTYPE=RL,
           SEQFIELD=PERSONNEL_ID,FETCH=ON,FETCHSIZE=15,$
```

Mapping Adabas Descriptors

In this section:

Adabas descriptors, superdescriptors, and subdescriptors must be declared in Master and Access Files in certain ways:

- When creating Master Files for fields that are descriptors, use the INDEX=I attribute. The Access File must also contain information about the descriptors, specifying the TYPE as DSC, SPR, or NOP. Identifying descriptors in the Access File directs the server to inverted lists to speed retrieval of these fields.

- Adabas fields described as superdescriptors must be declared in the Master File using the GROUP attribute. In the Access File, the TYPE must be SPR.

Specifying INDEX=I

Descriptors act as key values associated with a single field in an Adabas record. They enable records to be retrieved more efficiently based on the selection of a value, a set of values, or a range of values.

Additionally, all descriptors, superdescriptors, and subdescriptors are used to establish logical relationships between files. There are two methods for establishing logical relationships:

- By establishing the relationship dynamically at runtime with the JOIN command. Refer to *Adapter Navigation* on page 2-105 for more information about using Adabas files that are to be cross-referenced using JOIN.
- By using a static cross-reference in the Master and Access Files. See *Implementing Embedded JOINs: KEYFLD and IXFLD* on page 2-65 for information on how to define shared files.

Adabas descriptor fields are identified in the Master File using the INDEX=I attribute. For example:

```
FIELD=MAKE ,ALIAS=AD ,USAGE=A20 ,ACTUAL=A20 ,INDEX=I ,$
```

INDEX=I is optional in the Master File, but using it is recommended so that keys in your Master File are easily recognized without reference to the Access File.

Describing Superdescriptors

A superdescriptor is a key value associated with all or part of two to twenty Adabas fields. If the superdescriptor consists of a group of complete fields, it is declared differently than a superdescriptor which consists of one or more partial fields. For information on superdescriptors consisting of partial fields, see *Customizing the Adabas Environment* on page 2-78.

Superdescriptors can be printed if they contain only alphabetic fields.

Specifying Superdescriptors Using the GROUP Attribute

A superdescriptor composed of several complete fields is described with the GROUP attribute. All the fields that are part of the group must be specified immediately after the GROUP atttribute in the order in which they appear in the superdescriptor.

The following example shows a superdescriptor defined in server terminology:

```
GROUP=TRANS         ,ALIAS=S1 ,USAGE=A20    ,ACTUAL=A20 ,INDEX=I ,$
   FIELD=ACCT_NO    ,ALIAS=C2 ,USAGE=A9     ,ACTUAL=A9  ,$
   FIELD=ITEM_NO    ,ALIAS=GD ,USAGE=A5     ,ACTUAL=A5  ,$
   FIELD=TRANS_DATE ,ALIAS=D7 ,USAGE=A6YMD  ,ACTUAL=A6  ,$
```

The group field name, in this case, is TRANS. Since it is a superdescriptor, the INDEX=I attribute is included on the GROUP specification.

The ALIAS is the two-character Adabas field name for the superdescriptor.

The USAGE and ACTUAL values must be specified in alphanumeric format. See the next topic on calculating group length for more information.

If any of the components of the TRANS group field were descriptors, they could also contain the INDEX=I attribute.

Note: TYPE=SPR must be defined in the Access File. For more information, see *Customizing the Adabas Environment* on page 2-78.

Calculating GROUP Length

For a group field, you must supply both the USAGE and ACTUAL values in alphanumeric format.

If the component fields of the group are alphanumeric, the USAGE and ACTUAL lengths of the group field must be exactly the sum of the component field lengths. For example, in the *Specifying Superdescriptors Using the GROUP Attribute* on page 2-61, the lengths specified in the USAGE and ACTUAL attributes for the component fields ACCT_NO, ITEM_NO, and TRANS_DATE each total 20. The lengths specified in the USAGE and ACTUAL attributes of the TRANS group field are consequently both equal to 20.

If the component fields of the group are *not* alphanumeric:

- The USAGE and ACTUAL formats of the group field must still be alphanumeric.
- The ACTUAL length of the group field is still the sum of the component field lengths.
- The USAGE length is the sum of the internal storage lengths of the component fields.

You determine these internal storage lengths as follows:

USAGE	Length
A (alphanumeric)	Value equal to the number of characters contained in the field
D (decimal, double-precision)	8
F (decimal, single precision)	4
I (integer)	4
P (packed decimal):	
P*n* or P*n.d*, where *n* is less than or equal to 15	8
P16	16
P16.*d*	8
P*n* or P*n.d*, where *n* is greater than 16 or less than or equal to 31	16
Z (zoned)	Value equal to the number of characters contained in the field

For example, consider the following GROUP specification in the EMPLOYEES Master File:

```
GROUP=LEAVE_DATA     ,ALIAS=A3 ,A16 ,A4 ,$
  FIELD=LEAVE_DUE    ,ALIAS=AU ,P2  ,Z2 ,$
  FIELD=LEAVE_TAKEN  ,ALIAS=AV ,P2  ,Z2 ,$
```

In this example:

- The lengths of the ACTUAL values for the component fields LEAVE_DUE and LEAVE_TAKEN total 4, which is the length of the ACTUAL value for the group field.
- The length of the USAGE value for the group field is determined by adding the internal storage lengths of the component fields LEAVE_DUE and LEAVE_TAKEN, as specified by the field types: a length of 8 for USAGE P2 (8 plus 8 = 16).

Here is a GROUP specification:

```
GROUP=MODEL_YEAR_MAKE ,ALIAS=AO ,A28 ,A22 ,INDEX=I ,$
  FIELD=YEAR          ,ALIAS=AG ,P2  ,Z2  ,$
  FIELD=MAKE          ,ALIAS=AD ,A20 ,A20 ,INDEX=I ,$
```

In this example:

- The lengths of the ACTUAL values for the component fields YEAR and MAKE total 22, which is the length for the ACTUAL value for the group field.
- The length of the USAGE value for the group field is determined by adding the internal storage lengths of the component fields YEAR and MAKE, as specified by the field types: a length of 8 for USAGE P2 and a length of 20 for USAGE A20 (8 plus 20 = 28).

Describing Descriptors in the Access File

Field suffixes are specified in the Access File to identify descriptors, superdescriptors, and subdescriptors.

The syntax is

```
FIELD[NAME]=fieldname, TYPE=fieldsuffix, [NU={YES|NO}] ,$
```

where:

fieldname

Is the field name or group in the Access File.

fieldsuffix

Indicates the field suffix. Possible values are:

DSC which indicates a descriptor.

SPR which indicates a superdescriptor made up of whole fields.

NOP which indicates a subdescriptor or superdescriptor made up of partial fields.

PDS which indicates a phonetic descriptor.

HDS which indicates a hyperdescriptor.

blank which indicates non-descriptor fields.

NU

Specifies whether null-suppression is in use. Possible values are:

YES which indicates that the field is described in the adapter with null-suppression.

Note: Descriptors with null values are not stored in inverted lists.

NO which indicates null-suppression is not used. NO is the default value.

Specifying Superdescriptors Containing Partial Fields

If a superdescriptor consists of one or more partial fields, that superdescriptor (field) is defined with TYPE=NOP (non-printable) in the Access File. This type of descriptor can be used in selection tests, but neither it nor any part of it can be printed or displayed unless the field is alphanumeric.

The partial fields that comprise such superdescriptors are stored in Adabas; there is no facility or necessity for identifying them to the server or the adapter. If you look at an Access File that contains a superdescriptor composed of partial fields, you would recognize it by TYPE=NOP, but you cannot list the partial fields that are its components. To use the adapter, identify such a superdescriptor in the Access File with TYPE=NOP.

Any superdescriptor defined to the server with TYPE=NOP has the following limitation: CALLTYPE=RL is not supported when this field is the IXFLD in an embedded cross-reference or JOIN.

See your Software AG documentation for more information about using superdescriptors containing partial fields.

Specifying Subdescriptors

A subdescriptor consists of a partial field value for which an index has been created in Adabas. It is described at the field level in the Access File as TYPE=NOP. It can be used in selection tests, but it cannot be printed.

Any subdescriptor defined to the server in the Access File with TYPE=NOP has the following limitation: CALLTYPE=RL is not supported when this field is the IXFLD in an embedded cross-reference or JOIN.

Specifying Phonetic Descriptors

A phonetic descriptor consists of similar phonetic values for which an index has been created in Adabas. It is described at the field level in the Access File as TYPE=PDS. It can be used in selection tests, but it cannot be printed.

Specifying Hyperdescriptors

A hyperdescriptor consists of values generated, based on a user-supplied algorithm, for which an index has been created in Adabas. It is described at the field level in the Access File as TYPE=HDS. It can be used in selection tests, but it cannot be printed.

Any phonetic descriptor or hyperdescriptor defined to the server in the Access File with TYPE=PDS/HDS has the following limitation: CALLTYPE=RL is not supported when this field is the IXFLD in an embedded cross-reference or JOIN.

If you have selected a field defined with TYPE=NOP (for example, subdescriptors), TYPE=PDS or TYPE=HDS, a FIND call is issued even if RL has been specified.

Specifying Null-Suppression

To allow the Adapter for Adabas to create the most efficient calls, while still maintaining integrity of the answer set, any null-suppressed fields that are components of a superdescriptor must be defined in the Access File. The NU attribute is used to define a null-suppressed field.

An NU field defined as a descriptor is not stored in inverted lists when it contains a null value. Any qualifying descriptor records containing a null value are not recognized by a FIND command that refers to that descriptor.

The same is true for subdescriptors and superdescriptors derived from fields described in the adapter with null-suppression. No entry is made for a subdescriptor if the bytes of the field from which it is derived contain a null value and that field is defined with null-suppression (NU). No entry is made for a superdescriptor if any of the fields from which it is derived is an NU field with a null value.

For more information about null-suppression and how it affects data retrieval, see your Software AG documentation.

Implementing Embedded JOINs: KEYFLD and IXFLD

The KEYFLD and IXFLD attributes identify the common fields for parent/descendant relationships in a multi-segment Master File. These relationships are referred to as embedded JOINs or server views.

For each descendant segment, the KEYFLD and IXFLD attributes specify the field names of the shared field that implements the embedded JOIN. The parent field supplies the value for cross-referencing; the descendant field contains the corresponding value. The adapter implements the relationship by matching values at run time.

The KEYFLD and IXFLD attributes are valid only for ACCESS=ADBS segments.

Attribute	Description
`KEYFLD`	Field name in the parent segment whose value is used to retrieve the child segment. This is also known as the primary key.
`IXFLD`	Field name in the child or cross-referenced segment containing the related data. This is also known as the foreign key.

The value for the KEYFLD attribute is a 1- to 66-character field name or alias from the parent segment. The value for the IXFLD attribute is a 1- to 66-character field name or alias from the descendant segment. This is an example of a Master File with an embedded JOIN using KEYFLD and IXFLD.

```
FILENAME=AMKTORDR     ,SUFFIX=ADBSINX ,$
SEGNAME=MKTORDER
FIELDNAME=NMARKET_GRP     ,BA          ,I3      ,I2,    INDEX=I,$
FIELDNAME=QPRODUCT        ,BB          ,I3      ,I2        ,$
FIELDNAME=QNEEDITM        ,BC          ,A3      ,I2        ,$
FIELDNAME=FBUILD          ,BD          ,A1      ,A1        ,$
FIELDNAME=FK_NPRODUCT     ,BE          ,A4      ,A4        ,$
FIELDNAME=FK_NCUSTOMER    ,BF          ,I3      ,I2        ,$
FIELDNAME=DATEMKTO        ,BG          ,A8      ,A8        ,$
FIELDNAME=DATEFBLD        ,BH          ,A8      ,A8        ,$

SEGNAME=CUSTOMER,  SEGTYPE=U,PARENT=MKTORDER,$
FIELDNAME=NCUSTMR_GRP     ,AA          ,I3      ,I2,    INDEX=I,$
FIELDNAME=NAMECUST        ,AB          ,A15     ,A15       ,$
FIELDNAME=DCUSROAD        ,AC          ,A20     ,A20       ,$
FIELDNAME=DCUSTOWN        ,AD          ,A20     ,A20       ,$
```

This is an example of an Access File with an embedded JOIN using KEYFLD and IXFLD.

```
RELEASE=6,    OPEN=YES,$
  SEGNAM=MKTORDER,    ACCESS=ADBS,FILENO=022,DBNO=001 ,$
  SEGNAM=CUSTOMER,    ACCESS=ADBS,FILENO=021,DBNO=001 ,CALLTYPE=FIND,
IXFLD=NCUSTMR_GRP,KEYFLD=FK_NCUSTOMER,$
```

Note: Include the pair of attributes in the Access File segment declaration for descendant segments. Do not specify them in the segment declaration for the root segment.

A JOIN can be based on more than one field in the host and cross-referenced logical record types. If the Adabas file uses multiple fields to establish a relationship or link between logical record types, you can specify concatenated fields in an embedded JOIN. You can also specify multiple fields with the dynamic JOIN command.

In the multi-field embedded JOIN, the KEYFLD and IXFLD values consist of a list of their component fields separated by slashes (/). Additional Access File attributes are not required. The syntax is

```
KEYFLD=field1/field2/...,
IXFLD=cfield1/cfield2/...,$
```

where:

field1/...

Is a composite of up to 16 key fields from the parent segment. Slashes are required.

`cfield1/...`

Is a composite of up to 16 key fields from the descendant segment.

The adapter compares each field pair for the data formats prior to format conversion; it evaluates each field pair with the following rules:

- The cross-referenced field in the embedded JOIN must be an Adabas descriptor field. The host field can be any field.
- Parent and descendant fields must be real fields.
- All participating fields for either the parent or descendant file must reside in the same segment.
- KEYFLD and IXFLD attributes must specify the same number of fields. If the format of the parent field is longer than the format of the descendant field, its values are truncated. If the format of the parent field is shorter, its values are padded with zeros or blanks.
- The KEYFLD and IXFLD attributes can consist of a maximum of 16 concatenated fields in order of high-to-low significance. You must specify the field order: the order determines how the values will be matched.
- The pair of fields in the KEYFLD/IXFLD attribute does not have to have comparable data formats.

To implement JOINs, the Adabas DBMS converts the alphanumeric field formats on the server to equivalent Adabas field formats in order to perform the necessary search and match operations. When the Adabas DBMS returns the answer set of matched values, it also converts the values back to alphanumeric formats. The cross-referenced fields must be descriptor fields.

Mapping Adabas Files With Variable-length Records and Repeating Fields

In this section:

Describing Multi-value Fields and Periodic Groups With the OCCURS Attribute

Describing Multi-value Fields Within Periodic Groups

Specifying OCCURS Segment Sequence: The ORDER Field

The OCCURS segment attribute is used by the server to describe MU and PE field types in a Master File.

Consider the sample Adabas file VEHICLES. The FDT for the VEHICLES file is shown below.

```
*********************************
* FILE     2 (VEHICLES        ) *
*********************************
FIELD DESCRIPTION TABLE

      I      I        I        I              I                   I
LEVEL I NAME I LENGTH I FORMAT I  OPTIONS     I  PARENT OF        I
      I      I        I        I              I                   I
------I------I--------I--------I--------------I-------------------I
      I      I        I        I              I                   I
  1   I  AA  I   15   I    A   I DE,UQ,NU     I                   I
  1   I  AB  I    4   I    F   I FI           I                   I
  1   I  AC  I    8   I    A   I DE           I                   I
  1   I  CD  I        I        I              I                   I
  2   I  AD  I   20   I    A   I DE,NU        I SUPERDE           I
  2   I  AE  I   20   I    A   I NU           I                   I
  2   I  AF  I   10   I    A   I DE,NU        I                   I
  1   I  AG  I    2   I    U   I NU           I SUPERDE           I
  1   I  AH  I    1   I    A   I DE,FI        I                   I
  1   I  AI  I    1   I    A   I FI           I                   I
  1   I  AJ  I    6   I    U   I NU           I SUPERDE           I
  1   I  AK  I        I        I              I                   I
  2   I  AL  I    3   I    A   I NU           I                   I
  2   I  AM  I    4   I    P   I MU,NU        I                   I
      I      I        I        I              I                   I
-------------------------------------------------------------------
```

```
SPECIAL DESCRIPTOR TABLE

       I      I        I        I                I                I
 TYPE  I NAME I LENGTH I FORMAT I    OPTIONS     I   STRUCTURE    I
       I      I        I        I                I                I
-------I------I--------I--------I----------------I----------------I
       I      I        I                         I                I
 SUPER I  AN  I    4   I    B   I DE,NU          I AJ (  5 -   6) I
       I      I        I        I                I AJ (  3 -   4) I
 SUPER I  AO  I   22   I    A   I DE,NU          I AG (  1 -   2) I
       I      I        I        I                I AD (  1 -  20) I
       I      I        I        I                I                I
-------------------------------------------------------------------
```

The MU field, MAINT_COST, is a field in VEHICLES. It is allocated to a new segment, AM0101, required by the adapter, as shown in the Master File for a partial view of the VEHICLES file, illustrating the use of OCCURS=*occursname*:

```
$$$ CREATED ON 12/10/97 AT 10.17.27 BY PMSMJB
FILENAME=ADACAR,SUFFIX=ADBSINX,$

$ ADABAS FILE = VEHICLES-FILE                     DICTIONARY = 6
SEGNAME=S01     ,SEGTYPE=S,$

 FIELD=REG_NUM                    ,ALIAS=AA      ,A15 ,A15  ,INDEX=I,$
 FIELD=CHASSIS_NUM                ,ALIAS=AB      ,I9  ,I4   ,$
 FIELD=PERSONNEL_ID               ,ALIAS=AC      ,A8  ,A8   ,INDEX=I,$
 FIELD=CLASS                      ,ALIAS=AH      ,A1  ,A1   ,INDEX=I,$
 FIELD=LEASE_PUR                  ,ALIAS=AI      ,A1  ,A1   ,$
 FIELD=DATE_ACQ                   ,ALIAS=AJ      ,P6  ,Z6   ,$
$GRMU=CAR_MAINTENANCE             ,ALIAS=AK      ,A11 ,A7   ,$
  FIELD=CURR_CODE                 ,ALIAS=AL      ,A3  ,A3   ,$
  FIELD=MAINT_COST_CNT            ,ALIAS=AMC     ,I4  ,I2   ,$
 FIELD=DAT_ACQ_DESC               ,ALIAS=AN      ,A4  ,A4   ,INDEX=I,$
 GROUP=MODEL_YEAR_MAKE            ,ALIAS=AO      ,A28 ,A22  ,INDEX=I,$
  FIELD=YEAR_S02                  ,ALIAS=AG      ,P2  ,Z2   ,$
  FIELD=MAKE_S02                  ,ALIAS=AD      ,A20 ,A20  ,INDEX=I,$

SEGNAME=AM0101 ,SEGTYPE=S,PARENT=S01  ,OCCURS=AMC,$  MAX= 60
 FIELD=MAINT_COST                 ,ALIAS=AM      ,P7  ,P4  ,$
 FIELD=AM0101_OCC                 ,ALIAS=ORDER   ,I4  ,I2  ,$
```

In the original segment, the MAINT_COST_CNT (ALIAS=AMC) field contains the total number of occurrences of the MAINT_COST field. The new segment, AM0101, contains the MAINT_COST data field and the newly created AM0101_OCC field. The AM0101_OCC field contains the position of each occurrence.

This example includes the group field called CAR_MAINTENANCE, which contains the MU field, MAINT_COST. You cannot report on a group with an unknown length. However, the component fields can be referenced for reporting purposes.

Note: The CREATE SYNONYM facility creates the new segment, AM0101, for you. Otherwise, you must create the new segment using the OCCURS segment attribute.

The following is the Access File for the VEHICLES file:

```
$$$ CREATED ON 12/10/03 AT 10.17.27 BY PMSMJB
$$$ FILENAME=ADACAR,SUFFIX=ADBSINX,$
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = VEHICLES-FILE                     DICTIONARY = 6
SEGNAM=S01   ,ACCESS=ADBS,FILENO=002,
              CALLTYPE=RL,SEQFIELD=PERSONNEL_ID,$
 FIELD=DAT_ACQ_DESC                         ,TYPE=NOP,$
 FIELD=MODEL_YEAR_MAKE                      ,TYPE=SPR,$
  FIELD=YEAR_S02                            ,TYPE=    ,NU=YES,$
  FIELD=MAKE_S02                            ,TYPE=DSC,NU=YES,$
SEGNAM=AM0101,ACCESS=MU  ,FILENO=002,$ MAINT_COST
```

A new segment is created for the MU field. Note that for this new segment, ACCESS=MU.

Here is an example of the counter fields. The AMC field contains the total count of occurrences, the AM field contains the actual data, and the ORDER field contains the position of each occurrence.

```
     (AMC)             (AM)         (ORDER)
MAINT_COST_CNT    MAINT_COST    AM0101_OCC
--------------    ----------    ----------
             1           520             1
----------------------------------------
             1           210             1
----------------------------------------
             4            44             1
             4           322             2
             4            66             3
             4           188             4
----------------------------------------
             6           324             1
             6          1103             2
             6           566             3
             6           755             4
             6           988             5
             6          1899             6
----------------------------------------
             2           344             1
             2           500             2
```

In some cases, the ORDER field may describe the months of the year (1 to 12). If all occurrences for the month of June are needed, for example, you would print data in which the ORDER field is equal to 6 (for June).

Using the preceding sample data, here is an example of how to use the ORDER field; this request always selects the sixth occurrence of maintenance costs:

```
TABLE FILE VEHICLES
PRINT MAINT_COST
WHERE AM0101_OCC EQ 6
END

MAINT_COST
----------
      1899
```

Using the same data, this request always selects the second occurrence of maintenance costs:

```
TABLE FILE VEHICLES
PRINT MAINT_COST
WHERE AM0101_OCC EQ 2
END

MAINT_COST
----------
       322
      1103
       500
```

Describing Multi-value Fields and Periodic Groups With the OCCURS Attribute

The OCCURS attribute is a segment attribute used to describe portions of records containing a multi-value field or a periodic group. When describing Adabas files, the OCCURS attribute may be equal to the field name or (another option) to the two-character internal Adabas name appended with the letter C.

Records with repeating fields are defined to the server by describing the singly occurring fields in the parent segment, and the multiply occurring fields or groups in their own subordinate segment. The OCCURS attribute appears in the declaration for the subordinate segments.

The syntax is

```
OCCURS=occursname ,$
```

where:

occursname

Indicates a data field in the parent segment that contains the number of occurrences of the descendant segment. Its value is derived from the two-character internal Adabas name (ALIAS field in the Access File) appended with the letter C.

The following is an example of the VEHICLES Master File, containing an MU segment with the counter field in the parent segment:

```
FILENAME=VEHICLES,SUFFIX=ADBSINX,$
  SEGNAME=S01      ,SEGTYPE=S,$
   FIELD=REG_NUM           ,ALIAS=AA          ,A15    ,A15   , INDEX=I,$
   FIELD=CHASSIS_NUM       ,ALIAS=AB          ,I9     ,I4    ,$
   FIELD=PERSONNEL_ID      ,ALIAS=AC          ,A8     ,A8    , INDEX=I,$
   FIELD=CLASS             ,ALIAS=AH          ,A1     ,A1    , INDEX=I,$
   FIELD=LEASE_PUR         ,ALIAS=AI          ,A1     ,A1    ,$
   FIELD=CURR_CODE         ,ALIAS=AL          ,A3     ,A3    ,$
   FIELD=MAINT_COST_CNT    ,ALIAS=AMC         ,I4     ,I2    ,$
   GROUP=MODEL_YEAR_MAKE   ,ALIAS=AO          ,A28    ,A22   , INDEX=I,$
    FIELD=YEAR             ,ALIAS=AG          ,P2     ,Z2    ,$
    FIELD=MAKE             ,ALIAS=AD          ,A20    ,A20   , INDEX=I,$
SEGNAME=AM0101  ,SEGTYPE=S ,PARENT=S01  ,OCCURS=AMC,$
   FIELD=MAINT_COST        ,ALIAS=AM          ,P7     ,P4    ,$
   FIELD=AM0101_OCC        ,ALIAS=ORDER       ,I4     ,I2    ,$
```

MAINT_COST is the MU field. It is allocated to the newly created segment, AM0101. The counter field, MAINT_COST_CNT, is located in the parent segment. The new segment, AM0101, uses this counter field to determine the number of occurrences of the MAINT_COST field (OCCURS=AMC).

Notice that the fields MAINT_COST_CNT and AM0101_OCC (ALIAS=ORDER) have USAGE=I4 and ACTUAL=I2. These are the required values for the USAGE and ACTUAL attributes of the counter field and the ORDER field.

The adapter builds the PE segment in the same way. However, the component fields within the PE group are defined in the new segment.

The following is a view of the EMPLOYEES Master File, containing a PE group:

```
FILENAME=EMPFILE1,SUFFIX=ADBSINX,$

SEGNAME=S01     ,SEGTYPE=S,$
 FIELD=PERSONNEL_ID                ,ALIAS=AA       ,A8     ,A8   ,INDEX=I,$
  FIELD=FIRST_NAME                 ,ALIAS=AC       ,A20    ,A20  ,$
  FIELD=NAME                       ,ALIAS=AE       ,A20    ,A20  ,INDEX=I,$
FIELD= LEAVE_BOOKED_CNT            ,ALIAS=AWC      ,I4     ,I2   ,$
SEGNAME=AW0401   ,SEGTYPE=S,PARENT=S01    ,OCCURS=AWC,$
 GROUP=LEAVE_BOOKED                ,ALIAS=AW       ,A16    ,A12  ,$
  FIELD=LEAVE_START                ,ALIAS=AX       ,P6     ,Z6   ,$
  FIELD=LEAVE_END                  ,ALIAS=AY       ,P6     ,Z6   ,$
 FIELD=AW0401_OCC                  ,ALIAS=ORDER    ,I4     ,I2   ,$
```

LEAVE_BOOKED is the PE group. It is allocated to the newly created segment, AW0401. The counter field, LEAVE_BOOKED_CNT, is located in the parent segment. The new segment, AW0401, uses this counter field to determine the number of occurrences of the LEAVE_BOOKED group (OCCURS=AWC).

The counter field, LEAVE_BOOKED_CNT, and the ORDER field, AW0401_OCC, require values of I4 for the USAGE attribute and I2 for the ACTUAL attribute.

Note: For both counter (CNT) and ORDER fields, Release 7.0.8 and higher supports ACTUAL=I2. Prior releases required ACTUAL=I1.

The number of occurrences is limited by the original definition of the file to Adabas.

To retrieve the appropriate occurrence of a repeating field in a report request, specify the ORDER field in your Master File. See *Specifying OCCURS Segment Sequence: The ORDER Field* on page 2-75 for details.

For detailed information about defining MU fields and PE groups in the Access File, see *Segment Attributes in Access Files* on page 2-52.

Describing Multi-value Fields Within Periodic Groups

This topic describes how to specify multi-value fields within periodic groups in the Master File. Define each multi-value field contained in the periodic group segment as a separate descendant segment of the periodic group.

A periodic group containing a multi-value field is defined as two separate segments.

All fields must be defined in the order in which they appear in the FDT.

Consider the EMPLOYEES file. It has a periodic group called INCOME, which contains one multi-value field, BONUS. Each counter field (for example, AQC) must be in its appropriate parent segment. The INCOME_CNT field is in the parent segment for the periodic group, INCOME. The BONUS_CNT field is in the parent segment for the MU field, BONUS. Any repeating fields within the periodic group are defined in the AQ0201 segment. Their corresponding Adabas two-character field names are the values for ALIAS with the appropriate values for USAGE and ACTUAL.

Here is the view of the Master File for this structure:

```
 FILENAME=EMPFILE1,SUFFIX=ADBSINX,$

   SEGNAME=S01     ,SEGTYPE=S,$
    FIELD=PERSONNEL_ID      ,ALIAS=AA         ,A8    ,A8   ,INDEX=I,$
     FIELD=FIRST_NAME       ,ALIAS=AC         ,A20   ,A20  ,$
     FIELD=NAME             ,ALIAS=AE         ,A20   ,A20  ,INDEX=I,$
1.  FIELD=INCOME_CNT        ,ALIAS=AQC        ,I4    ,I2   ,$

1. SEGNAME=AQ0201  ,SEGTYPE=S,PARENT=S01   ,OCCURS=AQC,$
1. $PEMU=INCOME             ,ALIAS=AQ         ,A19   ,A13  ,$
1.   FIELD=CURR_CODE        ,ALIAS=AR         ,A3    ,A3   ,$
1.   FIELD=SALARY           ,ALIAS=AS         ,P9    ,P5   ,$
2.   FIELD=BONUS_CNT        ,ALIAS=ATC        ,I4    ,I2   ,$
1.  FIELD=AQ0201_OCC        ,ALIAS=ORDER      ,I4    ,I2   ,$

2. SEGNAME=AT0301  ,SEGTYPE=S,PARENT=AQ0201,OCCURS=ATC,$
2.  FIELD=BONUS             ,ALIAS=AT         ,P9    ,P5   ,$
2.  FIELD=AT0301_OCC        ,ALIAS=ORDER      ,I4    ,I2   ,$
```

1. PE fields.

This example includes the periodic group called INCOME, which contains the MU field, BONUS. CREATE SYNONYM comments out the group field with $PEMU because its length is variable (depending on the number of occurrences of the MU field). The server cannot report on a group with an unknown length. However, the component fields can be referenced for reporting purposes.

2. MU fields.

BONUS_CNT (ALIAS=ATC) is the counter field. It is defined in the PE segment and tells us the number of occurrences for that field. Used in the MU segment, it indicates how many times this repeating field occurs, for example, OCCURS=ATC. Defined in the parent segment, the counter field is used by the child segment to tell us how many occurrences to expect.

Any periodic group segment must be described in its entirety. This description has two requirements: each field of the periodic group must have a field declaration in the periodic group segment, and each field in the group must be described in the order in which it appears in the Adabas FDT.

Specifying OCCURS Segment Sequence: The ORDER Field

There may be occasions with OCCURS segment processing when the order of the data is significant. For example, the field values may represent monthly or quarterly data, but the record itself may not explicitly specify the month (or quarter) to which the data applies. The ORDER field maintains the sequence number for each multiply occurring instance.

The order of the occurrences of a multi-value field or periodic group also has some significance in your application. For example, if a periodic group contains 12 occurrences, each occurrence could correspond to one month of the year. The first occurrence represents January, the second February, and so on through to December.

To use the ORDER option, you must include an additional field in your Master File with an ALIAS of ORDER. This option instructs the server to determine the sequence number of fields in an OCCURS segment. The server automatically supplies a value for the new field that defines the sequence number of each repeating group. The ORDER field does not represent an existing Adabas field; it is used only for internal processing.

The syntax rules for the ORDER field are:

- It must be the last field described in the OCCURS segment.
- The field name must be unique.
- The ALIAS value must be ORDER.
- The USAGE value must be I4, with any appropriate edit options.
- The ACTUAL value must be I2 (in earlier releases, I1 was used).

Each field with the ALIAS value ORDER tracks the order of occurrence of the preceding fields in the segment. Therefore, in the sample Master File in *Describing Multi-value Fields Within Periodic Groups* on page 2-74, AT0301_OCC maintains the sequence of BONUS occurrences.

An ORDER field may be decoded into a meaningful value in a DEFINE.

Using the GROUP Attribute to Cross-Reference Files

The GROUP attribute can be used to cross-reference Adabas files if the file you want to search does not contain a descriptor. This cross-referencing can be done *only* if fields within the file you want to search correspond to descriptors in the file you are cross-referencing.

Consider this situation: You have a SALES file which contains salesman information. It also has account information, such as COMPANY_CODE, ITEM_NUMBER, and ITEM_NAME.

Suppose you also have an ACCOUNT file containing information about each company in a salesman's territory. The ACCOUNT file has a superdescriptor consisting of COMP_CODE, IT_NUMBER, and IT_NAME. These fields correspond to the fields specified in the SALES file.

You can create a link between the matching fields in the SALES and ACCOUNT files. To do this, describe the matching fields from the SALES file as a group. Then you can join the group field to the superdescriptor in the ACCOUNTS file. The host file is the file, without a descriptor, containing the held values which must be grouped in order to retrieve related records in the second file. For example:

```
FILE=SALES    ,SUFFIX=ADBSINX    ,$
  SEGNAME=ACCT_SEG  ,SEGTYPE=S   ,$
    GROUP=LINKFLDS          ,ALIAS=   ,USAGE=A20 ,ACTUAL=A20 ,$
      FIELD=COMPANY_CODE    ,ALIAS=BA ,USAGE=A3  ,ACTUAL=A3  ,$
      FIELD=ITEM_NUMBER     ,ALIAS=BB ,USAGE=A5  ,ACTUAL=A5  ,$
      FIELD-ITEM_NAME       ,ALIAS=BC ,USAGE=A12 ,ACTUAL=A12 ,$
```

SALES is the host file which uses the superdescriptor (COMPANY) in the ACCOUNT file for the company-related data. The superdescriptor in the ACCOUNT file, easily recognized by TYPE=SPR in the Access File, is composed of fields that correspond to the dummy group in the host. Looking at the GROUP attribute used to cross-reference files, above, and the GROUP attribute used to cross-reference files, below, you see how the fields in the LINKFLDS group in the SALES file relate to the superdescriptor COMPANY in the ACCOUNT file:

```
FILE=ACCOUNT    ,SUFFIX=ADBSINX ,$
  SEGNAME=PROD_SEG  ,SEGTYPE=S  ,PARENT=ACCTS ,$
    GROUP=COMPANY     ,ALIAS=S1 ,USAGE=A20 ,ACTUAL=A20, INDEX=I,$
      FIELD=COMP_CODE ,ALIAS=AA ,USAGE=A3  ,ACTUAL=A3  ,$
      FIELD=IT_NUMBER ,ALIAS=AB ,USAGE=A5  ,ACTUAL=A5  ,$
      FIELD-IT_NAME   ,ALIAS=AC ,USAGE=A12 ,ACTUAL=A12 ,$
```

Use the GROUP superdescriptor COMPANY to retrieve data for those values using the JOIN command within a procedure. The JOIN command or embedded cross-reference completes the link. In this example, you would join LINKFLDS in SALES to COMPANY in ACCOUNT using the following syntax:

```
JOIN LINKFLDS IN SALES TO ALL COMPANY IN ACCOUNT AS J1
```

Platform-Specific Functionality

The Adapter for Adabas on UNIX, Windows, and OpenVMS, has more restrictions than on OS/390 and z/OS. These restrictions are described below.

Note: On all platforms, if an error occurs while Adabas commands are being processed, then the communication block (CB) is returned along with the Response Code (RC) Additions 2 field, in 2 or 4 bytes binary format. The Additions 2 field indicates the specific reason for the Response Code. The Adapter for Adabas displays both RC and Additions 2 fields in the message. This information is available in the trace as well.

Reference: Functionality on UNIX and Windows

The following conditions apply when using the adapter on UNIX and Windows platforms:

- You cannot use non-descriptor fields as search criteria.
- The parameter NDFIND is automatically set to the OFF value if the adapter is executed in a non-mainframe environment. If non-descriptor fields are used as key fields, the entire database is read.
- If the Master File non-descriptor field is declared as an indexed field (INDEX=I or FIELDTYPE=I) and used as a key field, the result from Adabas is RC 61.
- If an occurrence of an MU (multi-value) field is deleted, the adapter shifts all other occurrences into its place and assigns an empty value for the last occurrence. If this field has an NU (null suppression) option, the last occurrence will be automatically deleted by Adabas.
- DBNO must be assigned to a valid numeric value for the existing database.

Reference: Functionality for OpenVMS Platform

Adabas on OpenVMS does not allow a direct request for sub/superdescriptors values. For this reason, there are some limitations on using these fields (with TYPE=NOP/SPR in ACX) in server requests to Adabas:

- NOP fields cannot be printed or displayed, even if they are alphanumeric.
- NOP/SPR fields cannot be used in functions (MIN, MAX, and so on).
- If NOP/SPR fields are used in the IF/WHERE criteria, then CALLTYPE=RL is automatically changed to CALLTYPE=FIND. It can change the sequence of returning data rows on OpenVMS in comparison with sequence on other platforms.
- NOP fields cannot be used in a complex IF/WHERE clause.
- NOP fields cannot be used for joining.

Customizing the Adabas Environment

In this section:

The Adapter for Adabas provides several parameters for customizing the environment and optimizing performance.

The sections *Multifetch and Prefetch* on page 2-81, *Adabas Dynamic Database Number* on page 2-84, and *Running in 24-Bit Mode* on page 2-87, describe the environment commands that display and change the parameters governing your server session.

ORDER Fields in the Indexed Field List

ORDER fields maintain the sequence number for each multiply occurring instance. ON is the default value.

Setting ORDERKEYS to OFF forces the Adapter for Adabas to exclude ORDER fields from the list of indexed fields.

Syntax: How to Set ORDERKEYS in the Indexed Field List

```
ENGINE ADBSINX SET ORDERKEYS {ON|OFF}
```

Switching the Access Mode Using NOPACCESS

This setting allows you to switch the access mode from RL to FIND if field defined with TYPE=NOP was selected (even if CALLTYPE=RL has been specified).

The default value for this setting is MIXED. A new setting can be useful when the NOP superdescriptor defined in a root segment is derived from fields that belong to different Master File segments.

Syntax: How to Set NOPACCESS

```
ENGINE ADBSINX SET NOPACCESS FIND
```

NEW Synonym Setting for Superdescriptors

How to:

Set Synonyms for Superdescriptors

Example:

Setting Synonyms for Superdescriptors

Setting SYNONYM to NEW instructs the Adapter for Adabas to use new logic when:

- Describing metadata.
- Processing requests that can take advantage of superdescriptors for screening conditions.

Note: This command must be specified in the EDASPROF.PRF file.

Syntax: How to Set Synonyms for Superdescriptors

```
ENGINE ADBSINX SET SYNONYM {NEW|STD}
```

`NEW`

Describes superdescriptors as groups in the Master File. With NEW synonym format, neither the groups nor their components (fields) have field names specified in the Master File. Aliases are populated using the Adabas FDT (2-byte name).

When a synonym contains the NEW syntax, the Adapter for Adabas uses NEW logic to process the selection criteria of requests involving that synonym. Screening conditions within the request are analyzed and the superdescriptor that covers the highest order component fields required by the selection criteria are used.

Note that you must specify CALLTYPE=RL in the Access File in order to take advantage of superdescriptor-based access.

`STD`

Incorporates the usual CREATE SYNONYM behavior, which does not trigger the new logic for requests using superdescriptors. STD is the default value.

For more information on synonyms, see *Creating Synonyms* on page 2-19.

Example: Setting Synonyms for Superdescriptors

The following Master File illustrates superdescriptors Y6 and Y7 defined as GROUP, with the NEW synonym format:

```
GROUP=       ,ALIAS=Y6          ,A2    ,A2    ,INDEX=I,$
   FIELD=    ,ALIAS=AF          ,A1    ,A1    ,       ,$
   FIELD=    ,ALIAS=AG          ,A1    ,A1    ,       ,$
GROUP=       ,ALIAS=Y7          ,A22   ,A22   ,INDEX=I,$
   FIELD=    ,ALIAS=AF          ,A1    ,A1    ,       ,$
   FIELD=    ,ALIAS=AG          ,A1    ,A1    ,       ,$
   FIELD=    ,ALIAS=AC          ,A20   ,A20   ,       ,$
```

The corresponding Access File specifies the superdescriptors Y6 and Y7 and their components, using the keywords SUPER and NUMFLDS:

```
SUPER=Y6                ,NUMFLDS=2           ,$
  FIELD=AF_FIELD        ,TYPE=    ,NU=NO     ,$
  FIELD=AG_FIELD        ,TYPE=    ,NU=NO     ,$

SUPER=Y7                ,NUMFLDS=3           ,$
  FIELD=AF_FIELD        ,TYPE=    ,NU=NO     ,$
  FIELD=AG_FIELD        ,TYPE=    ,NU=NO     ,$
  FIELD=AC_FIELD        ,TYPE=    ,NU=NO     ,$
```

SUPER specifies the native Adabas two-byte field name of the superdescriptor.

NUMFLDS defines the number of participating fields. Descriptions of all participating fields immediately follow the SUPER statement.

The request that follows uses superdescriptor Y6 in the L3 (Read Logical) command in the Access File:

```
TABLE FILE EMP211
PRINT AF_FIELD AG_FIELD AC_FIELD AD_FIELD
IF AF_FIELD EQ 'S'
IF AG_FIELD EQ 'M'
END
```

The next request uses superdescriptor Y7 in the L3 (Read Logical) command. Notice that it references an extra field in the last line of the Access File:

```
TABLE FILE EMP211
PRINT AF_FIELD AG_FIELD AC_FIELD AD_FIELD
IF AF_FIELD EQ 'S'
IF AG_FIELD EQ 'M'
IF AC_FIELD EQ 'C'
END
```

Multifetch and Prefetch

How to:

Set FETCH and FETCHSIZE

Declare the FETCH and FETCHSIZE Attributes in the Access File

The Adabas Multifetch and Prefetch options reduce execution time and allow Adabas data to be retrieved with a high degree of efficiency. By buffering multiple record results from a single call, Multifetch and Prefetch reduce the communication overhead between the Adapter for Adabas and the Adabas nucleus.

Adabas Release 5.3.2 is the first release to support Multifetch, and Release 4.0 is the first release to support Prefetch. The adapter uses the Multifetch option if it is available (this is the default option), or the Prefetch option if it is available. The option available is determined by your site's environment.

The adapter trace file FSTRACE4 contains the following information about the Fetch option:

- Fetch is allowed or disallowed.
- Fetch technique being used (Multifetch or Prefetch).
- Number of records per Fetch buffer.
- Size of the Fetch buffer (ISN buffer).

The Multifetch and Prefetch options require that OPEN=YES (the default value) is specified in the Access File. The Multifetch and Prefetch options are activated as long as OPEN=YES is specified. There are two ways to deactivate (or reactivate) these options:

- Issue SET commands before running the request.
- Include the FETCH and FETCHSIZE attributes in the Access File.

Note: The SET commands override the Access File settings. This setting will be in effect for the entire server session.

Syntax: How to Set FETCH and FETCHSIZE

```
ENGINE ADBSINX SET FETCH {ON|OFF|DEFAULT}
ENGINE ADBSINX SET FETCHSIZE {n|MAX[IMUM]}
```

where:

`ADBSINX`

Indicates the Adapter for Adabas.

`ON`

Sets the Fetch feature on for the user session.

`OFF`

Sets the Fetch feature off for the user session.

`DEFAULT`

Uses the information from the Access File. DEFAULT is the default value.

`n`

Is the number of records per buffer (1–999). The buffer size varies with the size of the record and can be different for each TABLE request. The buffer size limit is 32K. 10 is the default value.

`MAX[IMUM]`

Is automatically calculated by the Adapter for Adabas to allow the maximum number of records that fit in a 32K buffer.

Syntax: How to Declare the FETCH and FETCHSIZE Attributes in the Access File

FETCH and FETCHSIZE are applicable only to segments described as ACCESS=ADBS in the Access File.

The Access File can contain the following attributes in the SEGNAM statement

```
FETCH={ON|OFF}
FETCHSIZE={n|MAX[IMUM]}
```

where:

`ON`

Sets the Fetch feature on for the user session. ON is the default value.

`OFF`

Sets the Fetch feature off for the user session.

`n`

Is the number of records per buffer (1 to 999). 10 is the default value.

`MAX[IMUM]`

Is automatically calculated by the Adapter for Adabas to allow the maximum number of records that fit in a 32K buffer.

Here is an example of the FETCH and FETCHSIZE attributes in the sample Access File for EMPLOYEES:

```
$$$ FILENAME=EMPFILE1,SUFFIX=ADBSINX,$
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = EMPLOYEES                         DICTIONARY =
SEGNAM=S01 ,ACCESS=ADBS,FILENO=001,CALLTYPE=RL,
             SEQFIELD=PERSONNEL_ID,FETCH=ON,FETCHSIZE=15,$
FIELD=DEPT_PERSON        ,TYPE=SPR,$
FIELD=DEPT_S03           ,TYPE=DSC,NU=YES,$
FIELD=NAME_S03           ,TYPE=   ,NU=NO,$
```

See your Software AG documentation for more information about the Multifetch/Prefetch feature.

Controlling Adabas Multifetch for Join Operations

You can use the FETCHJOIN command to take advantage of Multifetch efficiencies in Join operations for a session. This feature is particularly useful when you are joining to a base file in which sorting is based on the key field that is used for the join.

Syntax: How to Set Adabas FETCHJOIN

```
ENGINE ADBSINX SET FETCHJOIN {ON|OFF}
```

where:

`ADBSINX`

Indicates the Adapter for Adabas.

`ON`

Sets the FETCHJOIN feature on for the user session. ON is the default value.

`OFF`

Sets the FETCHJOIN feature off for the user session. This value is recommended when joining with a base file that is unsorted by the key field used for joining. This option avoids buffering and reduces processing time.

Adabas Dynamic Database Number

Previous versions of the Adapter for Adabas required specification of the Adabas database number in the Access File using the DBNO attribute. This feature required a duplicate copy of the Access File for each database accessed.

Adabas database numbers can now be set from the server's profile. A new SET command allows users to override the DBNO in the Access File. Note that specifying database numbers in the Access File is still supported. Use of the dynamic database number makes Master and Access Files shareable among databases.

The SET command remains valid throughout the server session.

Syntax: How to Set the Adabas Dynamic Database Number

```
ENGINE ADBSINX SET DBNO dbno
ENGINE ADBSINX SET DBNO dbno AFD afd
ENGINE ADBSINX SET DBNO dbno AFD afd SEG[NAM] segname
ENGINE ADBSINX SET ?
ENGINE ADBSINX SET DBNO DEFAULT
```

where:

`ADBSINX`

Indicates the Adapter for Adabas.

`dbno`

Is any valid numeric database value between 0 and 255.

`afd`

Is any valid Access File name.

`segname`

Is any valid ADBS segname in the Access File.

`?`

Queries the current settings.

`DEFAULT`

Returns to the default settings for all previous settings. The DBNO is read from the Access File. If the attributes are not specified in the Access File, the adapter will determine the DBNO from the DDCARD.

Examples for OS/390 and z/OS are shown below:

`ENGINE ADBSINX SET DBNO 5`	Retrieves all data from database number 5.
`ENGINE ADBSINX SET DBNO 2 AFD EMPFILE`	Retrieves data for the EMPFILE from database number 2.

Note: Several commands can be issued for different files. Each command must be issued separately. Changes are cumulative. For example:

```
ENGINE ADBSINX SET DBNO 1 AFD EMPFILE
ENGINE ADBSINX SET DBNO 2 AFD EMP2
ENGINE ADBSINX SET DBNO 3 AFD VEHICLES
```

`ENGINE ADBSINX SET DBNO 2 AFD EMPFILE SEG S02`	Retrieves data for the EMPFILE in database number 2 for a particular ADBS segment (segment S02).
`ENGINE ADBSINX SET ?`	Displays your settings.
`ENGINE ADBSINX SET DBNO DEFAULT`	Returns to the default settings. This resets all SET DBNO commands for all Access Files.

Controlling Data Retrieval Types

How to:

Set the Adabas Dynamic Database Number

Example:

Using Dynamic CALLTYPE

The SET CALLTYPE command enables you to set the data retrieval type per session, file, or segment by switching dynamically between physical and logical reads.

Syntax: How to Set Adabas Dynamic CALLTYPE

```
ENGINE ADBSINX SET CALLTYPE ctype
ENGINE ADBSINX SET CALLTYPE ctype AFD afd
ENGINE ADBSINX SET CALLTYPE ctype AFD afd SEGNAME segname
ENGINE ADBSINX SET ?
ENGINE ADBSINX SET CALLTYPE DEFAULT
```

where:

`ADBSINX`

Indicates the Adapter for Adabas.

`ctype`

Is any CALLTYPE value: FIND/RL/MIXED.

`afd`

Is any valid Access File name.

`segname`

Is any valid ADBS segname in the Access File.

`?`

Queries the current settings.

`DEFAULT`

Returns to the default settings for all previous settings. The CALLTYPE is read from the Access File.

Example: Using Dynamic CALLTYPE

The following examples illustrate the use of variations of dynamic CALLTYPE syntax:

```
ENGINE ADBSINX SET CALLTYPE FIND
```

Uses FIND as the type of data retrieval call for all files.

```
ENGINE ADBSINX SET CALLTYPE RL AFD EMPFILE
```

Retrieves data for the EMPFILE using RL as the type of data retrieval call.

Note that you can issue several commands for different files. Each command must be issued separately. Changes are cumulative. For example, the following commands

```
ENGINE ADBSINX SET CALLTYPE RL AFD EMPFILE

ENGINE ADBSINX SET CALLTYPE FIND AFD EMP2

ENGINE ADBSINX SET CALLTYPE MIXED AFD VEHICLES

ENGINE ADBSINX SET CALLTYPE FIND AFD EMPFILE SEGNAME S02
```

retrieve data for the EMPFILE using FIND as the type of data retrieval call for a particular ADBS segment (segment S02).

```
ENGINE ADBSINX SET ?
```

Displays your settings.

```
ENGINE ADBSINX SET CALLTYPE DEFAULT
```

Returns to the default settings. This command resets all SET CALLTYPE commands for all Access Files.

Running in 24-Bit Mode

The following command is used to allocate buffers below the line because of certain user exits that must run below the line. Issue this command in your server session:

```
ENGINE ADBSINX SET AMODE 24
```

Adabas Reporting Considerations

In this section:

Designers have great flexibility in coding the description of the part of the Adabas structure that they want to access. These topics describe the methods of file traversal that the Adapter for Adabas uses to determine the relative advantages of different file descriptions.

Adabas File Navigation Techniques

When you define specific hierarchical representations of Adabas structures in Master Files, you specify the order in which you want the Adapter for Adabas to retrieve records. This procedure is called navigational logic and is usually part of the application program. Navigation techniques using the JOIN command also work this way.

Referencing Subtrees and Record Retrieval

The Adapter for Adabas selects where to enter the subtree, called the point-of-entry, and the subsequent navigational processing by analyzing the tree structure defined by your Master File (or JOIN structure) and report request. The adapter determines the smallest subtree that contains all the fields needed for retrieval to produce a report.

The smallest subtree is composed of those segments that contain fields referenced by the request, plus the minimum number of additional segments required to connect all the files used in the request.

The adapter retrieves records only in segments in the referenced subtree. Within the subtree, it retrieves records that contain fields required for the report request or records that are needed to provide the correct links between report fields.

The adapter always enters a database through the root segment of the referenced subtree.

Segment Retrieval Methodology

The Adapter for Adabas retrieves segments from top-to-bottom, then left-to-right at each level of the hierarchy. It retrieves all unique descendant segments before any non-unique descendant segments.

This treatment of unique segments is consistent with a basic server principle: for reporting purposes (though not for updating or file organization), a unique child is logically a direct extension of its parent. This principle is an important factor in selecting a structure that properly reflects your Adabas file. The results of SUM and COUNT operations on fields in child segments may depend on whether they have been declared unique or non-unique. The server also treats missing segments differently, depending on whether the segment is declared unique or non-unique.

Missing Unique Segments

If a segment is specified as unique (SEGTYPE=U or KU), the server regards it as a logical extension of the parent segment. The Adapter for Adabas automatically inserts default values (blanks for alphanumeric fields and zeros for numeric fields) if the unique child segment does not exist. As a result, unique segments are always present.

Syntax: How to Use Missing Non-Unique Segments

If a segment is specified as non-unique (SEGTYPE=S or KM), select one of three options for processing a record without descendant segments.

```
SET ALL=all_option
```

where:

all_option

Allows for the processing of records with no descendant segments. Possible values are:

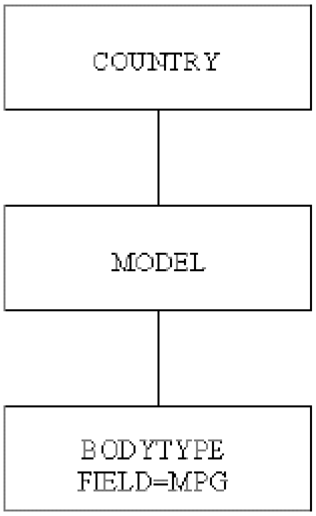
`OFF` which omits parent instances that are missing descendant segments from the report. OFF is the default value.

`ON` which includes parent instances that are missing descendant segments in the report.

`PASS` which includes parent instances that are missing descendant segments, even when IF statements exist to screen fields in the descendant segment's missing instances.

You can specify SET ALL in a profile or procedure.

The examples in the following topics describing the SET ALL command are based on the following structure:



SET ALL=OFF

The default option (SET ALL=OFF) rejects a record if the request calls for retrieval of a descendant segment and there is no descendent segment present.

For example, assume you have a file in which the parent segment is COUNTRY, which has a descendant segment named MODEL, which in turn has a descendant segment named BODYTYPE. Using the SET ALL=OFF option, the statement

```
COUNT BODYTYPE BY COUNTRY
```

does not print in the report the details of any country that did not produce at least one bodytype of a model of a car.

SET ALL=ON

The Adapter for Adabas displays the parent record, even if it has no descendant segments. In this case, using the SET ALL=ON option when processing the statement

```
COUNT BODYTYPE BY COUNTRY
```

displays the names of all countries and gives a count of zero (0) bodytypes for those without descendant segments.

However, if the request has an explicit screening test on the descendant segment, the absence of any descendant segments results in test failure. For example, the request

```
COUNT BODYTYPE BY COUNTRY
IF MPG GT 22
```

excludes any country without any bodytype segments from the report.

SET ALL=PASS

The third option, SET ALL=PASS, allows parents without a descendant segment to pass an explicit screening test on that descendant segment. The request

```
COUNT BODYTYPE BY COUNTRY
```

lists all countries with or without bodytype segments. The request

```
COUNT BODYTYPE BY COUNTRY
IF MPG GT 22
```

includes records without any bodytype segments, and those with an MPG greater than or equal to 22.

The ALL Prefix

Selectively apply SET ALL=ON by adding the ALL prefix to any field from the desired segment.

Reference the field either as a sort field (for example, BY ALL.COUNTRY or ACROSS ALL.COUNTRY) or as a verb object (WRITE ALL.COUNTRY). As a result, the SET ALL=ON strategy is applied to any missing, immediate, non-unique descendants of the segment containing the ALL-prefixed field. The SET ALL=OFF option is in effect for all other segments.

For example, in the request

```
COUNT MODEL AND BODYTYPE BY ALL.COUNTRY
```

the SET ALL=ON option applies to the country segment and its descendant segments. Therefore, if there is a country without models (and consequently without bodytypes), the report shows that country. Any test condition on either the model or the bodytype segment nullifies the effect of the ALL prefix.

The global SET ALL settings of ON and PASS take precedence over the selective ALL prefix. The selective ALL prefix is effective only when the global setting is OFF, either explicitly or by default.

Adabas Selection Considerations

The Adapter for Adabas analyzes all selection criteria that apply to a specific report request and uses the criteria to minimize its search for data. If a record fails any of the selection tests, the server does not attempt to retrieve any descendant records. Retrieval continues with the next record in the parent segment. If there is no other record in that parent segment and it is not the root of the Master File, the server moves back up to the next record in the previous segment.

The selection tests that you impose on a high-level segment are much more efficient at reducing I/O operations than criteria imposed on lower level segments. If you know in advance which selection criteria are likely to be used most frequently, design the Master File to take advantage of the hierarchical structure in the Adapter for Adabas.

Selection Order in the Access File

How to:

Use RECORDLIMIT and READLIMIT

Use Optimization of the FIND Call Using Non-Descriptor Fields

When a report request contains multiple optimizable selection tests, the order of descriptors in the Access File determines the order in which the server applies the selection tests. The server issues a Read Logical (RL) call using the first descriptor listed in the Access File that participates in a selection test.

Therefore, for efficient processing you should describe the most restrictive descriptor at the beginning of its segment in the Access File. The order of descriptors in the Master File has no effect on selection processing.

Syntax: How to Use RECORDLIMIT and READLIMIT

For any request, you may limit the number of Adabas records retrieved from the database and the number of read operations performed. Use the RECORDLIMIT and READLIMIT keywords to impose these limitations by adding the following conditions to a report request

```
{WHERE|IF} RECORDLIMIT {EQ|IS} n
{WHERE|IF} READLIMIT {EQ|IS} n
```

where:

n

Is the number of records or read operations at which you want to limit the search.

You add these conditions to any report request, or incorporate them into file security through the DBA facility.

Consider the following example

```
TABLE FILE VEHICLES
PRINT PERSONNEL_ID MAKE MODEL YEAR
WHERE RECORDLIMIT EQ 5
END
```

which produces this report:

```
PAGE      1

PERSONNEL_ID    MAKE              MODEL                 YEAR
------------    ----              -----                 ----
11500330        CITROEN           BX LEADER               85
11400313        ALFA ROMEO        SPRINT GRAND PRIX       84
30034217        AUSTIN            MINI                    80
30034228        TALBOT            SOLARA                  83
30008427        AUSTIN            MINI                    80
```

Notice that only five records are reported as requested.

Syntax: How to Use Optimization of the FIND Call Using Non-Descriptor Fields

It is possible to use non-descriptor fields as search criteria and have the calls to Adabas use the search buffer rather than read through the entire database.

The Adapter for Adabas provides improved optimization by allowing the search buffer to be generated using non-descriptor fields. This optimization occurs whenever CALLTYPE=FIND is specified in the Access File and you include an IF or WHERE test referencing a non-descriptor field in your report request.

It may prove to be more efficient to alter your retrieval strategy and perform a Read Physical (L2) call when large amounts of data exist. A SET command is provided for changing the default Adabas call when selecting non-descriptor fields.

```
ENGINE ADBSINX SET NDFIND {ON|OFF}
```

where:

`ENGINE`

Indicates the Adapter for Adabas.

`ON`

Causes the search buffer to be generated with any field (non-descriptor and/or descriptor field). ON is the default value.

`OFF`

Causes the search buffer to be generated with only descriptor fields. If the request does not use any descriptor field, the Read Physical call is generated.

This command applies only if CALLTYPE=FIND is specified in the Access File.

Using the JOIN Command to Process Multiple Files

You can JOIN Adabas databases defined to the server to other Adabas databases or to any other fully joinable database that the server can read. Each JOIN creates a parent-child relationship. The parent field is called the host or *from* field. The child field is called the cross-referenced or *to* field.

You can JOIN to Adabas databases if the cross-referenced field is one of the following:

- A descriptor field.
- A superdescriptor defined with TYPE=SPR or NOP in the Access File.
- A subdescriptor defined with TYPE=NOP in the Access File.

In every case, in the Access File, the cross-referenced segment must specify ACCESS=ADBS.

If CALLTYPE=RL is specified for the cross-referenced segment, the host field can be joined to the high-order portion of a descriptor, superdescriptor, or subdescriptor.

When an Adabas file is the host file, the host field is one of the following:

- A non-recurring field (ACCESS=ADBS).
- A field in a periodic group (ACCESS=PE).
- A multi-value field (ACCESS=MU).

The Adapter for Adabas also supports DEFINE-based JOINs. Up to 16 JOINs can be in effect at one time.

Example: Multi-field JOIN and Short-to-Long JOIN Capability

For a multi-field JOIN, the number of fields used in the JOIN must be the same for both the host and the cross-referenced files. The cross-referenced fields must describe the left-most portion of a superdescriptor defined to the server using the GROUP attribute. Consider the following example.

```
JOIN FLDA AND FLDB IN ADBS1 TO KEY1 AND KEY2 IN ADBS2 AS J1
```

For the short-to-long JOIN, the cross-referenced field must be a descriptor, subdescriptor, or superdescriptor, or a field that describes the left-most portion of a GROUP superdescriptor.

Adabas Optimization With Null-Suppression for CALLTYPE=RL

In the Adapter for Adabas, optimization refers to using an index to retrieve the answer set. To ensure data integrity and complete answer sets, the Adapter for Adabas will perform optimization when all:

- Non-referenced component fields of a superdescriptor are *not* null-suppressed (NU=NO in the Access File).
- Component fields of a superdescriptor that contain null-suppressed fields (NU=YES in the Access File) are used in the selection criteria.

If a field is null-suppressed in Adabas (NU=YES in the Access File), any superdescriptor that uses this field has no entry on the inverted list when this field is blank (alphanumeric) or zero (numeric).

Master File With a Three-Field Superdescriptor

```
GROUP=SUPERG  ,ALIAS=S1 ,USAGE=A9 ,ACTUAL=A9 ,INDEX=I ,$
   FIELD=FLD1 ,ALIAS=AA ,USAGE=A3 ,ACTUAL=A3 ,$
   FIELD=FLD2 ,ALIAS=AB ,USAGE=A3 ,ACTUAL=A3 ,$
   FIELD=FLD3 ,ALIAS=AC ,USAGE=A3 ,ACTUAL=A3 ,$
```

Access File With a Three-Field Superdescriptor

```
FIELD=SUPERG    ,TYPE=SPR,$
   FIELD=FLD1  ,TYPE=   ,NU=YES ,$
   FIELD=FLD2  ,TYPE=   ,NU=NO  ,$
   FIELD=FLD3  ,TYPE=   ,NU=YES ,$
```

In order to optimize a selection test against a superdescriptor with null-suppression, you must explicitly reference the null-suppressed field in the superdescriptor. If you do not reference the null-suppressed field and the field has no data, there is no record in the index and optimization would return no records. To ensure correct results, the server will not optimize the selection test if the null-suppressed field is not referenced.

For more information about null-suppression and how it affects data retrieval, see your Software AG documentation.

Adabas Optimization on Group Fields

If a report request contains IF or WHERE selection tests against one or more fields that describe the left-most portion of a GROUP descriptor, the Adapter for Adabas combines this request into a test that uses the superdescriptor for greater efficiency. If any of the component (or parent) fields of the superdescriptor are defined to Adabas with null-suppression, be sure to note this information in the Access File to ensure accuracy of reads.

For example, consider the following Master File extract:

```
GROUP=SUPERD ,ALIAS=SD ,USAGE=A8 ,ACTUAL=A8 ,INDEX=I ,$
   FIELD=PART1 ,ALIAS=P1 ,USAGE=A4 ,ACTUAL=A4 ,$
   FIELD=PART2 ,ALIAS=P2 ,USAGE=A4 ,ACTUAL=A4 ,$
```

If, in a report request, you include the following two tests,

```
WHERE PART1 EQ 'ABCD'
WHERE PART2 EQ 'EFGH'
```

these two tests are equivalent to the following syntax:

```
WHERE SUPERD EQ 'ABCD/EFGH'
```

This combination uses the superdescriptor's inverted list and optimizes the Adabas call. This optimization is performed only if all null-suppressed (NU=YES) superdescriptor components are explicitly referenced in IF or WHERE tests.

Test on Group Field With Numerics

If you are testing on a group that contains numeric fields, the test must contain the sign byte. The Adapter for Adabas passes only one value per numeric field, based on the preferred sign values in Adabas. A sign value of F is passed for positive numbers and a sign value of D is passed for negative numbers. This sign value reduces the number of calls sent to Adabas and also eliminates the need for Adabas to perform any logical sign translation.

For example:

```
GROUP=GROUP1  ,ALIAS=S1  ,USAGE=A9  ,ACTUAL=A14  ,INDEX=I  ,$
   FIELD=FLD1 ,ALIAS=AA  ,USAGE=A3  ,ACTUAL=A3   ,$
   FIELD=FLD2 ,ALIAS=AB  ,USAGE=P3  ,ACTUAL=P3   ,$
   FIELD=FLD3 ,ALIAS=AC  ,USAGE=A3  ,ACTUAL=A3   ,$
```

In a report request, you must include the sign for the P3 field:

Mainframe platforms:

```
WHERE GROUP1 EQ 'ABC/123F/XYZ'
```

Non-mainframe platforms:

```
WHERE GROUP1 EQ 'ABC/123/XYZ'
```

Adabas Writing Considerations

In this section:

Adabas Write Adapter

SQL Commands for Adapter for Adabas

The Adapter for Adabas is designed to support SQL update commands for an Adabas data source without the use of remote procedures. These topics describe the methods and rules related to write capability.

Adabas Write Adapter

Before you can use any commands that write to an Adabas data source, you must make the following changes to the Master and Access Files:

In the Master File:

- All segments must have SEGTYPE = S0 (for the Read Adapter, SEGTYPE=S is sufficient).
- Fields with ACTUAL format Z (zoned) must have a numeric USAGE format (P or I). Format A is supported only for reading an Adabas data source.

  Note: For the best performance, you can change ACTUAL format Z to ACTUAL format P in the Master File. In this case, you must also change the ALIAS attribute to contain the length and type. If the ALIAS is ff, and the field length is lll, the ALIAS attribute should be coded as:

  ```
  ALIAS='ff,lll,P'
  ```

  For example, consider the following field declaration:

  ```
  FIELD=LEAVE_DUE   ,ALIAS=AU   ,USAGE=P2   ,ACTUAL=P2   ,$
  ```

  You would edit this declarations as follows:

  ```
  FIELD=LEAVE_DUE   ,ALIAS='AU,2,P'   ,USAGE=P2   ,ACTUAL=P2   ,$
  ```

- If two or more fields in the Master File are synonyms (they refer to the same field in the data source and, therefore, have the same ALIAS attribute), only the first field encountered in the Master File can be used in INSERT and UPDATE commands. If a synonym other than the first is used in an INSERT command, it is ignored. If one is used in an UPDATE command, processing is terminated with the following error message:

  ```
  (FOC4565) IGNORED ATTEMPT TO CHANGE NONUPDATABLE FIELD
  ```

In the Access File:

- The segment attribute WRITE=YES indicates that values of fields from the segment may be modified.
- If a segment has the attribute ACCESS=ADBS, you can define a unique key by adding the following attribute:

  ```
  UNQKEYNAME=name
  ```

  where:

  name

  Is the name of the elementary or group field to be used as the unique key.

  The UNQKEYNAME attribute does not necessarily define the key described in the ADABAS FDT (the UQ option). The adapter uses it to decide which rules to apply in an INSERT, DELETE, or UPDATE command. If the UNQKEYNAME attribute does not correspond to the key described with the UQ option in the ADABAS FDT, Adabas and the adapter may not agree on whether a segment instance is unique. This can affect the results of INSERT commands. If this attribute is not present, the adapter uses the rules for modifying a segment with a non-unique key or no key. If it is present, the adapter uses the rules for modifying a segment with a unique key. Subsequent sections describe these rules.
- We recommend the use of CALLTYPE=FIND instead of CALLTYPE=RL.

When using the Write Adapter for Adabas, the adapter automatically sets the values of the following two options:

- ADABAS OPEN is set to YES.
- FETCH is set to OFF for all segments.

SQL Commands for Adapter for Adabas

The Adapter for Adabas supports the following commands:

- SQL INSERT
- SQL UPDATE

  Note: Certain types of fields listed in the Master File cannot be updated. For more information, see *Fields That Cannot be Updated* on page 2-100.
- SQL DELETE

The adapter issues a COMMIT after each INSERT, UPDATE, or DELETE call. If the Adabas process fails, the adapter issues a ROLLBACK. User rollback of the Adabas process is not supported for SQL commands.

Reference: Fields That Cannot be Updated

The SQL UPDATE command for the following types of fields is ignored:

- Counter fields (ALIAS=xxC).
- ORDER fields (ALIAS=ORDER).
- Group fields.
- Unique key fields (fields specified by the UNQKEYNAME attributes in the Access File).

The SQL UPDATE command for the following types of fields produces an error:

- Synonym fields.
- Fields created from subdescriptors or superdescriptors (as defined an the Access File).

Reference: Using Synonym Fields (Not Related to CREATE SYNONYM)

In a Master File, two or more field declarations can refer to the same Adabas field. Each duplicate field declaration after the first is called a synonym field. Such synonyms fields can be used in report commands, but cannot be used in write commands. The following actions occur as a result of using synonyms in write commands:

- Synonym fields used in SQL INSERT are ignored.
- Synonym fields used in SQL UPDATE cause processing to terminate and generate the message:

```
(FOC4565) IGNORED ATTEMPT TO CHANGE NONUPDATABLE FIELD
```

Syntax: How to Insert Records Into an Adabas Data Source

```
INSERT INTO mfname [(field1, field2, ...)]
VALUES ('value1','value2', ...)
```

where:

mfname

Is the name of the Master File to use.

field1, field2, ...

Is an optional list of fields to be inserted. If you omit the list, the value list must contain values for all fields in all segments in the Master File as long as the Master File only specifies a single path Adabas file. If the Master File is multi-path, the field list is required and can specify only a single path. Elementary fields not included in the field list have empty values in the data source. The field list can contain only elementary fields. Groups, subdescriptors, superdescriptors, and hyperdescriptors, cannot be used in the field list.

'value1', 'value2', ...

Is the list of values to be inserted. Each alphanumeric value must be enclosed in single quotation marks. If you specify the field name list, the value list only needs to specify, in field name list order, the values for the field names supplied. At a minimum, you must supply all the key fields for all the segments in the path to the target.

If a segment is not present, default values are generated for all fields that are not supplied, and the missing path segments are inserted along with the target segment.

Reference: Rules for Inserting Records Into an Adabas Data Source

- For a segment with a unique key:

  The key field value is used to insert the target segment. If any additional fieldname=value pairs are supplied for a segment in the path, they are used to qualify that path segment. If a segment with ACCESS=ADBS has a unique key or group of keys, only these key fields should be used in the INSERT command. All other fields should be added to the record using the UPDATE command.

- For a segment with a non-unique key or no key:

  If you want to insert an additional record for an existing key field, you must have at least one field in addition to the key field in the field list in order to make the record unique. If you do not, the insert is rejected.

- For segments with ACCESS=PE or MU, if a new occurrence is inserted, you must set the occurrence number (ORDER field) to 0 (zero indicates the next occurrence) or to a value greater than the number of existing occurrences. This new occurrence is always inserted after the last existing occurrence. For example, if a PE or MU segment has two existing occurrences, the next occurrence added is always the third. If the occurrence with the given number already exists, processing terminates with the following message:

  ```
  (FOC4564) THIS OCCURRENCE ALREADY EXISTS. USE UPDATE COMMAND
  ```

  You should use the UPDATE command instead of INSERT in this case.

- For segments in which the parent segment contains ACCESS=PE, and the child segment contains ACCESS=MU without the NU option in the ADABAS FDT:

  - If an INSERT command is issued for the parent (PE) segment alone, Adabas automatically inserts an empty child (MU) segment. You can use the UPDATE command to provide values for this child.

  - If one INSERT command is issued for the parent (PE) and child (MU) segments simultaneously, the first occurrence in the child segment is inserted by the Write Adapter for Adabas using the values from the INSERT command.

- For segments in which the parent segment has ACCESS=PE, and the child segment has ACCESS=MU with the NU option in the ADABAS FDT, the empty child is automatically suppressed by Adabas. You can use an INSERT command for the parent and child segments separately or simultaneously.

Reference: Effect of UNQKEYNAME on INSERT Actions

The UNQKEYNAME attribute in the Access File determines how the adapter presents an INSERT command to Adabas. The UQ option in the ADABAS FDT and the specific field values listed in the INSERT command determine whether Adabas actually inserts the segment instance. The following table describes how these factors affect the result of the INSERT command. Assume that the Access File specifies UNQKEYNAME=EMPLOYEE_ID and that the employee ID value EMPID005 already exists in the data source:

Result of the INSERT Command for Existing EMPLOYEE_ID EDMPID005

EMPLOYEE_ID has the UQ Option in FDT	Fields in INSERT Command	Instance Inserted
No	EMPLOYEE_ID only.	No - rejected duplicate
Yes	EMPLOYEE_ID only.	No - rejected duplicate
No	EMPLOYEE_ID plus fields with values that do not already exist.	Yes
Yes	EMPLOYEE_ID plus fields with values that do not already exist.	No - message (FOC4561), RC=198

Result of INSERT Command when EMPLOYEE_ID is not in the field list

UNQKEYNAME = EMPLOYEE_ID	Instance Inserted
Yes	No - message (FOC4563)
No	Yes, with empty EMPLOYEE_ID value.

Syntax: How to Update Field Values in an Adabas Data Source

```
UPDATE mfname SET field1 ='value1' [,field2 ='value2'...]
WHERE kfield1 ='kvalue1' [AND kfield2 ='kvalue2'...]
```

where:

mfname

Is the name of the Master File to use.

field1, field2, ...

Are the names of the fields to be updated.

`'value1', 'value2', ...`

Are the new values for the updated fields, enclosed in single quotation marks.

`kfield1, kfield2, ...`

Are, at a minimum, all of the key fields for all the segments in the path to the target. Only one target segment is updated. You can include additional fieldname=value pairs for the target or any segment in the path. When additional fieldname=value pairs are present for the target segment, these are used for change verification protocol processing. If all target fieldname=value pairs that are present match the Adabas segment field values, the segment is updated with the SET fieldname values. The field list can contain only elementary fields. Groups, subdescriptors, superdescriptors, and hyperdescriptors, cannot be used in the field list.

`'kvalue1', 'kvalue2', ...`

Are the values that identify the target segment, enclosed in single quotation marks.

Reference: Rules for Updating Records in an Adabas Data Source

- For a segment with a unique key, the key field value and any additional fieldname=value pairs are used to qualify the target segment for an update.
- For a segment with a non-unique key, you must supply the key field. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is updated. The use of Adabas descriptors for this type of segment is highly recommended for efficiency.
- For a segment with no key, you must supply at least one fieldname=value pair. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is updated.

Syntax: How to Delete Records From an Adabas Data Source

```
DELETE FROM mfname WHERE field1 ='value1' [AND field2 ='value2'...]
```

where:

`mfname`

Is the name of the Master File to use.

`field1, field2, ...`

Are, at a minimum, the names of all of the key fields for all the segments in the path to the target. Only one target segment is deleted; Adabas cascades the delete to dependent segments. Additional fieldname=value pairs may be included for the target segment and any segment in the path that has no key or a non-unique key. It is recommended to use Adabas descriptor fields for any additional fieldname=value pairs.

`'value1', 'value2', ...`

Are the values that identify the target segment to be deleted.

Reference: Rules for Deleting Records From an Adabas Data Source

- For a segment with a unique key, the key field value is used to delete the target segment.
- For a segment with a non-unique key, you must supply the key field. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is deleted. The use of Adabas descriptors for this type of segment is highly recommended for efficiency.
- For a segment with no key, you must supply at least one fieldname=value pair. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is deleted.
- A segment occurrence with ACCESS=PE or MU is deleted from the data source except if it is the last occurrence for an ACCESS=PE segment. Adabas only deletes the last occurrence if all fields have the NU option in the FDT; if they do not all have this option, the occurrence has empty values in all fields.
- When you delete segments that have dependent segments, the DELETED counter for the session may have an incorrect value. For a first level segment with descendants, this counter is always incorrect. For a second level segment, this counter is incorrect if there are multiple descendant segments. For a third level segment, this counter is always correct.

Adapter Navigation

To retrieve the necessary records that fulfill a request, the Adapter for Adabas navigates the Adabas database through direct calls in read-only mode. The calls generated depend on three factors:

- Information supplied in the Master File.
- Information supplied in the Access File for a specified segment.
- Particulars of the report request.

The adapter uses information from these sources to choose the most appropriate and efficient retrieval method.

There are three logical types of Adabas access:

- Entry segment retrieval of Adabas records.
- Retrieval of descendant periodic groups and multi-value fields.
- Retrieval of descendant Adabas records.

Entry Segment Retrieval of Adabas Records

In this section:

The server constructs an Adabas request based on the Master File, Access File, and the report request. The root of the subtree is the entry segment into the database for a particular request.

For Adabas structures, the root of the subtree must be a segment in the Access File containing singly occurring fields (ACCESS=ADBS). The Adabas calls generated to retrieve data for this entry segment depend on the Access File information that has been supplied for the segment and the selection tests in the request.

The first decision the Adapter for Adabas makes for the entry segment is whether to retrieve all the records of the segment or just a subset.

All records are retrieved for a segment if either of the following conditions exists:

- There is *no* selection test on any field that is a descriptor (defined in the Access File with TYPE=SPR, DSC, or NOP) and the SEQFIELD has not been defined.
- The specified selection test is *not* IF or WHERE, and does not use the relational operators:

  ```
  IS, EQ, GE, GT, LT, LE, or FROM...TO
  ```

  For example:

  ```
  field IS value1 [(OR value2... OR valuen)]
  ```

  ```
  field IS (ddname)
  ```

  ```
  field FROM value1 TO value2 [(OR value3 TO value4...)]
  ```

  ```
  field GE value1
  ```

```
field GT value1
field LT value1
field LE value1
group EQ 'value1/value2/value3'
```

When you are using one or more of these selection tests against a descriptor, the adapter uses the inverted list to retrieve a subset of the records for the segment.

For example, consider the following selection criterion:

```
field IS value1 [(OR value2... OR valuen)]
```

In this example, any combination of full values or partial values triggers the use of inverted lists except when the $ (used to indicate a masked field) is in the first position of the value. If the $ is in the first position of the value, the search conducted by Adabas is not done through the descriptor's inverted list. Instead, the search is conducted physically through the database in the same manner that a Read Physical call performs a retrieval. See *Read Physical Calls* on page 2-108 for more information.

Once the adapter determines whether to retrieve all or a subset of the segment records, it decides which access strategy to use based on the Access File attribute settings. The following are the strategies that the adapter can use:

- Retrieval of all records through Read Physical (L2) calls discussed in *Read Physical Calls* on page 2-108.
- Retrieval of all records through Read Logical (L3) calls without using a starting value discussed in *Read Logical Navigation* on page 2-108.
- Retrieval of a subset of records through Read Logical (L3) calls using a starting value discussed in *Read Logical Navigation* on page 2-108.
- Retrieval of all records for the entry segment through Read Logical by ISN (L1) calls discussed in *Read Logical Navigation* on page 2-108.
- Retrieval of a subset of records through simple FIND (S1) calls discussed in *FIND Navigation* on page 2-113.
- Retrieval of a subset of records through complex FIND (S1) calls discussed in *FIND Navigation* on page 2-113.
- Retrieval of all records through Read Descriptor Value (L9) direct calls discussed in *Read Descriptor Value (L9) Direct Calls* on page 2-116.

The selection logic is designed to implement as many of the record selection tests as possible at the Adabas level. It minimizes the number of I/O operations required to access the necessary data by issuing efficient calls to Adabas.

Read Physical Calls

Read Physical calls read every database record in a physical file, and then return every record for the entry segment to the server. The server must then select the records that meet the request's selection criteria and discard the rest.

The Adapter for Adabas issues Read Physical (L2) calls to retrieve all records for the entry segment when:

- The request contains no optimizable selection tests on an inverted list (descriptor) and the Access File does not have a SEQFIELD defined.
- With CALLTYPE=FIND, there is no screening on a descriptor, the screening for a non-descriptor has been set off, and the Access File does not have a SEQFIELD defined.

See *Adabas Reporting Considerations* on page 2-87 for more information about retrieval of records with no screening on a descriptor.

The steps the adapter and Adabas perform to complete a Read Physical call are:

1. The adapter constructs a Read Physical (L2) call.
2. Adabas returns each record in the physical file corresponding to the Adabas file number specified. The records are retrieved in the order in which they are physically stored.

Subsequent retrieval for the entry segment is completed by issuing the same Read Physical call with the User Control Block unmodified. The adapter terminates retrieval when one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached (see *Adabas Reporting Considerations* on page 2-87 for information about these keywords).

Read Physical (L2) calls can be faster than Read Logical (L3) calls depending on the request and the data dispersion of the physical storage. Ask your Adabas database administrator if you should use SEQFIELD for a given Adabas record, or if Read Physical would be more efficient.

Read Logical Navigation

When designing a database, it is necessary to know the contents of the files being created. Defining descriptors is very important for efficiency. The Adapter for Adabas does not have information about the different fields, or about which field is more efficient to parse first, unless the Access File has this information. The Access File tells the adapter how to access your data in the most efficient manner. The order of the fields (for example, DSC, NOP, or SPR descriptors) is important in the Access File, as the adapter uses this order when executing an IF or WHERE statement.

The Access File needs to have information about the index fields used in the report request. The order in which you define these fields is important. Adabas is field oriented, not positional. The order of the fields will not affect retrieval, except for the selection of which inverted list to use.

The Access File controls the order of the IF or WHERE statements, so the database administrator can set up the Access File in the order of the most efficient reads. The user will then receive the requested data much more efficiently. The order in the Access File controls the order of retrieval.

Read Logical Calls Without a Starting Value

The Adapter for Adabas uses Read Logical (L3) calls to retrieve all records for the entry segment when the Access File contains a SEQFIELD value for that segment and the report request does not contain an optimizable selection test on an inverted list, and CALLTYPE=RL.

SEQFIELD is generally used to suppress Read Physical calls when there is no optimizable selection test on an inverted list. A Read Physical call will be issued if a SEQFIELD is not defined.

The steps the adapter and Adabas perform to complete a Read Logical call without starting values are:

1. The adapter constructs a Read Logical (L3) call without the 'Value Start' option and without the Adabas Search and Value buffers.
2. Adabas returns each record corresponding to an entry in the inverted list. The records are in the same order as the inverted list.

Subsequent retrieval for the entry segment is done by issuing the same Read Logical call with the User Control Block unmodified. The records are returned in ascending value of the inverted list. The adapter terminates retrieval when one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

In an Adabas file that has no record types, you can select the value of SEQFIELD from any inverted list containing entries for all records on the file.

SEQFIELD is required when the Adabas file has several record types. In this case, the value of SEQFIELD is an inverted list which has entries for a single record type.

If the descriptor used as the SEQFIELD is null-suppressed or contains null-suppressed fields, only records where the descriptor is populated will be included in the retrieval. An entry is not included in the inverted list if a field is null and will not be returned by Adabas to the server. See your Software AG documentation for more information about null-suppression and how it affects data retrieval.

Read Logical Calls With a Starting Value

The Adapter for Adabas constructs Read Logical (L3) calls for the root of the accessed subtree when the following conditions exist:

- The Access File contains CALLTYPE=RL for that segment.
- The request contains an optimizable selection test against a descriptor or superdescriptor identified by TYPE=SPR or DSC in the Access File.
- None of the selection tests is on a field identified by TYPE=NOP in the Access File.

When a report request contains multiple optimizable selection tests, the order of descriptors in the Access File determines the order in which the server applies the selection tests. The server issues a Read Logical (RL) call using the first descriptor listed in the Access File that participates in a selection test.

Therefore, for efficient processing, you should describe the most restrictive descriptor at the beginning of its segment in the Access File. The order of descriptors in the Master File has no effect on selection processing.

The steps Adabas performs to complete a Read Logical call are:

1. The adapter constructs a Read Logical (L3) call with the starting value to be retrieved for that inverted list.
2. The adapter calls Adabas with a 'V' in Command Option 2, which instructs Adabas to use the 'Value Start' option.
3. The call retrieves the first record whose value in the inverted list is equal to or greater than the value specified in the request.

Subsequent retrieval for the entry segment is done by issuing the same Read Logical call with the User Control Block unmodified. The records are returned in ascending value of the inverted list.

The adapter terminates retrieval when one of the following occurs:

- The value of the inverted list is out of range of the request.
- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

If there are multiple values specified in the request (for example, WHERE *field* EQ *value* OR *value*..., WHERE *field* FROM *val1* TO *val2* OR *val3* TO *val4*) and RECORDLIMIT or READLIMIT has not been reached, the Adapter for Adabas releases the Command ID and reissues the Read Logical call with the next starting value. For each set of values, the adapter terminates retrieval when one of the above occurs.

Retrieval Through Read Logical by ISN (L1) Calls

The Adapter for Adabas uses Read Logical by ISN (L1) calls to retrieve all records for the entry segment when the:

- Report request does not contain an optimizable selection test on an inverted list or an ISN list.
- Access File contains a SEQFIELD value for that segment and this value is equal to the ISN field name.

Adabas returns each record corresponding to an entry in the Address Converter. The records are in ascending value of the ISN. For example:

Access File ADATEST

```
SEGNAM=S01, ACCESS=ADBS, ... , SEQFIELD=ISN_FIELD ,$
```

Request

```
SELECT AA_FIELD, AJ_FIELD FROM ADATEST;
```

The Adabas Interface uses Read Logical by ISN (L1) calls to retrieve a single record for the entry segment when the:

- Report request contains the only EQ relational operator in the selection test on an ISN list.
- ISN field is used as the cross-referenced field in the Join to Adabas file.

Adabas returns RC 113 if the record with the ISN defined in the test is not present in the Address Converter for the file. Then the Adabas Interface returns the answer "Record is not found".

For example:

```
SELECT AA_FIELD, AJ_FIELD FROM ADATEST
WHERE ISN_FIELD = 1100;
```

Note: The Multifetch option will be suppressed for this call.

The Adabas Interface uses Read Logical by ISN (L1) calls to retrieve subset of records for the entry segment when the:

- Report request contains the only GE/GT relational operator in the selection test on an ISN list.
- Report request does not contain any selection test on an inverted list.
- Access File contains a SEQFIELD value for that segment.

Adabas returns each record corresponding to an entry in the Address Converter. The records are in ascending value of the ISN.

For example, the Access File ADATEST contains:

```
SEGNAM=S01, ACCESS=ADBS , ... ,SEQFIELD=AA_FIELD ,$
```

Request 1. When Read Logical by ISN (L1) used:

```
SELECT AA_FIELD, AE_FIELD FROM ADATEST
WHERE ISN_FIELD > 1100;
```

Request 2. When Read Logical (L3) used:

```
SELECT AA_FIELD, AE_FIELD FROM ADATEST
WHERE ISN_FIELD > 1100 AND AJ_FIELD = 'TAMPA';
```

After issuing an Insert request to the file with the ISN field in the Master File, the resulting ISN from Adabas is printed in the message FOC4592:

```
(FOC4592) RECORD IS INSERTED WITH ISN : nnnnn
```

For example:

```
SQL
INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD)
VALUES ('11111111', 'TAMPA');
END
```

```
ADBTEST ADBSINX ON 09/20/2002 AT 15.22.16
(FOC4592) RECORD IS INSERTED WITH ISN : 11
(FOC4566) RECORDS AFFECTED DURING CURRENT REQUEST : 1/INSERT
TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 1 UPDATED = 0 DELETED = 0
```

If the Insert request contains an ISN field in the fields list and a corresponding value is not 0, then the adapter issues an N2 call to Adabas. Adabas assigns this ISN value to the record provided by the user. Adabas returns RC 113 if this value was already assigned to another record in the file or if it is larger than the MAXISN in effect for the file.

For example:

Request 1

```
SQL
INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD, ISN_FIELD)
VALUES ('11111114', 'TAMPA', 14);
END
ADBTEST ADBSINX ON 09/20/2002 AT 15.22.16
(FOC4592) RECORD IS INSERTED WITH ISN : 14
(FOC4566) RECORDS AFFECTED DURING CURRENT REQUEST : 1/INSERT
TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 1 UPDATED = 0 DELETED = 0
```

Request 2

```
SQL
INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD, ISN_FIELD)
VALUES ('11111114', 'TAMPA', 999999);
END
ADBEMP43ADBSIN ON 09/19/2002 AT 16.16.22
(FOC4561) ERROR IN ADABAS INCLUDE /113
TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 0 UPDATED = 0 DELETED = 0
```

FIND Navigation

The manipulation of ISN lists is done on the initial call to Adabas. Both the intermediate lists and the resulting list may be very large and may take up a large percentage of the Adabas workspace. The Adabas work area is shared by all Adabas users, and complex FINDs may affect programs that are updating the database. You may want to suppress FINDs altogether. Accomplish this suppression by specifying CALLTYPE=RL on every Access File segment. Ask your Adabas database administrator whether to use FIND processing.

Simple FIND Calls

The Adapter for Adabas constructs a simple FIND (S1) call, using a single inverted list structure, for the root of the accessed subtree if one of the following conditions is met:

- The Access File contains either CALLTYPE=FIND for that segment or does not specify the CALLTYPE attribute, in which case the adapter default is CALLTYPE=FIND. Additionally, the request contains a single selection test on a descriptor identified with TYPE=DSC or a superdescriptor identified with TYPE=SPR in the Access File.
- For any value of CALLTYPE (RL or FIND), if the request contains a single selection test on a subdescriptor or superdescriptor containing partial fields (identified with TYPE=NOP in the Access File).

CALLTYPE=FIND instructs the adapter to generate FIND calls to retrieve records from the inverted lists. This method is the default for processing selection criteria on inverted lists for a segment (if CALLTYPE is omitted from the Access File segment declaration).

If you have selected a field defined with TYPE=NOP (for example, subdescriptors), a FIND call is issued even if RL has been specified in cases when:

- SET NOPACCESS FIND was performed.
- The server is running in an OpenVMS environment.
- The same Adabas field is defined in a descendant PE/MU segment.

Switching from RL to FIND mode is also performed if a:

- Field is defined with TYPE=SPR and the server is running in an OpenVMS environment.
- Field is defined with TYPE=PDS (phonetic descriptor) or TYPE=HDS (hyperdescriptor).
- Field is defined in a PE segment (that it is a part of periodic group).

For performance considerations, a FIND (S1) call does not issue a READ ISN (L1) call if an answer set containing only one record is returned. The GET FIRST option returns the first record in the inverted list. This reduces unnecessary I/O calls. If the FIND call returns more than one ISN in a list, the GET NEXT option of L1 is used to start reading the ISN list from the second entry.

The steps Adabas performs to complete a FIND (S1) call are:

1. The adapter constructs a FIND (S1) call with the value(s) specified.
2. The adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of Internal Sequence Numbers (ISNs) in the Adabas work area.
3. Adabas constructs a complete list of ISNs for every record matching the selection criteria in the work area of Adabas. This list is sorted in ascending ISN order.
4. The adapter issues a Read ISN (L1) call to retrieve the first record from the Adabas work area which matches the selection criteria. The records are returned in ascending ISN order.

All subsequent retrieval for this entry segment is done using the Read ISN (L1) call issued against the ISN list held by Adabas. L1 commands are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Complex FIND Calls

The Adapter for Adabas constructs complex FIND (S1) calls, using two or more inverted list structures, for the root of the accessed subtree if one of the following conditions is met:

- The Access File contains CALLTYPE=FIND for that segment (or omits CALLTYPE). The request contains multiple selection tests on a descriptor or superdescriptor described with TYPE=DSC or TYPE=SPR in the Access File.
- For any value of CALLTYPE, the request contains selection tests on a subdescriptor or superdescriptor containing partial fields (identified with TYPE=NOP in the Access File).

The steps Adabas performs to complete a FIND call on multiple inverted list structures are:

1. The adapter constructs a FIND (S1) call with all the values specified for each inverted list on which there are selection criteria.
2. The adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of ISNs in the Adabas work area.
3. Adabas constructs an ISN list for each inverted list specified.
4. Adabas merges these lists into a final list of ISNs which match all of the selection criteria in the call. This list is kept in the Adabas work area and is sorted in ascending ISN order.
5. The adapter then issues a Read ISN (L1) call to retrieve the first record from the list in the Adabas work area. The records are returned in ascending ISN order.

All subsequent retrieval for this entry segment is completed using Read ISN (L1) calls issued against the ISN list held by Adabas. L1 commands are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Read Descriptor Value (L9) Direct Calls

The Adapter for Adabas issues the Adabas Read Descriptor Value (L9) direct call when you use the COUNT command or SUM CNT.*field* within a report request. L9 calls are, by definition, limited to descriptors defined to the server with the following:

- INDEX=I in the Master File.
- A field type of NOP, DSC, or SPR (defined in the Access File).

L9 calls allow an aggregate ISN count to be returned for each unique value on the inverted list at a cost of only one call per unique value. This technique allows a dramatic efficiency gain over passing each record to the server to process the count.

Selection criteria (for example, WHERE *descriptor* EQ '1234') are passed along with the L9 call to further limit the number of calls. If any non-descriptor fields are mentioned in the TABLE request, a Read Descriptor Value (L9) direct call is not possible and the server uses its own internal count processing. If a BY field is used, it must be the same field or GROUP name used as the object of the COUNT verb.

Here is an example of the SUM CNT.*field* using EMPFILE1:

```
TABLE FILE EMPFILE1
SUM CNT.DEPT_S03
BY DEPT_S03
END
```

Descendant Periodic Groups and Multi-value Fields

The Adapter for Adabas must retrieve a count of the number of occurrences before attempting to retrieve a periodic group (PE) or multi-value (MU) field. This retrieval is accomplished by taking the appropriate Adabas counter field (indicated by the OCCURS attribute on the segment declaration in the Master File) and placing its ALIAS (the two-character Adabas name of the PE or MU appended with the letter C) in the Format buffer of the call for the parent record fields.

Adabas returns to this counter field the total number of occurrences that exist for the PE or MU.

The adapter retrieves the descendant data in one of the following ways:

- For MU fields or PE groups which do not contain an MU, all the occurrences are retrieved at once. The counter field ALIAS is used without the letter C, to retrieve the entire set of occurrences in one Read ISN (L2) call.
- For PE groups which contain one or more MU fields, the adapter retrieves a single occurrence of all component fields at a time, including the count(s) of the number of MU descendants, if required. For example, a PE group containing an MU field which has five occurrences of the PE requires five Read ISN (L2) calls to retrieve all of them.

Descendant Adabas Records

In this section:

The access strategy used to retrieve descendant records in a subtree depends on the Access File attributes specified for a given segment and the SEGTYPE attribute in the Master File (or the SEGTYPE created on the cross-referenced segment as the result of a JOIN). In some cases, the selection criteria in a request further affect access strategy for descendant segments.

Descendant records are related to their parents by a field or GROUP in the parent that corresponds to a field or GROUP in the descendant. This relationship is set up either in the Access File using the KEYFLD/IXFLD pair or through the host field/cross-referenced field combination in a JOIN.

The most appropriate access strategy is selected based on the value of CALLTYPE of the descendent segment in the Access File. The three basic strategies used to obtain descendant records are:

- Retrieval of descendant records through Read Logical calls using a starting value.
- Retrieval of descendant records through simple FIND calls.
- Retrieval of descendant records through complex FIND calls.

The CALLTYPE attribute, the KEYFLD/IXFLD attributes, the SEGTYPE, and the TABLE request are the determining factors in the way descendant data is retrieved.

Read Logical Calls Using a Starting Value

The Adapter for Adabas constructs Read Logical (L3) calls for related descendant records of a parent when the following conditions are met:

- The Access File contains CALLTYPE=RL for that segment.
- The IXFLD or JOIN *to* field is described as TYPE=DSC or TYPE=SPR in the Access File.
- No other selection criteria on inverted lists for this segment have selection on a descriptor with TYPE=NOP.

Read Logical processing is performed on a single inverted list only. The steps the adapter performs to complete a Read Logical call using a starting value are:

1. The adapter constructs a Read Logical (L3) call with the value of the KEYFLD from the parent segment for the IXFLD inverted list. This call retrieves the first record whose value in the inverted list is equal to or greater than the value of the KEYFLD.
2. For unique descendants, no additional retrieval is performed on the descendant segment for any given parent. (For an embedded cross-referenced segment, the SEGTYPE is U or KU or the segment was the cross-referenced segment in a unique JOIN.)
3. For non-unique descendants, subsequent retrieval for the entry segment is completed by issuing the same Read Logical call with the User Control Block unmodified. The records are returned in ascending value of the inverted list.

The adapter terminates retrieval when one of the following conditions is met:

- The value of the inverted list is greater than the value of the KEYFLD.
- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Simple FIND Calls (Descendant Records)

The Adapter for Adabas constructs a simple FIND (S1) call, using a single inverted list, to obtain related descendant records for a parent when one of the following conditions exists:

- The Access File contains CALLTYPE=FIND (or omits CALLTYPE) and the request contains no selection criteria on the descendant segment.
- The Access File contains CALLTYPE=FIND (or omits CALLTYPE) and the request contains selection criteria on the descendant segment and the descendant is unique. (For an embedded cross-referenced segment, the SEGTYPE is U or KU or the segment was the cross-referenced segment in a unique JOIN.)
- For any value of CALLTYPE, IXFLD is described with TYPE=NOP in the Access File and the request contains no selection criteria on the descendant segment.

The steps Adabas performs to complete simple FIND calls against descendant segments are:

1. The adapter constructs a FIND (S1) call with the value of the KEYFLD from the parent segment for the inverted IXFLD list.
2. For unique descendants, Adabas returns the record on the FIND call; no 'H' command is issued to Adabas. Only one call will be issued, and only the first instance of data will be returned.
3. For non-unique descendants, the adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of ISNs in the Adabas work area.
4. Adabas constructs a complete list of ISNs for every record related to the parent record in the work area. This list is sorted in ascending ISN order.
5. The adapter issues a Read ISN (L1) call to retrieve the first record from the Adabas work area which matches the selection criteria.

If the descendant is not unique, subsequent retrieval of records for this descendant segment is completed using the Read ISN (L1) call issued against the ISN list held by Adabas. L1 calls are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Complex FIND Calls (Descendant Records)

The Adapter for Adabas constructs a complex FIND (S1) call, using several inverted list structures, to obtain related descendant records for a parent when one of the following conditions is met:

- The Access File contains CALLTYPE=FIND for that segment (or omits CALLTYPE) and the descendant is non-unique. (For an embedded cross-referenced segment, the SEGTYPE is S or KM, or the segment was the cross-referenced segment in a non-unique JOIN.) The report request contains one or more selection criteria on the descendant segment.
- The Access File contains any value of CALLTYPE. The descendant is non-unique. (For an embedded cross-referenced segment, the SEGTYPE is S or KM, or the segment was the cross-referenced segment in a non-unique JOIN.) The request contains one or more selection criteria on the descendant segment, one of which is on a descriptor, or the IXFLD is described as TYPE=NOP in the Access File.

The steps Adabas performs to complete a complex FIND call are:

1. The adapter constructs a FIND (S1) call with the value of the KEYFLD from the parent segment using the inverted IXFLD list, in addition to all other selection criteria on inverted lists from the request.

2. For non-unique descendants, the adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of ISNs in the Adabas work area.

3. Adabas constructs an ISN list for each inverted list specified, and merges these lists into a final list of ISNs that match all of the selection criteria in the call. This list is kept in the Adabas work area and is sorted in ascending ISN order.

4. The adapter then issues a Read ISN (L1) call to retrieve the first record from the Adabas work area which matches the selection criteria.

Subsequent retrieval of records for this descendant segment is done through the Read ISN (L1) call issued against the ISN list held by Adabas. L1 commands are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

CHAPTER 3

Using the Adapter for Adabas Stored Procedures

Topics:

The Adapter for Adabas Stored Procedures enables you to invoke an Adabas stored procedure, which is a Natural subprogram that has been created and tested using Natural facilities.

Note: This adapter is only available on the z/OS and OS/390 operating systems.

You invoke a stored procedure by:

- Issuing a data retrieval command in an SQL request
- Issuing a Data Manipulation Language (DML) request. (DML is iWay's internal retrieval language, also known as TABLE.)
- Executing a remote procedure call (RPC).

This enables you to invoke an Adabas stored procedure from a variety of environments, including WebFOCUS and DataMigrator.

You can pass input parameter values to a stored procedure by specifying filtering criteria in the request or to an RPC by by passing a parameter list. You retrieve output as an answer set.

You use the Web Console to generate a synonym for a stored procedure using information from Natural data areas; the synonym maps the procedure's input and output parameters.

Preparing the Adabas Stored Procedures Environment

The Adapter for Adabas must be installed and configured prior to configuring the Adapter for Adabas Stored Procedures. For information about configuring the Adapter for Adabas, see Chapter 2, *Using the Adapter for Adabas*.

Configuring the Adapter for Adabas Stored Procedures

You configure the Adapter for Adabas Stored Procedures from the Web Console.

Procedure: How to Configure the Adapter From the Web Console

Note: You must have configured the Adapter for Adabas in order to be able to configure the Adapter for Adabas Stored Procedures. For configuration instructions, see Chapter 2, *Using the Adapter for Adabas*.

To add the Adapter for Adabas Stored Procedures to the configuration:

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.
2. Expand the *Add* folder, expand the *Procedures* group folder, then expand the *Adabas/NAT* adapter folder and click a connection. The Add Adabas/NAT to Configuration pane opens.
3. No input is required. Simply click the *Configure* button.

   Adabas/NAT is added to the configuration.

Managing Adabas Stored Procedure Metadata

When the adapter invokes an Adabas stored procedure, it needs to know where to find the stored procedure and how to process its parameters. For each stored procedure the adapter will access, you create a synonym that describes these things.

Reference: Requirements for Creating a Synonym

Creating a synonym—that is, metadata—for Adabas Stored Procedures consists of mapping input and output parameters that are described in a Natural library as local or global (LDA/GDA) data areas.

Only a definition's first record can be level 1. If a data area has more than one parameter, the parameters must be defined as a structure.

Example: Defining One Parameter in an Adabas Stored Procedure

```
Local     SAMPPRM1 Library SYSRPC                 DBID     3 FNR     8
Command                                                             > +
I T L Name                           F Leng Index/Init/EM/Name/Comment
All - ------------------------------ - ---- -------------------------
    1 PARAMETER                      A   10

-------------------------------------------------------- S 1    L 1
```

Example: Defining Multiple Parameters in an Adabas Stored Procedure

```
Local     SAMPPRM2 Library SYSRPC                 DBID     3 FNR     8
Command                                                             > +
I T L Name                           F Leng Index/Init/EM/Name/Comment
All - ------------------------------ - ---- -------------------------
    1 PARAMETERS
    2 TEXTINP                        A   60
    2 TEXTOUT                        A   60
    2 KEYWORD                        A   20 (1:20)

-------------------------------------------------------- S 4    L 1
```

Creating Synonyms

How to:

Create a Synonym From the Web Console

Example:

Generating a Synonym

Reference:

Managing Synonyms

CHECKNAMES for Special Characters and Reserved Words

Master File Attributes

Access File Attributes

A synonym defines a unique logical name (also known as an alias) for each Adabas stored procedure, and one set of input/output parameters. The adapter requires that you generate a synonym for each Adabas stored procedure that you want to invoke with the adapter.

Synonyms are also useful because they insulate client applications from changes to a procedure's location. You can move or rename a procedure without modifying the client applications that use it; you need make only one change, redefining the procedure's synonym on the server.

Creating a synonym generates a Master File and an Access File: these are metadata that describe to the server the Adabas stored procedure's name, parameters, and options.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata pane opens.

   To create a synonym, you must have configured the adapter. See *Configuring the Adapter for Adabas Stored Procedures* on page 3-2 for more information.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The Select List of Natural Subprograms pane (Step 1 of 5) opens.

4. Enter the following parameters to identify the stored procedure you want to invoke:

Parameter	**Description**
Data Base ID	The database ID.
Natural System System File	The Natural System system file.
Natural Stored Procedure Library	The stored procedure library. Natural library with subprograms, which are candidates to a stored procedure.
Mask for Subprogram Name	The mask for returning a list of subprograms from which to chose. Enter a string for filtering subprogram names, inserting the wildcard character (%) at the beginning and/or end of the string. For example, enter ABC% to display subprograms whose names begin with the letters ABC; %ABC to display subprograms whose names end with the letters ABC; %ABC% to display subprograms whose names contain the letters ABC at the beginning, middle, or end; or % to display all subprograms.

5. Click *Select Subprogram*.

The List of Stored Procedure Candidates pane (Step 2 of 5) opens.

6. Select the name of the stored procedure you want to invoke from the drop-down list box, and click *Next*.

The List of Natural Libraries pane (Step 3 of 5) opens.

7. Enter the following parameters to identify the stored procedure's request buffer:

Parameter	**Description**
Data Base ID	The database ID.
Natural System User File	The Natural System user file.
Mask for the LIBRARIES	The mask for returning a list of libraries from which to chose the one containing local/global data areas that describe the stored procedure's parameters.

8. Click *Select Libraries*.

   The NATURAL Library pane (Step 4 of 5) opens.

9. Enter the mask for returning a list of the data areas from which you will chose the data area defining the stored procedure's parameters.

10. Select the LDA/GDA that defines the stored procedure's parameters from the drop-down list box, and click *Next*.

    The Input/Output Area (Step 5 of 5) pane opens.

11. Enter the following parameters to specify the stored procedure's parameters:

Parameter	**Description**
Synonym name	The name of the synonym. You can retain the current name or change it. Changing a synonym's name enables you to manage multiple synonym versions to reflect, for example, multiple environments, or synonyms with different application logic such as different sets of Master File DEFINE attributes.
Prefix	An optional prefix for the synonym. Providing a prefix enables you to manage multiple synonym versions to reflect multiple environments, or synonyms with different application logic such as different sets of Master File DEFINE attributes.
Suffix	An optional suffix for the synonym. Providing a suffix enables you to manage multiple synonym versions to reflect multiple environments, or synonyms with different application logic such as different sets of Master File DEFINE attributes.
Application Directory	The application directory in which the synonym will be created. Chose a directory from the drop-down list. The list of application directories was specified when your iWay server was configured.
Overwrite existing synonyms	Specifies that this synonym should overwrite any earlier synonym with the same fully-qualified name.
Trigger File Data Base ID	The trigger file's database ID.

Parameter	**Description**
Trigger File Number	The trigger file number.
Transaction Option	Specifies the stored procedure's transaction property.
Parameter Option	Specifies the type of parameters the stored procedure has: no parameters, control, or error message.
Input / Output	Specifies which data area's are for input parameters and which are for output parameters. You can specify the same data area as the source of both the input and output parameters. You can specify a data area only if Parameter Option is set to a value other than None.

12. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

    When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

13. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

14. Click *Create Synonym*.

    The synonym for the stored procedure is now created and added under the specified application directory. A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

15. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '| ', '&', ' (', ') ', ' <', '> ', '' ', ' =', ''''

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Example: Generating a Synonym

The following sample Master File and Access File are generated from the Web Console using the CREATE SYNONYM command.

Generated Master File SAMP_LEM.MAS

```
FILENAME=SAMP_LEM, SUFFIX=ADANAT  , $
  SEGMENT=SEG1, SEGTYPE=S0, $
$  GROUP=REC_BUFFER, USAGE=A49, ACTUAL=A28, $
    FIELDNAME=#REF_NAME, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#REF_COUNT, USAGE=P4, ACTUAL=Z3, $
    FIELDNAME=#REF_FROM, USAGE=P16, ACTUAL=P8, $
    FIELDNAME=#REF_TO, USAGE=P16, ACTUAL=P8, $
    FIELDNAME=#REF_RC, USAGE=A1, ACTUAL=A1, $
  SEGMENT=SEG2, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_NAME, $
   GROUP=#REF_NAME, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#REF_NAME_FIRST_2, USAGE=A2, ACTUAL=A2, $
    FIELDNAME=#REF_NAME_LAST_6, USAGE=A6, ACTUAL=A6, $
  SEGMENT=SEG3, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_FROM, $
   GROUP=#REF_FROM, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#FCHAR, USAGE=A8, ACTUAL=A8, $
  SEGMENT=SEG4, SEGTYPE=S0, PARENT=SEG3, OCCURS=8, POSITION=#FCHAR, $
    FIELDNAME=#FCHAR, USAGE=I9, ACTUAL=I1, $
  SEGMENT=SEG5, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_FROM, $
   GROUP=#REF_FROM, USAGE=A5, ACTUAL=A5, $
    FIELDNAME=#RESET_PARM, USAGE=A5, ACTUAL=A5, $
  SEGMENT=SEG6, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_FROM, $
   GROUP=#REF_FROM, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#REF_FROM_A, USAGE=A8, ACTUAL=A8, $
  SEGMENT=SEG7, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#TCHAR, USAGE=A8, ACTUAL=A8, $
  SEGMENT=SEG8, SEGTYPE=S0, PARENT=SEG7, OCCURS=8, POSITION=#TCHAR, $
    FIELDNAME=#TCHAR, USAGE=I9, ACTUAL=I1, $
  SEGMENT=SEG9, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#REF_TO_A, USAGE=A8, ACTUAL=A8, $
  SEGMENT=SEG10, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A1, $
    FIELDNAME=#REF_TO_P1, USAGE=P2, ACTUAL=P1, $
  SEGMENT=SEG11, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A2, $
    FIELDNAME=#REF_TO_P3, USAGE=P4, ACTUAL=P2, $
  SEGMENT=SEG12, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A3, $
    FIELDNAME=#REF_TO_P5, USAGE=P6, ACTUAL=P3, $
  SEGMENT=SEG13, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A4, $
```

```
    FIELDNAME=#REF_TO_P7, USAGE=P8, ACTUAL=P4, $
  SEGMENT=SEG14, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A5, $
    FIELDNAME=#REF_TO_P9, USAGE=P10, ACTUAL=P5, $
  SEGMENT=SEG15, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A6, $
    FIELDNAME=#REF_TO_P11, USAGE=P12, ACTUAL=P6, $
  SEGMENT=SEG16, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A7, $
    FIELDNAME=#REF_TO_P13, USAGE=P14, ACTUAL=P7, $
  SEGMENT=SEG17, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A16, ACTUAL=A8, $
    FIELDNAME=#REF_TO_P15, USAGE=P16, ACTUAL=P8, $
```

Generated Access File SAMP_LEM.ACX

```
SEGNAME=SEG1, STPNAME=SAMPP001, OPTTRN=N, OPTPRM=C, OPTUPD=U,
  TRGDBID=3, TRGFILE=5, $
```

Reference: Master File Attributes

The following Master File attributes describe Adabas data segments.

Attribute	Description
`FILE`	The Master File name. This name may or may not match the stored procedure name.
`SUFFIX`	Identifies the adapter, and is always ADANAT.
`SEGNAME`	The segments in the description that are created when the synonym is generated. The segment names follow a logical format to provide uniqueness within the file.
`FIELD`	The field name from the data area.
`GROUP`	The fields from the data areas that were redefined or that were defined as arrays.
`USAGE`	The display format and length of the field. This attribute determines how the value is displayed in reports. Values are determined based on the format and length specified by the ACTUAL attribute.
`ACTUAL`	The format and length of the field as described in the data area.

Reference: Access File Attributes

Attribute	Description
`SEGNAME`	The name of the Master File segment that describes the stored procedure's input parameters. If the stored procedure does not have input parameters, the synonym generation process creates a dummy segment.
`STPNAME`	The name of the stored procedure. You specify the value when you create the stored procedure's synonym.
`TRGDBID`	The trigger's database ID. You specify the value when you create the stored procedure's synonym.
`TRGFILE`	The name of the trigger file. You specify the value when you create the stored procedure's synonym.
`OPTTRN`	The transaction option. You specify the value when you create the stored procedure's synonym.
`OPTPRM`	The parameter option. You specify the value when you create the stored procedure's synonym.
`OPTUPD`	The update option. This is based on values that you specify when you create the stored procedure's synonym.

Invoking an Adabas Stored Procedure

How to:

Invoke an Adabas Stored Procedure Using TABLE

Invoke an Adabas Stored Procedure Using SELECT

Invoke an Adabas Stored Procedure Using EX

Example:

Invoking the SAMP_LEM Adabas Stored Procedure

To invoke an Adabas stored procedure, you issue a a Data Manipulation Language (DML) request, also known as a TABLE command, or an SQL SELECT statement. (DML is iWay's internal retrieval language.) For information about the Data Manipulation Language, see the *iWay Stored Procedure Reference* manual.

Syntax: How to Invoke an Adabas Stored Procedure Using TABLE

The syntax for invoking an Adabas stored procedure using a TABLE command is

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
   .
   .
   .
END
```

where:

synonym

Is the synonym of the Adabas stored procedure you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

`*`

Indicates that you want to display all indicated parameters.

`IF/WHERE`

Is used if you want to pass values to input parameters.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Syntax: How to Invoke an Adabas Stored Procedure Using SELECT

The syntax for invoking an Adabas stored procedure using a SELECT statement is

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
   .
   .
   .
END
```

where:

`synonym`

Is the synonym of the Adabas stored procedure you want to invoke.

`parameter`

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

`*`

Indicates that you want to display all indicated parameters.

`WHERE`

Is used if you want to pass values to input parameters.

You must specify the value of each parameter on a separate line.

`in-parameter`

Is the name of an input parameter to which you want to pass a value.

`value`

Is the value you are passing to an input parameter.

Syntax: How to Invoke an Adabas Stored Procedure Using EX

```
EX [app_name_space/]synonym  1=parm1_val,..., N=parmN_val
```

```
EX [app_name_space/]synonym  parm1_name=parm1_val,... ,
  parmN_name=parmN_val
```

```
EX [app_name_space/]synonym [, parm1_val [,...[, parmN_val]]]
```

where:

`app_name_space`

Is the apps directory name under which the synonyms are stored. This value is optional.

`synonym`

Is the user friendly name of a repository entry that represents the object to be executed in the vendor environment.

parm_name

Is the name of the parameter taken from the input metadata description.

1=parm1_val
N=parmN_val
parm1_name
parm1_val
parmN_val

Are the parameter values that match the input metadata description.

Note:

- Consecutive commas denote missing parameters in the positional form of the request.
- Values containing special characters (<equal sign> <space> <comma>) must be encapsulated in double or single quotation marks.

Example: Invoking the SAMP_LEM Adabas Stored Procedure

The following TABLE command invokes the SAMP_LEM Adabas stored procedure, passing three parameter values to it.

```
TABLE FILE SAMP_LEM
   PRINT #REF_RC SEG4.#FCHAR
   IF #REF_NAME EQ '12345678'
   IF #REF_COUNT EQ 123
   IF #RESET_PARM EQ 'RESET'
END
```

The following SELECT command invokes the SAMP_LEM Adabas stored procedure:

```
SQL
SELECT #REF_RC SEG4.#FCHAR FROM SAMP_LEM
   WHERE #REF_COUNT = 123
END
```

The following EX commands invoke the SAMP_LEM Adabas stored procedure:

```
EX SAMP_LEM 225,2004
EX SAMP_LEM AM=225,YEAR=2004
EX SAMP_LEM 1=225,2=2004
```

CHAPTER 4

Using the Adapter for Allbase

Topics:

The Adapter for Allbase allows applications to access Allbase data sources. The adapter converts application requests into native Allbase statements and returns optimized answer sets to the requesting application.

Preparing the Allbase Environment

For the Adapter for Allbase, no environment variables need to be set prior to starting the server.

Configuring the Adapter for Allbase

In this section:

Declaring Connection Attributes

Overriding the Default Connection

Controlling the Connection Scope

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

Set the Home Location Path

Example:

Declaring Connection Attributes Manually

Setting the Home Location Path

Operating System (Trusted Mode) is the only method by which a user can be authenticated when connecting to an Allbase database server. User ID and password used to log on to the operating system are automatically used to connect to the Allbase database server. The server data access agent impersonates an operating system user according to the server deployment mode. The agent process establishes a connection to an Allbase database server based on the impersonated operating system user credentials.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	Description
Data source	Allbase database name.
Home directory	Location of the Allbase database.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

```
ENGINE [SQLALB] SET DATABASE database_name
```

where:

`SQLALB`

Indicates the Adapter for Allbase. You can omit this value if you previously issued the SET SQLENGINE command.

`database_name`

Indicates the Allbase database name used for this connection. This parameter must be supplied.

Note: Parameters containing special characters should be enclosed in single quotation marks.

Example: Declaring Connection Attributes Manually

The following SET DATABASE command connects to the Allbase database server named SAMPLESERVER. To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLALB SET DATABASE sampleserver
```

Syntax: How to Set the Home Location Path

```
ENGINE [SQLALB] SET HOME location_path
```

where:

`SQLALB`

Indicates the Adapter for Allbase. You can omit this value if you previously issued the SET SQLENGINE command.

`location_path`

Indicates the location of the Allbase database used for this connection. This parameter must be supplied.

Note: Parameters containing special characters should be enclosed in single quotation marks.

Example: Setting the Home Location Path

```
ENGINE SQLALB SET HOME /opt/allbase/mydb
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLALB] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLALB`

Indicates the Adapter for Allbase. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Allbase database server named SAMPLENAME as the default Allbase database server:

```
ENGINE SQLALB SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLALB] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLALB`

Indicates the Adapter for Allbase. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Allbase Metadata

In this section:

Creating Synonyms

Data Type Support

CLOB Activation

Changing the Precision and Scale of Numeric Columns

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Allbase data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Allbase table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.
10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value.

11. Complete your table or view selection:

To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

Synonyms are created and added under the specified application directory.

A status window displays the message:

```
All Synonyms Created Successfully
```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)

Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLALB [NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLALB

Indicates the Data Adapter for Allbase.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR edaqa.nf29004 DBMS SQLALB
END
```

Generated Master File nf29004.mas

```
FILE=NF29004, SUFFIX=SQLALB ,$
SEGNAME=NF29004, SEGTYPE=S0 ,$
FIELD=DIVISION4,     'DIVISION4 ',     I11, I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, 'DIVISION_NA4 ', A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, 'DIVISION_HE4 ', I11, I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=NF29004 ,
TABLENAME=EDAQA.NF29004,
CONNECTION=local,
KEYS=1
WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the Allbase table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: `TABLENAME=[`*`owner.`*`]`*`table`*
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following chart provides information about the default mapping of Allbase data types to server data types:

Allbase Data Type	Data Type		Remarks
	USAGE	ACTUAL	
CHAR (n)	A(1...3996)	A(1...3996)	
VARCHAR (n)			
DECIMAL(p,s) p in (1...27) s in (0...27) DEC(27,0) ... default value	P14	P29.27	NUMERIC is a synonym for DECIMAL.
FLOAT(p) p in (25...53) Float(53) ... default value	D8	D20.2	DOUBLE PRECISION is a synonym for FLOAT(53).
FLOAT(p) OR REAL p(1...24) Float(24) ... default value	D8	D20.2	REAL is a synonym for FLOAT(24).
INTEGER	I4	I11	
SMALLINT	I4	I6	
DATE	A10	A10	
TIME	A8	A8	
DATETIME	A23	A23	
INTERVAL	A20	A20	

Allbase Data Type	Data Type		Remarks
	USAGE	**ACTUAL**	
BINARY(n) in (1...3996) LONG VARBINARY(n) in (2^{31} - 1) BLOB BLOB	A(1...3996)	A(1...3996)	
VARBINARY(n) in (1...3996)	A(1...3996)	A(1...3996)	
LONG BINARY	BLOB	BLOB	
LONG VARBINARY	BLOB	BLOB	

CLOB Activation

The default mapping for the Allbase data types CHAR and NCHAR is the data type ALPHA. The ALPHA data type maximum supported length is 4096 characters for TABLE/MODIFY and 32768 characters for API applications. You can perform SELECT, INSERT, and UPDATE on columns with these data types without using the server data type CLOB.

Syntax: How to Set Server CLOB Activation

To activate the support for the server data type CLOB, you must issue the following command in one of the supported server profiles:

```
ENGINE [SQLALB] SET CONVERSION LONGCHAR TEXT
```

where:

`SQLALB`

Indicates the Adapter for Allbase. You can omit this value if you previously issued the SET SQLENGINE command.

`TEXT`

Activates long character support using data type CLOB. ALPHA is the default value.

Changing the Precision and Scale of Numeric Columns

How to:

Change the Precision and Scale of Numeric Columns

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Change the Precision and Scale of Numeric Columns

```
ENGINE [SQLALB] SET CONVERSION RESET
ENGINE [SQLALB] SET CONVERSION format RESET
ENGINE [SQLALB] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLALB] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLALB`

Indicates the Adapter for Allbase. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

precision

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

scale

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLALB SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLALB SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLALB SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLALB SET CONVERSION RESET
```

Customizing the Allbase Environment

The Adapter for Allbase provides several parameters for customizing the environment and optimizing performance.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLALB] SET PASSRECS {ON|OFF}
```

where:

`SQLALB`

Indicates the Adapter for Allbase. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLALB] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLALB`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

`setting`

Is the optimization setting. Valid values are as follows:

`OFF` instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

`ON` instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

`FOCUS` passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

`SQL` passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLALB set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLALB set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLALB] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLALB`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLALB SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLALB SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLALB SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

CHAPTER 5

Using the Adapter for CICS Transactions

Topics:

The Adapter for CICS processes input parameters, creates and sends requests to the CICS Transaction Server, and creates an answer set based on the received response.

This adapter has three different transport layers:

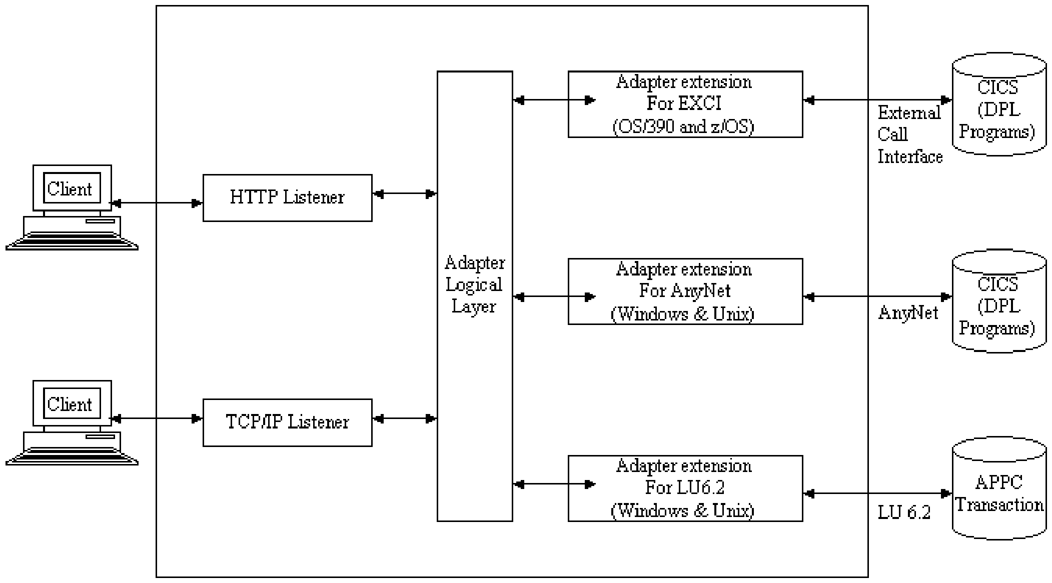
- EXCI for OS/390 and z/OS platforms.
- TCP/IP via AnyNet for UNIX and Microsoft Windows platforms.
- LU6.2 for UNIX and Microsoft Windows platforms.

Note:

- Throughout this chapter, the CICS Transaction Server is referred to as CICS.
- The Adapter for CICS replaces the Application Adapter Server module, SPGCICS.

Preparing the CICS Transactions Environment

The following diagram illustrates the basic flow of the Adapter for CICS Transactions:



This diagram illustrates how the client application communicates with the adapter. The communication to the CICS region depends upon the platform the adapter is running on. If you are using UNIX or Microsoft Windows, you can use a communication transport of TCP62 (AnyNet) for DPL programs or LU6.2 for APPC programs. If you are using OS/390 and z/OS, a direct CICS External Call Interface is used for DPL programs (use of LU6.2 for APPC programs from OS/390 and z/OS is not supported in this release of the server).

To provide for transparent execution of CICS programs, metadata describing the program input and output areas is held on the server. For DPL programs, the COBOL FD describing DFHCOMMAREA is used as a source for metadata creation. For APPC-based programs, the COBOL FD describing the storage layout for CICS SEND and RECEIVE calls is used as a source for metadata creation; RECEIVE for input, SEND for output.

A Web Console is available as the primary configuration and maintenance tool. It is used to add or change adapter communication parameters and in creating/maintaining the metadata associated with the programs to be executed.

Supported Platforms and Release Information for CICS Transactions

The following platforms are supported:

- Microsoft Windows
- UNIX
- OS/390 and z/OS

For Microsoft Windows and UNIX based servers that use AnyNet or LU6.2 to communicate with any CICS region, the following minimum software release levels on a target are required:

- OS/390 Version 2.6 or higher
- z/OS Version 1.1 or higher
- CICS Version 4 or higher

Note: To use TCP/IP, all VTAM options for AnyNet must be configured and AnyNet must be active.

For OS/390 and z/OS based servers, the following minimum software release levels are required:

- OS/390 Version 2.10 or higher
- z/OS Version 1.1 or higher
- CICS Version 4 or higher

Note: Because the EXCI option is being used, the adapter can only communicate with CICS regions that are running on the same LPAR. Also, the STEPLIB of the EDASTART JCL member of the configuration dataset needs to include the CICS SDFHEXCI library.

CICS and VTAM Configuration

In this section:

These topics provide the setup and communications configuration information the adapter requires to communicate with CICS.

After you configure any VTAM definitions, the major nodes they describe need to be started in the VTAM system. If you are using AnyNet, it should be tested from the Adapter for CICS platform.

AnyNet VTAM Definitions

Example:

The AnyNet product requires three major VTAM nodes for its operation. The following are examples of those nodes. Replace relevant parameters with your site specific values.

Example: Setting Up the Major Node for the AnyNet Listener

```
**********  TCP62 MAJOR NODE DEFINITION FOR ANYNET
TCP62    VBUILD TYPE=TCP,DNSUFX=IBI.COM,PORT=397
TCP62G   GROUP  ISTATUS=ACTIVE
TCP62L   LINE   ISTATUS=ACTIVE
TCP62P   PU     ISTATUS=ACTIVE,NETID=MYNET
```

Note: Port 397 is used in this example. On UNIX, using a port number above 1000 is recommended due to the root privileges associated with using a port number under 1000. On IBM, 397 is the default port; it is well established and frequently used.

Example: Setting Up the Major Node for Cross Domain Resources

If CDRDYN=YES is not specified in the ATCSTR00 VTAM startup parameter file, then the name of the machine that the Transaction Adapter is running on needs to be coded in the follow node:

```
        VBUILD TYPE=CDRSC
CDRSC62 NETWORK NETID=MYNET
CDRSC62 GROUP
EDABGR2 CDRSC ALSLIST=TCP62P
```

Note: EDABGR2 is the machine name on which the server is running. The machine name must be limited to eight bytes.

Example: Setting Up the Standard VTAM LU Definitions

```
        VBUILD TYPE=SWNET
MYNAMEPU  PU   ADDR=01,IDBLK=05D,IDNUM=10101,                X
               MAXPATH=3,                                    X
               PUTYPE=2,                                     X
               MODETAB=MTOS2EE,                              X
               DLOGMOD=PARALLEL,                             X
               ISTATUS=ACTIVE
MYLUNAME  LU   LOCADDR=2
```

LU6.2 VTAM Definitions

For LU6.2, one major VTAM node is required for its operation. See *Setting Up the Standard VTAM LU Definitions* on page 5-5 for an example of this node. Replace relevant parameters with your site specific values.

Example: Setting Up the Standard VTAM LU Definitions

```
        VBUILD TYPE=SWNET
MYNAMEPU  PU   ADDR=01,IDBLK=05D,IDNUM=10101,                X
               MAXPATH=3,                                    X
               PUTYPE=2,                                     X
               MODETAB=MTOS2EE,                              X
               DLOGMOD=PARALLEL,                             X
               ISTATUS=ACTIVE
MYLUNAME  LU   LOCADDR=2
```

CICS Connection and Sessions for Microsoft Windows and UNIX

For the Adapter for CICS to be able to connect to CICS (for both TCP/IP and LU6.2 where supported), Connection and Session definitions are required. The definitions are dependent on the platform on which the adapter is running.

If you deploy the adapter on Microsoft Windows or UNIX, the connection and sessions definitions described in *Using the CEDA View Connection Command* on page 5-6 and *Using the CEDA View Sessions Command* on page 5-6 are required.

Note: For DPL program execution, all users must be able to access and run transaction CPMI (the mirror transaction).

Example: Using the CEDA View Connection Command

The following is an example of a CEDA view connection command that illustrates what is required:

```
CEDA  View Connection(  MYLU )
   Connection     :  MYLU
   Netname        :  MYLUNAME
   ACcessmethod   :  Vtam           Vtam | IRc | INdirect | Xm
   PRotocol       :  Appc           Appc | Lu61 | Exci
   SIngless       :  No             No | Yes
   DAtastream     :  User           User | 3270 | SCs | STrfield | Lms
   INService      :  Yes            Yes | No
   ATtachsec      :  Verify         Local | Identify | Verify | Persistent
```

Example: Using the CEDA View Sessions Command

The following is an example of a CEDA view sessions command that illustrates what is required:

```
CEDA View Sessions(  MYLUNAME )
   Sessions       :  MYLUNAME
   Connection     :  MYLU
   MOdename       :  PARALLEL
   Protocol       :  Appc                Appc | Lu61 |
   MAximum        :  008 , 000           0-999
```

The MAximum parameter determines the number of concurrent sessions available to process requests. Its first value should be in the range of 4 - 255, and the second one should always be set to zero.

Note: Log mode must support parallel sessions.

Configuring the Adapter for CICS Transactions

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to CICS, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to CICS takes place when the first query that references that connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the last SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the last SET CONNECTION_ATTRIBUTES command.

For detailed instructions about declaring connection attributes for your operating system, see *Configuring the Adapter on Microsoft Windows and UNIX* on page 5-8 or *Configuring the Adapter on OS/390 and z/OS* on page 5-17.

Configuring the Adapter on Microsoft Windows and UNIX

How to:

Configure Communication on Windows and UNIX From the Web Console

Declare Connection Attributes From the Web Console on Windows and UNIX

Declare Connection Attributes Manually on Windows and UNIX

Example:

Sample Node Definitions Using TCP/IP

Sample Node Definitions Using LU6.2

Configuring the adapter consists of the following two steps:

- Configure Communication on Windows and UNIX From the Web Console.
- Declare Connection Attributes From the Web Console on Windows and UNIX.

Procedure: How to Configure Communication on Windows and UNIX From the Web Console

1. Start the Web Console and click *Listeners* in the navigation pane.
2. Expand the *Add* folder, and click *CICST.*

3. Type the appropriate values for your configuration.

Keyword	**Description**

NODE	8-character string
	
	Internal default: none
	
	Defines a logical name of a node block, which must be unique in the file. The logical name can be a maximum of 8 characters, must begin with a letter, and can include any characters, except semicolon and equal sign. A node block contains keyword-value pairs, which are enclosed in a BEGIN-END statement. The following is an example of a node block:
	
	`NODE = ABC`
	`BEGIN`
	`   DESCRIPTION = client node block`
	`   CLASS = CLIENT`
	`   PROTOCOL = TCP`
	`   HOST = localhost`
	`   PORT = 8100`
	`END`

PARTNER_LU_NAME	string Internal default: none Defines the APPLID of the target CICS region.
LOCAL_LU_NAME	string Internal default: none Defines the LU name to be used. This value is taken from an LU entry in the major node. The value is also specified in the CICS connection/sessions definition.
MODE_NAME	string Internal default: none Defines the DLOGMODE parameter value.

Keyword	Description
CODEPAGE	string Internal default: none Defines the code page that the CICS region is using.
COMMUNICATION	Select one of the following: • TCP/IP for DPL program execution. • LU6.2 for APPC program execution. Internal default: none Your selection in this field determines what additional configuration information that is required.
Additional required information for TCI/IP	
VTAM_NETWORK_ID	string Internal default: none Defines the VTAM network ID. This value is taken from the AnyNet listener major node under parameter NETID.
CONNECT_LIMIT	number of seconds. Internal default: none Defines the maximum time, in seconds, that the client waits for a connection response from the AnyNet environment. Provide a value greater than 0. (Do not enter a value of -1.)

Keyword	Description
PROXY	string Internal default: none Defines the name of the supplied proxy (user-written) program, which converts the Full Function Server calling syntax to the syntax that is valid for the CICS program to be executed. This parameter is *optional,* and should be left blank for normal operations. Use it only at the direction of your local support personnel.
MIRROR	string Internal default: none Defines the IBM-supplied mirror transaction CPMI that the Adapter for CICS Transactions calls by default. If this transaction cannot be used, you can create another transaction name to point to the IBM-supplied mirror program DFHMIRS. Enter the alternate name in the MIRROR field.
NATPROG	string Identifies the Adabas Natural program to be executed by a CICS transaction. Internal default: none
ATTACH_SECURITY	VERIFY or LOCAL Internal default: none Defines the (Attach) security type that was specified in the CICS Connection definition under the ATtachsec parameter. A number of values can be specified, but only Verify and Local are supported. Select the appropriate value for this configuration.
HOST	string Internal default: none for client; all IPs for listener Defines the host that a client is connecting to or an IP address that a listener is listening on.

Keyword	**Description**
SERVER_ID	string Internal default: current machine Defines the machine name where the server is configured with the CICS AGENT NODE (Listener).
ANYNET_PORT	positive integer number Internal default: none Defines the port number that the AnyNet product is using. The default AnyNet port is 397, but any number can be used. For UNIX installations, a port number above 1000 is recommended.
Additional required information for LU6.2	
PROXY	string Internal default: none Defines the name of the supplied proxy (user-written) program, which converts the Full Function Server calling syntax to the syntax that is valid for the CICS program to be executed. This parameter is *optional,* and should be left blank for normal operations. Use it only at the direction of your local support personnel.
SIDE_INFO	string Internal default: none SIDE_INFO is a table with definitions for all partners to the currently active CICS system. CICS implements the side information table by means of the PARTNER resource. Partner resources are defined by the CEDA DEFINE command. To become known to an active CICS system, a defined partner resource must be installed using the CEDA INSTALL command.

Keyword	Description
ATTACH_SEC	VERIFY or LOCAL Internal default: none Defines the (Attach) security type that was specified in the CICS Connection definition under the ATtachsec parameter. A number of values can be specified, but only Verify and Local are supported. Select the appropriate value for this configuration.
HOST	string Internal default: none Defines the DNS name of the host LPAR of the target CICS region. You can also code the four part IP address.

Tip: You can obtain details about each entry by clicking the *?* icon to the left of any parameter field to access the Web Console Help.

4. Click *Save and Restart*.

The configuration procedure creates three nodes for TCP/IP and one node to Lu6.2. For illustrations, see *Sample Node Definitions Using TCP/IP* on page 5-13 and *Sample Node Definitions Using LU6.2* on page 5-14.

Example: Sample Node Definitions Using TCP/IP

```
NODE = LST_CICS
BEGIN
  PROTOCOL = CICS
  PORT = 8139
  CLASS = AGENT
END
NODE = SPGCSRV
BEGIN
  PROTOCOL = CICS
  PORT = 8139
  CLASS = AGENT
  TARGET = INTERNAL
  HOST = IBIMVS
  CODEPAGE = 037
  ANYNET_PORT = 397
END
NODE = CICSCLNT
```

```
BEGIN
  PROTOCOL = CICS
  CLASS = CLIENT
  TARGET = CICS
  INTEGER = HOBF
  FLOAT = IBM
  PARTNER_LU_NAME = EDBGM010
  LOCAL_LU_NAME = T29DPB04
  MODE_NAME = PARALLEL
  HOST = SPGCSRV
  CODEPAGE = 037
  COMMUNICATION = TCP
  VTAM_NETWORK_ID = USIBINET
  CONNECT_LIMIT = 60
  MIRROR = CPMI
  ATTACH_SECURITY = VERIFY
  SECURITY = C2B591CC2A92650028E8A570F3BE36F3
END
```

Example: Sample Node Definitions Using LU6.2

```
NODE = CICSTLU6
BEGIN
  PROTOCOL = CICS
  CLASS = CICSCLIENT
  TARGET = CICS
  PARTNER_LU_NAME = EDBGM010
  LOCAL_LU_NAME = T29DPB41
  MODE_NAME = PARALLEL
  HOST = IBIMVS
  CODEPAGE = 037
  COMMUNICATION = LU62
END
```

Procedure: How to Declare Connection Attributes From the Web Console on Windows and UNIX

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *Procedures* group folder, then expand the *CICS TRAN* adapter folder and click a connection. The Add CICS TRAN to Configuration pane opens.
3. Provide valid values for all input fields.

Field	**Description**
Connection name	Is a logical name used to identify a specific set of connection attributes.
Node name	Is a name of the NODE definition in the ODIN configuration file.
Security	There are two methods by which a user can be authenticated when connecting to the Adapter for CICS: • **Explicit.** The user ID and password are explicitly specified for each connection and passed to the adapter, at connection time, for authentication. • **Trusted.** The adapter connects to CICS as an operating system login using the credentials of the operating system user impersonated by the server data access agent.
User	Is the user name by which you are known to the adapter. This field is only displayed when the security mode of the server is non-trusted.
Password	Is the password associated with the user name. This field is only displayed when the security mode of the server is non-trusted.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Once the adapter is configured, the connection name appears in the navigation pane under Configured. If you left-click the connection name, several options become available:

- **Add connection.** This option allows you to add another connection.
- **View Settings.** This option displays the current default connection.
- **Change settings.** This option enables you to revise conversion, optimization, session, and other settings in one place.
- **Remove.** This option removes the connection from the server.

Following the configuration, refer to *Creating Synonyms* on page 5-21 for information about how to create metadata.

Syntax: How to Declare Connection Attributes Manually on Windows and UNIX

Explicit. The user ID and password are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.

```
ENGINE CICSTRAN SET CONNECTION_ATTRIBUTES connection
  node_name/userid,password
```

Trusted. The adapter connects to CICS as an operating system login using the credentials of the operating system user impersonated by the server data access agent.

```
ENGINE CICSTRAN SET CONNECTION_ATTRIBUTES connection node_name/,
```

where:

`CICSTRAN`

Indicates the Adapter for CICS.

`connection`

Is a logical name used to identify this particular set of connection attributes. In the Access File, it becomes the CONNECTION=connection_name value. See *Access File Attributes* on page 5-32.

`node_name`

Is a name of the NODE definition in the ODIN configuration file.

`userid`

Is the primary authorization ID by which you are known to the Adapter for CICS Transactions.

`password`

Is the password associated with the primary authorization ID.

Configuring the Adapter on OS/390 and z/OS

How to:

Declare Connection Attributes From the Web Console on OS/390 and z/OS

Declare Connection Attributes Manually on OS/390 and z/OS

Configuring the adapter consists of specifying connection and authentication information for connections on OS/390 and z/OS.

Procedure: How to Declare Connection Attributes From the Web Console on OS/390 and z/OS

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *Procedures* group folder, then expand the *CICS TRAN* adapter folder and click a connection. The Add CICS TRAN to Configuration pane opens.
3. Provide valid values for all input fields.

Field	**Description**
Connection name	Is a logical name used to identify a specific set of connection attributes.
Application ID	Is the application ID of the CICS region.
Mirror	Is the name of the mirror transaction.
Security	There are two methods by which a user can be authenticated when connecting to the Adapter for CICS: • **Explicit.** The user ID and password are explicitly specified for each connection and passed to the adapter, at connection time, for authentication. • **Trusted.** The adapter connects to CICS as an operating system login using the credentials of the operating system user impersonated by the server data access agent.
User	Is the user name by which you are known to the adapter. This field is only displayed when the security mode of the server is non-trusted.

Field	Description
Password	Is the password associated with the user name. This field is only displayed when the security mode of the server is non-trusted.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Once the adapter is configured, the connection name appears in the navigation pane under Configured. If you left-click the connection name, several options become available:

- **Add connection.** This option allows you to add another connection.
- **View Settings.** This option displays the current default connection.
- **Change settings.** This option enables you to revise conversion, optimization, session, and other settings in one place.
- **Remove.** This option removes the connection from the server.

Following the configuration, refer to *Creating Synonyms* on page 5-21 for information about how to create metadata.

Syntax: How to Declare Connection Attributes Manually on OS/390 and z/OS

Explicit. The user ID and password are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.

```
ENGINE CICSTRAN SET CONNECTION_ATTRIBUTES connection/<userid>,<password>:
"APPL_ID EDBGM010 MIRROR EXCI"
```

Trusted. The user ID and password received from the client application are passed to the adapter for authentication. When a client connects to the server, the user ID and password are passed to the adapter for authentication and are not authenticated by the server.

```
ENGINE CICSTRAN SET CONNECTION_ATTRIBUTES connection/,:
"APPL_ID EDBGM010 MIRROR EXCI"
```

where:

`CICSTRAN`

Indicates the Adapter for CICS Transactions.

`connection`

Is a logical name used to identify this particular set of connection attributes. In the Access File, it becomes the CONNECTION=connection_name value. See *Access File Attributes* on page 5-32.

`userid`

Is the primary authorization ID by which you are known to the Adapter for CICS.

`password`

Is the password associated with the primary authorization ID.

Overriding the Default Connection

How to:

Change the Default Connection

Example:

Selecting CON1 as the Default Connection

Once the connections have been defined, the connection named in the last SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET CONNECTION_ATTRIBUTES command.

You can also include the CONNECTION attribute in the Access File of the business component specified in the current query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION attribute is automatically included in the Access File. This attribute supersedes the default connection.

Syntax: How to Change the Default Connection

```
ENGINE CICSTRAN SET DEFAULT_CONNECTION connection
```

where:

`CICSTRAN`

Indicates the Adapter for CICS.

`connection`

Is the connection name specified in a previously issued SET CONNECTION_ATTRIBUTES command. If this connection name has not been previously declared, a FOC44221 message is issued.

Note: If you use the SET DEFAULT_CONNECTION command more than once, the connection_name specified in the last command will be the active one.

Example: Selecting CON1 as the Default Connection

The following example selects the previously defined connection named CON1 as the default CICS connection:

```
ENGINE CICSTRAN SET DEFAULT_CONNECTION CON1
```

Managing CICS Transaction Metadata

When the server invokes a transaction or procedure, it needs to know how to build the request, what parameters to pass, and how to format an answer set from the response. For each transaction the server will execute, you must create a synonym that describes the layout of the request/response area.

Creating Synonyms

How to:

Create a Synonym From the Web Console on Windows and UNIX

Create a Synonym From the Web Console on OS/390 and z/OS

Create a Synonym Manually

Example:

Cobol Copybook for the ETPBRW8 Master File

Master File for the Transaction/Program

Access File for Transaction/Program

Sample CICS Transaction

Reference:

Managing Synonyms

Customization Options for COBOL File Descriptions

CHECKNAMES for Special Characters and Reserved Words

Master File Guidelines

Access File Attributes

Synonyms define unique names (or aliases) for each transaction or procedure that is accessible from the server. Synonyms are useful because they hide the underlying transaction or procedure from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows the input parameters and the response layout to be moved while allowing client applications to continue functioning without modification. For example, moving a transaction or procedure from a test region to production. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym From the Web Console on Windows and UNIX

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options for the adapter.
4. Select *Collection of COBOL* definitions and type the location of a collection of Cobol Copybooks. If you select this option you can choose all COBOL definitions in the collection (%) or filter by name and extension.
5. Click *Submit*. The Create Synonym pane (Step 2 of 2) opens.

   If selected, the filtered COBOL definition is displayed at the top of the pane.
6. Type the name of the synonym in the Synonym name box.
7. Type the name of the CICS program or APPC transaction in the Program name box.
8. Enter a COMMAREA value: type CALC in the input box to indicate maximal segment size or enter a value that represents the physical length of the data. The default value is 32500.
9. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

   When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.
10. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.
11. Optionally, select *Customize options* to customize how the COBOL FD is translated. For details about these options, see *Customization Options for COBOL File Descriptions* on page 5-26. If you do not select the check box, default translation settings are applied.

12. Enter the following parameters:

Parameter	**Description**
Select Application	Select an application directory. The default value is baseapp.
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

13. The Cobol Copybook(s) appear on the screen. You can choose the same Copybook or different Copybooks for input and output parameters.

Note: There can be no more than one set of input/output Copybooks per synonym.

14. Click *Create Synonym*. The result of the Create Synonym process is a Master File and an Access File, which are created and added under the specified application directory.

15. A status window displays the message:

```
All Synonyms Created Successfully
```

From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Procedure: How to Create a Synonym From the Web Console on OS/390 and z/OS

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options for the adapter.

4. Click *Location of COBOL definition*, then choose:
    - *Fully qualified PDS* to indicate a partitioned dataset on MVS.

      or
    - *Full HFS path* to indicate a hierarchical file structure on USS.
5. Click *Select all COBOL Definitions* or filter the COBOL FDs in a PDS by name.
6. Click *HFS Location of a single COBOL* definition.
7. Click *Submit*. The Create Synonym pane (Step 2 of 2) opens.

   If selected, the filtered COBOL definition is displayed at the top of the pane.
8. Select an application directory. The default value is baseapp.
9. Type the name of the synonym.
10. Type the name of the CICS program or APPC transaction.
11. Enter a COMMAREA value. Choose CALC to use the maximal segment size or enter a value that represents the physical length of the data. The default value is 32500.
12. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

    When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.
13. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.
14. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
15. The Cobol Copybook(s) appear on the screen. You can choose the same Copybook or different Copybooks for input and output parameters.

    Note: There can be no more than one set of input/output Copybooks per synonym.
16. Optionally, select *Customize options* to customize how the COBOL FD is translated. For details about these options, see *Customization Options for COBOL File Descriptions* on page 5-26. If you do not select the check box, default translation settings are applied.

17. Click *Create Synonym*. The result of the Create Synonym process is a Master File and an Access File, which are created and added under the specified application directory.

18. A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

    From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: Customization Options for COBOL File Descriptions

You can customize how a COBOL FD is translated by selecting *Customize options* in the Synonym creation pane. The following options are added to the right pane:

Parameter	**Definition**
On Error	Choose *Continue* to continue generating the Master File when an error occurs. Choose *Abort* to stop generating the Master File when an error occurs. Choose *Comment* to produce a commented Master File when an error occurs. Continue is the default value.
Hyphens as	Choose *No* to remove all hyphens in the COBOL name from the Master File field names. Choose *Yes* to replace all hyphens in the COBOL name with the underscore character. *Yes* is the default value.
Redefines	You may treat COBOL REDEFINE fields in one of three ways: Choose *Segments* to describe REDEFINE fields as segments in the Master File. Segments is the default value. Choose *Comments* to describe REDEFINE fields as comments in the Master File. Choose *None* to exclude REDEFINE fields altogether.
Occurs as	Choose *Segments* to describe OCCURS structures as segments, or *Field* otherwise. Segments is the default value.
Alignment	Choose *Yes* to insert slack bytes into a record to ensure alignment of numeric fields. Choose *No* to generate Master Files without alignment of slack bytes. No is the default value.

Parameter	**Definition**
Number of Hyphens to skip	FD Translator removes characters from the left, up to and including the Nth hyphen (depending on the value of N chosen in the menu). A value of *0* will retain the entire COBOL name. *All* means all prefixes will be removed. *0* is the default value.
Order fields	Choose *Yes* to generate Order fields in a Master File. Choose *No* to generate a Master File without Order fields. *No* is the default value.
Level 88 as	Choose *Comment* to include COBOL Level 88 fields as comments in the Master Files. Choose *Skip* to exclude level 88 fields. *Skip* is the default value.
Zoned Numeric Fields	Sets how zoned numeric values will be stored.
Numeric Fields Edit Options	
Zeroes	Choose *Suppress* to suppress printing of the digit zero for a field whose value is zero. Choose *Display* to display leading zeroes, for example, 00124. Choose *None* for no formatting.
Negative value	Choose *Bracket* to bracket negative values, for example, (1234). Choose *Credit* to credit negative values, for example, 1234 CR. Choose *None* for no formatting.
Dollar Sign	Choose *Floating* to display a floating dollar sign and commas, for example, $1,1234. Choose *Fixed* to display a fixed dollar sign and commas, for example, $ 1,1234. Choose *None* for no formatting.

Parameter	**Definition**
Separate Thousands	Choose *Comma* to include commas where appropriate. Choose *None* for no formatting.

For additional information about customization options, see Appendix D, *Translating COBOL File Descriptions*.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '| ', '&', ' (', ') ', ' <', '> ', '' ', ' =', ''''

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM  appname/synonym
       FOR      transaction_name
       DBMS     suffix
       [AT       connection]
       PARMS   '[INPDEF "input_file"]
                [OUTDEF "output_file"]
                [COMMAREA value]
                [FDPARMS FD_parms]'
                [CHECKNAMES][UNIQUENAMES]
END
```

where:

suffix

Indicates the Adapter for CICS Transactions.

connection

Is the connection_name as previously specified in a SET CONNECTION_ATTRIBUTES command. The default connection is used if this parameter omitted.

transaction_name

Is the name of transaction to be executed.

appname

Is the 1- to 64- character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the CICS transaction (maximum 64 characters for Microsoft Windows and UNIX server platforms).

COMMAREA

Is the keyword for the maximal size of COMMAREA for the Adapter for CICS. The options are:

CALC

If the maximal segment size is used.

value

The default value is 32500. You can enter a value that represents the physical length of the data.

`INPDEF`

Is the keyword for the file with input parameter descriptions.

`input_file`

Is the name of the COBOL FD file that contains information about input parameters. (Omit this entry if the transaction does not have input parameters).

`OUTDEF`

Is the keyword for the file with Input parameters description.

`output_file`

Is the name of the COBOL FD file that contains information about output parameters. Omit this entry if the transaction does not have output parameters.

`FDPARMS`

Is the keyword for the COBOL FD parameters.

`FD_`*`parms`*

Contains the values of the COBOL FD parameters.

`CHECKNAMES`

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

`UNIQUENAMES`

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

Reference: Master File Guidelines

Master Files contain definitions for input and output parameters used in the transaction. These parameters are described as separate segments.

- A dummy segment must be defined if input parameters are not used.
- Output parameters (if they exist) are defined in a descendant segment/segments.
- Fields from output segments that contain repeating values can be redefined as an Occurs segment. A value for the keyword POSITION defines the redefined field name.
- A multi-segment definition with the RECTYPE field is used when a transaction may return different answer sets.

Example: Cobol Copybook for the ETPBRW8 Master File

```
02  FILEREC.
  03  STAT            PIC X.
  03  NUMB            PIC X(6).
  03  NAME            PIC X(20).
  03  ADDRX           PIC X(20).
  03  PHONE           PIC X(8).
  03  DATEX           PIC X(8).
  03  AMOUNT          PIC X(8).
  03  COMMENT         PIC X(9).
```

Example: Master File for the Transaction/Program

```
FILENAME=ETPBRW8, SUFFIX=CICSTRAN, CODEPAGE=37, $
  SEGMENT=SEG1, SEGTYPE=S0, $
   GROUP=FILEREC, USAGE=A80, ACTUAL=A80, $
    FIELDNAME=STAT, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=NUMB, USAGE=A6, ACTUAL=A6, $
    FIELDNAME=NAME, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=ADDRX, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=PHONE, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=DATEX, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=AMOUNT, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=COMMENT, USAGE=A9, ACTUAL=A9, $
  SEGMENT=SEG11, SEGTYPE=S0, PARENT=SEG1, $
   GROUP=FILEREC, USAGE=A80, ACTUAL=A80, $
    FIELDNAME=STAT, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=NUMB, USAGE=A6, ACTUAL=A6, $
    FIELDNAME=NAME, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=ADDRX, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=PHONE, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=DATEX, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=AMOUNT, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=COMMENT, USAGE=A9, ACTUAL=A9, $
```

Reference: Access File Attributes

Keyword	Description
`SEGNAME`	Is the name of the input segment in the Master File.
`CONNECTION`	Indicates the connection_name as previously specified in a SET CONNECTION_ATTRIBUTES command. Defaulted to the default connection.
`TRANSACTION`	Is the name of the program to be executed.
`COMMAREA`	Is the maximal size of COMMAREA for the Adapter for CICS. The options are: `CALC` means that the maximal segment length will be used. `value` defaults to 32500. You can enter a value that represents the physical length of the data.

Example: Access File for Transaction/Program

```
SEGNAME=SEG1,CONNECTION=CICST1,TRANSACTION=etpbrw8,COMMAREA=32500,
  FLOAT=IBM,INTEGER=BigEndian,$
```

Note: Do not change the FLOAT or INTEGER parameter values.

Example: Sample CICS Transaction

```
001700 IDENTIFICATION DIVISION.
001800 PROGRAM-ID. ETPBRW8.
001900 ENVIRONMENT DIVISION.
002000 DATA DIVISION.
002100 WORKING-STORAGE SECTION.
002200 01  .
002300     02  WS-RESPONSE            PIC 9(8) BINARY.
002500     02  WS-RECORD-SUB          PIC 9 VALUE 1.
002510      02  WS-ACC-NUM                PIC X(6).
002600 01  COMMAREA.
002601     05  MESSAGES-OUT           PIC X(1000).
002602*---------------------------------------------------------------*
002700 LINKAGE SECTION.
002800 01 DFHCOMMAREA.
003000     05  WS-OUTPUT-RECORD OCCURS 4 TIMES.
003100         10  STAT                 PIC X.
003110         10  ACC-NUM              PIC X(6).
003300         10  NAME                 PIC X(20).
003400         10  ADDRX                PIC X(20).
003500         10  PHONE                PIC X(8).
003600         10  DATEX                PIC X(8).
003700         10  AMOUNT               PIC X(8).
003800         10  COMMENTS             PIC X(9).
003900*---------------------------------------------------------------*
004000 PROCEDURE DIVISION.
004100 PROCESS-INPUT.
004110     MOVE ACC-NUM(1) TO WS-ACC-NUM
004200     EXEC CICS STARTBR FILE('FILEA')
004300               RIDFLD(WS-ACC-NUM)
004400               GTEQ
004500     END-EXEC
004600     .
004700
004800        MOVE 1 TO WS-RECORD-SUB
004900
005000     PERFORM 4 TIMES
005100
005200     EXEC CICS READNEXT FILE('FILEA')
005300                        RIDFLD(WS-ACC-NUM)
005400                        INTO(WS-OUTPUT-RECORD(WS-RECORD-SUB))
005500                        RESP(WS-RESPONSE)
005600     END-EXEC
005700
005800        ADD 1 TO WS-RECORD-SUB
005900
006000     END-PERFORM.
006100*---------------------------------------------------------------*
006200     EXEC CICS RETURN
006300     END-EXEC.
006400     GOBACK
006500     .
```

Invoking a CICS Transaction

How to:

Invoke a CICS Transaction Using TABLE

Invoke a CICS Transaction Using SELECT

Invoke a CICS Transaction Using EX

Example:

Invoking the ETPBRW8 Transaction

To invoke a CICS transaction, you issue a Data Manipulation Language (DML) request, also known as a TABLE command, an SQL SELECT statement, or an EX command. (DML is iWay's internal retrieval language.) For information about the Data Manipulation Language, see the *iWay Stored Procedure Reference* manual.

Syntax: How to Invoke a CICS Transaction Using TABLE

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]

   .
   .
   .
END
```

where:

synonym

Is the synonym of the CICS transaction you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the transaction does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

`*`

Indicates that you want to display all indicated parameters.

`IF/WHERE`

Is used if you want to pass values to input parameters.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Syntax: How to Invoke a CICS Transaction Using SELECT

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]

   .
   .
   .
END
```

where:

synonym

Is the synonym of the CICS transaction you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the transaction does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

`*`

Indicates that you want to display all indicated parameters.

`WHERE`

Is used if you want to pass values to input parameters.

You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Syntax: How to Invoke a CICS Transaction Using EX

```
EX [app_name_space/]synonym  1=parm1_val,..., N=parmN_val

EX [app_name_space/]synonym  parm1_name=parm1_val,... ,
  parmN_name=parmN_val

EX [app_name_space/]synonym [, parm1_val [,...[, parmN_val]]]
```

where:

app_name_space

Is the apps directory name under which the synonyms are stored. This value is optional.

synonym

Is the user friendly name of a repository entry that represents the object to be executed in the vendor environment.

parm_name

Is the name of the parameter taken from the input metadata description.

1=parm1_val
N=parmN_val
parm1_name
parm1_val
parmN_val

Are the parameter values that match the input metadata description.

Note:

- Consecutive commas denote missing parameters in the positional form of the request.
- Values containing special characters (<equal sign> <space> <comma>) must be encapsulated in double or single quotation marks.

Example: Invoking the ETPBRW8 Transaction

The following TABLE command invokes the ETPBRW8 transaction, passing one parameter value to it.

```
TABLE FILE ETPBRW8
  PRINT SEG11.NUMB SEG11.STAT SEG11.NAME SEG11.ADDRX SEG11.PHONE
  IF SEG1.NUMB EQ '000700'
END
```

The output is:

```
NUMBER OF RECORDS IN TABLE=        4  LINES=      4


PAGE     1


NUMB    STAT  NAME                     ADDRX                   PHONE
----    ----  ----                     -----                   -----
000762        SUSAN MALAIKA            SAN JOSE,CALIFORNIA     22312121
000983        J. S. TILLING            WASHINGTON, DC          34512120
001222        D.J.VOWLES               BOBLINGEN, GERMANY      70315551
001781        TINA J YOUNG             SINDELFINGEN,GERMANY    70319990
```

The following SELECT command invokes the ETPBRW8 transaction:

```
SQL
SELECT  SEG11.NUMB, SEG11.STAT, SEG11.NAME, SEG11.ADDRX, SEG11.PHONE
FROM ETPBRW8
WHERE SEG1.NUMB = '000700';
END
```

The output is:

```
PAGE     1

NUMBER OF RECORDS IN TABLE=        4  LINES=      4

NUMB    STAT  NAME                     ADDRX                   PHONE
----    ----  ----                     -----                   -----
000762        SUSAN MALAIKA            SAN JOSE,CALIFORNIA     2231212
000983        J. S. TILLING            WASHINGTON, DC          34512120
001222        D.J.VOWLES               BOBLINGEN, GERMANY      70315551
001781        TINA J YOUNG             SINDELFINGEN,GERMANY    70319990
```

The following EX commands invoke the ETPBRW8 transaction:

```
EX ETPBRW8 '000700'

EX ETPBRW8 NUMB='000700'

EX ETPBRW8 1='000700'
```

The ETPBRW8 transaction browses the file starting from '000700' and, in these examples, returns values greater than '000700'.

Note: In order to use the EX command, you must set the SET EXORDER setting in either the FOCEXEC or in edasprof.prf.

```
SET EXORDER=PGM/FEX

EX baseapp/etpbrw8 NUMB='000700'
```

The output is:

```
FILE=SQLOUT,SUFFIX=FIX,$
  SEGNAME=SQLOUT,$
    FIELD=STAT,     E11       ,A1       ,A1       ,         ,$
    FIELD=NUMB,     E12       ,A6       ,A6       ,         ,$
    FIELD=NAME,     E13       ,A20      ,A20      ,         ,$
    FIELD=ADDRX,    E14       ,A20      ,A20      ,         ,$
    FIELD=PHONE,    E15       ,A8       ,A8       ,         ,$
    FIELD=DATEX,    E16       ,A8       ,A8       ,         ,$
    FIELD=AMOUNT,   E17       ,A8       ,A8       ,         ,$
    FIELD=COMMENT,  E18       ,A9       ,A9       ,         ,$

000762SUSAN MALAIKA       SAN JOSE,CALIFORNIA 2231212101 06
74$0000.00*********

000983J. S. TILLING       WASHINGTON, DC      3451212021 04
75$9999.99*********

001222D.J.VOWLES          BOBLINGEN, GERMANY  7031555110 04
73$3349.99*********

001781TINA J YOUNG        SINDELFINGEN,GERMANY7031999021 06
77$0009.99*********
```

CHAPTER 6

Using the Adapter for Informix C-ISAM, Micro Focus C-ISAM, FairCom c-tree ISAM, and Flat Files

Topics:

The Adapter for C-ISAM allows applications to access C-ISAM data sources. The adapter converts application requests into native C-ISAM statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the Environment

How to:

Identify the Micro Focus C-ISAM Installation Directory

Identify the Micro Focus C-ISAM Load Library Files

Identify the FairCom c-tree ISAM Load Library Files

The Informix Adapter for C-ISAM and Adapter for Flat Files do not require setting any environment variables.

For the Micro Focus Adapter for C-ISAM, you must set and export a $COBDIR UNIX environment variable, which identifies the installation directory to the operating system. You must add the location of the load libraries to the library path.

For the FairCom Adapter for c-tree ISAM, you must add the location of the c-tree ISAM load libraries to the library path. (This adapter is only supported in the Solaris environment.)

Syntax: How to Identify the Micro Focus C-ISAM Installation Directory

```
COBDIR=path
export COBDIR
```

where:

path

Is the location of the Micro Focus C-ISAM installation directory.

Syntax: How to Identify the Micro Focus C-ISAM Load Library Files

```
LD_LIBRARY_PATH=path
export LD_LIBRARY_PATH
```

where:

path

Is the location of the Micro Focus C-ISAM load library files.

Note: If the server is running with security on, the IBIPATH variable needs to be used in place of LD_LIBRARY_PATH.

Syntax: How to Identify the FairCom c-tree ISAM Load Library Files

```
LD_LIBRARY_PATH=path
export LD_LIBRARY_PATH
```

where:

path

Is the location of the FairCom c-tree ISAM load library files.

Note: If the server is running with security on, the IBIPATH variable needs to be used in place of LD_LIBRARY_PATH.

Configuring the Adapter

How to:

Configure the Adapter from the Web Console

Specify the Location of an Informix C-ISAM File

Specify the Location of a Micro Focus C-ISAM File

Specify the Location of a FairCom c-tree ISAM File

Specify the Location of a FairCom c-tree Parameter File

Specify the Location of a Flat File on UNIX and Windows

Specify the Location of a Flat File on OS/390

Specify the Location of a Flat File Residing on an HFS File System

You can configure the adapter from the Web Console, however, for each type of file, you must specify the location of the data files you want to access in a supported server profile.

Procedure: How to Configure the Adapter from the Web Console

To configure the adapter from the Web Console:

1. Start the Web Console and select *Data Adapters* from the left pane.
2. Expand the Add folder, expand the Other DBMS group folder, then expand the adapter folder and click a connection. The Add to Configuration pane opens.
3. No input parameters are required. Simply, click *Configure*. The adapter is added to the Configured adapters folder.

Syntax: How to Specify the Location of an Informix C-ISAM File

```
FILEDEF file_name DISK file_location
```

where:

file_name

Is the data file name.

file_location

Is the full path and file name of the data file. Include the .dat extension for the data file name.

Syntax: How to Specify the Location of a Micro Focus C-ISAM File

```
FILEDEF file_name DISK file_location
```

where:

file_name

Is the data file name.

file_location

Is the full path and file name of the data file.

Note:

- For each data file there must be a corresponding index file, except for Micro Focus compression 8 files, which do not have separate index files. The index is, instead, embedded in the data file.
- Read/write permissions must be given for both types of files.
- For the Informix Adapter for C-ISAM and the Micro Focus Adapter for C-ISAM, read-only access is supported.

Syntax: How to Specify the Location of a FairCom c-tree ISAM File

```
FILEDEF file_name DISK file_location
```

where:

file_name

Is the data file name.

file_location

Is the full path and file name of the data file, without the file extension.

Note:

- For each data file, there must be a corresponding index file and a corresponding parameter file.

- Read/write permissions must be given for all the above mentioned files.
- By default, the parameter file name is assumed to be the file_location followed by the ".p" extension.
- You can override the default parameter file name by specifying it in the environment variable EDACTPARMF.
- For the Adapter for FairCom c-tree ISAM, read-only access is supported.

Syntax: How to Specify the Location of a FairCom c-tree Parameter File

```
EDACTPARMF=path
export EDACTPARMF
```

where:

path

Is the location of the FairCom c-tree parameter file.

Note:

- By default, the parameter file name is assumed to be the file_location (as specified in a FILEDEF command or the DATASET attribute of a Master File), followed by the ".p" extension.
- You can override the default parameter file name by specifying it in the environment variable EDACTPARMF. This is required when selecting data from multiple ISAM files in a single query through a join. For such queries, a parameter file that describes all the tables involved must exist, and its name must be specified in EDACTPARMF.

Syntax: How to Specify the Location of a Flat File on UNIX and Windows

FIXED data sets may be allocated with the FILEDEF command

```
FILEDEF FILE1 DISK filename [(LRECL lrecl RECFM recfm]
```

where:

filename

Is the full path and file name of the data file.

lrecl

Is the record length in bytes. This parameter is optional. If you omit it, it is calculated based on the metadata description.

The left parenthesis preceding the optional parameters is required.

recfm

Is the record format. Specify F for fixed format, V for variable format. This parameter is optional. If you omit it, the default is fixed format.

Syntax: How to Specify the Location of a Flat File on OS/390

FIXED data sets may be allocated using JCL

```
//FILE1 DD DISP=SHR, DSN=dataset_name
```

or file definition

```
FILEDEF FILE1 DISK //'dataset_name' [(LRECL lrecl RECFM recfm]
```

where:

dataset_name
: Is the fully qualified data set name.

Syntax: How to Specify the Location of a Flat File Residing on an HFS File System

FIXED data sets may be allocated using file definition

```
FILEDEF FILE1 DISK filename [(LRECL lrecl RECFM recfm]
```

where:

filename
: Is the full path and file name of the data file.

lrecl
: Is the record length in bytes. This parameter is optional. If you omit it, it is calculated based on the metadata description.

  The left parenthesis preceding the optional parameters is required.

recfm
: Is the record format. Specify F for fixed format, V for variable format. This parameter is optional. If you omit it, the default is fixed format.

Note: The Adapter for Flat Files supports read/write access (write access can only be granted by INSERT only).

Managing Metadata for C-ISAM and Flat Files

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the data types.

Creating Synonyms

How to:

Create a Synonym From the Web Console

Create a Synonym Manually

Example:

CREATE SYNONYM Syntax for C-ISAM Informix, C-ISAM Micro Focus, Faircom c-tree ISAM, and Fixed Files

Reference:

Managing Synonyms

Customization Options for COBOL File Descriptions

CHECKNAMES for Special Characters and Reserved Words

Synonyms define unique names (or aliases) for each C-ISAM or Flat file or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File that represents the server's metadata.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The Select Synonym Candidates pane (Step 1 of 2) opens.

4. Enter values for the following parameters:

Parameter	**Description**
Collection of Cobol definitions	

Parameter	**Description**
Directory path	Path to the directory containing multiple COBOL files.
File name File extension	If you wish to limit retrieval, you can enter a file name and/or file extension: • In the File name box, enter a full name or a partial name with a wildcard symbol %. A full name returns just that entry. A name with a wildcard symbol may return many entries. • In the File extension box, enter an extension with or without the wildcard symbol %.
Location of Data Files	
Directory path	Path to the directory containing multiple data files.
File name File extension	If you wish to limit retrieval, you can enter a file name and/or file extension: • In the File name box, enter a full name or a partial name with a wildcard symbol %. A full name returns just that entry. A name with a wildcard symbol may return many entries. • In the File extension box, enter an extension with or without the wildcard symbol %. **Note:** Filtering data files based on file extensions is not supported in the Adapter for Informix C-ISAM.

5. Click *Submit*. The Create Synonym pane (Step 2 of 2) opens.
6. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

   When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.
7. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

8. Optionally, select *Customize options* to customize how the COBOL FD is translated. For details about these options, see *Customization Options for COBOL File Descriptions* on page 6-10. If you do not select the check box, default translation settings are applied.

9. Enter the following additional parameters, as appropriate:

Parameter	**Description**
Select Application	Select an application directory. baseapp is the default value.
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

10. Select the tables for which you wish to create synonyms:

- To select all tables in the list, select the check box to the left of the *Default Synonym Name* column heading.
- To select specific tables, select the corresponding check boxes.

11. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

12. After choosing the tables, choose the corresponding data file name from the drop-down list:

- For Micro Focus CISAM, Fixed Files, and Faircom c-tree ISAM, choose the file name *without* an extension.
- For Informix C-ISAM choose the file name with the extension *.idx*.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File. This option is available for MicroFocus C-ISAM, Informix C-ISAM, FairCom c-tree ISAM. It is *not* available for Flat files since no Access File is created.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: Customization Options for COBOL File Descriptions

You can customize how a COBOL FD is translated by selecting *Customize options* in the Synonym creation pane. The following options are added to the right pane:

Parameter	**Definition**
On Error	Choose *Continue* to continue generating the Master File when an error occurs. Choose *Abort* to stop generating the Master File when an error occurs. Choose *Comment* to produce a commented Master File when an error occurs. Continue is the default value.
Hyphens as	Choose *No* to remove all hyphens in the COBOL name from the Master File field names. Choose *Yes* to replace all hyphens in the COBOL name with the underscore character. *Yes* is the default value.
Redefines	You may treat COBOL REDEFINE fields in one of three ways: Choose *Segments* to describe REDEFINE fields as segments in the Master File. Segments is the default value. Choose *Comments* to describe REDEFINE fields as comments in the Master File. Choose *None* to exclude REDEFINE fields altogether.
Occurs as	Choose *Segments* to describe OCCURS structures as segments, or *Field* otherwise. Segments is the default value.
Alignment	Choose *Yes* to insert slack bytes into a record to ensure alignment of numeric fields. Choose *No* to generate Master Files without alignment of slack bytes. No is the default value.

Parameter	**Definition**
Number of Hyphens to skip	FD Translator removes characters from the left, up to and including the Nth hyphen (depending on the value of N chosen in the menu). A value of *0* will retain the entire COBOL name. *All* means all prefixes will be removed. *0* is the default value.
Order fields	Choose *Yes* to generate Order fields in a Master File. Choose *No* to generate a Master File without Order fields. *No* is the default value.
Level 88 as	Choose *Comment* to include COBOL Level 88 fields as comments in the Master Files. Choose *Skip* to exclude level 88 fields. *Skip* is the default value.
Zoned Numeric Fields	Sets how zoned numeric values will be stored.
Numeric Fields Edit Options	
Zeroes	Choose *Suppress* to suppress printing of the digit zero for a field whose value is zero. Choose *Display* to display leading zeroes, for example, 00124. Choose *None* for no formatting.
Negative value	Choose *Bracket* to bracket negative values, for example, (1234). Choose *Credit* to credit negative values, for example, 1234 CR. Choose *None* for no formatting.
Dollar Sign	Choose *Floating* to display a floating dollar sign and commas, for example, $1,1234. Choose *Fixed* to display a fixed dollar sign and commas, for example, $ 1,1234. Choose *None* for no formatting.

Parameter	**Definition**
Separate Thousands	Choose *Comma* to include commas where appropriate. Choose *None* for no formatting.

For additional information about customization options, see Appendix D, *Translating COBOL File Descriptions*.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '|', '&', ' (', ') ', ' <', '> ', " ', ' =', '"'

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR filename.cbl AT cobol_file_location
DATABASE 'data_file_location' DBMS database
[CHECKNAMES] [UNIQUENAMES]
END
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, this application name must not be used.

synonym

Is the name to be assigned the resulting synonym.

filename.cbl

Is the name of the COBOL file description, with its extension.

cobol_file_location

Is the directory where the .cbl file is located.

data_file_location

For Micro Focus, FIX, and Faircom c-tree ISAM, this is the full path to the data file, including file name and extension.

For Informix, this is the full path to the .idx file.

database

Type of database. The values are:

MFCISAM	for MicroFOCUS CJSAM
CISAM	for Informix CISAM
CTCISAM	for Faircom c-tree ISAM
FIX	for Fixed Files

CHECKNAMES

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

UNIQUENAMES

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

Example: CREATE SYNONYM Syntax for C-ISAM Informix, C-ISAM Micro Focus, Faircom c-tree ISAM, and Fixed Files

Informix C-ISAM

```
CREATE SYNONYM baseapp/nf2906if FOR nf2906.cbl AT /u4/qa/edanxc/mf
DATABASE '/u4/qa/edanxc/inf_cisam/csam2906.idx' DBMS CISAM
```

MicroFocus C-ISAM

```
CREATE SYNONYM mf2906 FOR nf2906.cbl AT /qa/edanxc/cbl DATABASE
'/qa/edanxc/cisam/mfcs2906' DBMS MFCISAM
```

Faircom c-tree ISAM

```
CREATE SYNONYM ctnf2906 FOR nf2906.cbl AT /qa/edanxc/cbl DATABASE
'/qa/edanxc/ctree/ctcs2906.dat' DBMS CTCISAM
```

Fixed File

```
CREATE SYNONYM baseapp/nf2906 FOR nf2906.cbl AT c:\ibi\apps\baseapp
DATABASE 'c:\ibi\apps\baseapp
```

CHAPTER 7

Using the Adapter for DATACOM

Topics:

The Adapter for CA-DATACOM/DB allows applications to access DATACOM data sources. The adapter converts application requests into native DATACOM statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Note: Throughout this manual, CA-DATACOM/DB is referred to as DATACOM.

Preparing the DATACOM Environment

The Adapter for CA-DATACOM/DB operates in OS/390, z/OS, and USS environments and issues standard DATACOM calls for record retrieval. The adapter uses DATACOM's Boolean selection capabilities, enabling it to retrieve only records that satisfy a request. This reduces the number of I/Os involved in data retrieval.

Prior to configuring the Adapter for DATACOM/DB using the Web Console, you must edit the ISTART JCL that is used to initiate the server by allocating the DATACOM.CAILIB, DATACOM.CUSLIB, and CAIRIM CAI.CS30.CAILIB libraries to the server's STEPLIB.

Note: If user requirement tables (URTs) are kept in a library other than CUSLIB, you may need to add an additional library to STEPLIB. For details about URTs, see *Controlling Data Access With a User Requirements Table* on page 7-4.

Configuring the Adapter for DATACOM

You configure Adapter for DATACOM from the Web Console.

Declaring Connection Attributes

Users accessing the DATACOM DBMS are authenticated by the Operating System security package, as well as any authentication performed by the DBMS itself.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *Other DBMS* group folder, then expand the *DATACOM* folder and click a connection. The Add DATACOM to Configuration pane opens.
3. Supply values for the following parameters:

Field	Description
//CXX	Data dictionary that stores and manages descriptive data about the database.
//LXX	Log file.
//PXX	Diagnostic area.
User	Identifies a DATACOM user to the data dictionary.
Password	Password associated with the user ID.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

   Tip: Once the adapter is configured, the connection name appears in the navigation pane under *Configured*. If you left-click either the adapter name or the connection name, an appropriate set of options becomes available.

Controlling Data Access With a User Requirements Table

First-level access to DATACOM data sources and fields is controlled by the DATACOM User Requirements Table (URT). DATACOM uses the URT to provide file and database-level security, and password protection for element-level security.The URT used for the server structure must include all the DATACOM database IDs contained in the Access File. At run time, the adapter dynamically loads the URT's load module.

The DATACOM Data Driver (like any other DATACOM application) uses a DATACOM User Requirements Table to access the appropriate DATACOM tables. A User Requirements Table contains all tables that an application can access.

One DATACOM URT containing many DATACOM tables can be described by one or many iWay Master Files. The iWay Access File contains the name of URT that dynamically calls the DATACOM URT.

An example of JCL that creates a DATACOM URT is located in HOME.DATA in the member GENURT1 and GENURT2. For your convenience, the sample code is provided in this documentation.

Example: Sample JCL for User Requirements Table (GETURT1 and GENURT2)

```
//GENURT1 JOB 'PROG',MSGCLASS=X,CLASS=S
//*************************************************************
//* NAME:     GENURT1
//*
//* FUNCTION: STEP1 IN CREATING THE DATACOM URT
//*
//* SUBSTITUTIONS:-CHANGE "DATACOM.CAIMAC" TO THE NAME OF YOUR
//*                DATACOM MACRO LIBRARY.
//*               -CHANGE MACRO DBURTBL ENTRIES:
//*                TBLNAME   DATACOM-NAME OF TABLE   (EX PMF)
//*                DBID      DATACOM-ID OF DATABASE (EX 001)
//*               -CHANGE MACRO DBUREND ENTRIES:
//*                USRINFO   1-16 BYTES OF USER INFO TO BE PLACED
//*                          IN URT.
//*               -CHANGE SYSLMOD TO LIBRARY WHERE YOU ARE GOING
//*                KEEP URTS.
//*************************************************************
//ASMBL    EXEC  PGM=ASMA90,PARM='OBJECT,NODECK,LIST',REGION=1024K
//SYSLIB   DD DSN=DATACOM.CAIMAC,DISP=SHR       <==CHANGE
//         DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DSN=&&OBJPASS,UNIT=SYSDA,
//         SPACE=(TRK,10),DISP=(NEW,PASS)
//SYSUT1   DD DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2   DD DSN=&&SYSUT2,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSUT3   DD DSN=&&SYSUT3,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSPUNCH DD DUMMY
```

```
//SYSIN    DD *
         DBURSTR MULTUSE=YES
         DBURTBL TBLNAM=XXX,DBID=XXX,SEQBUFS=2,UPDATE=YES
         DBURTBL TBLNAM=XXX,DBID=XXX,SEQBUFS=2,UPDATE=YES
         DBURTBL TBLNAM=XXX,DBID=XXX,SEQBUFS=2,UPDATE=YES
         DBUREND USRINFO=*USER COMMENTS *
         END
/*
//LINKEDIT EXEC  PGM=IEWL,PARM='LET,NORENT,NCAL,LIST,MAP,SIZE=1024K'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(10,1))
//DBNTRY   DD DISP=(OLD,DELETE),DSN=&&OBJPASS
//SYSLMOD DD DSN=HLQ.URTLIB,DISP=SHR                    <==CHANGE
//SYSLIN    DD *
  INCLUDE DBNTRY
  NAME TESTURT1(R)
/*
//
```

```
//GENURT2 JOB 'PROG',MSGCLASS=X,CLASS=S
//***************************************************************
//* NAME:     GENURT2
//*
//* FUNCTION: STEP2 IN CREATING THE DATACOM URT
//*
//* SUBSTITUTIONS:-CHANGE "DATACOM.CAIMAC" TO THE NAME OF YOUR
//*                DATACOM MACRO LIBRARY.
//*               -CHANGE MACRO DBURINF ENTRIES:
//*                LOADNAM OBJECT MODULE CREATED FROM STEP1
//*               -CHANGE MACRO DBUREND ENTRIES:
//*                USRINFO   1-16 BYTES OF USER INFO TO BE PLACED
//*                          IN URT.
//*               -CHANGE //DATAMOD TO HLQ.HOME.DATA LIBRARY
//*               -CHANGE SYSLMOD TO LIBRARY WHERE YOU ARE GOING
//*                KEEP URTS.
//***************************************************************
//ASMBL    EXEC  PGM=ASMA90,PARM='OBJECT,NODECK,LIST',REGION=1024K
//SYSLIB   DD DSN=DATACOM.CAIMAC,DISP=SHR          <==CHANGE
//                 DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DSN=&&OBJPASS,UNIT=SYSDA,
//                 SPACE=(TRK,10),DISP=(NEW,PASS)
//SYSUT1   DD DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2   DD DSN=&&SYSUT2,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSUT3   DD DSN=&&SYSUT3,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSPUNCH DD DUMMY
//SYSIN    DD *
```

```
         DBURINF URTABLE=LOAD,OPEN=USER,USRNTRY=USNTRY,                X
                 LOADNAM=ALXURT1                  <==CHANGE
         DBUREND USRINFO=*USER COMMENTS*          <==CHANGE
         END
/*
//LINKEDIT EXEC  PGM=IEWL,PARM='LET,NORENT,NCAL,LIST,MAP,SIZE=1024K'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//DATAMOD  DD DSN=HLQ.HOME.DATA,DISP=SHR            <==CHANGE
 //DBNTRY   DD DISP=(OLD,DELETE),DSN=&&OBJPASS
//SYSLMOD  DD DSN=PMSAEP.DATACOM.URTLIB,DISP=SHR    <==CHANGE
//SYSLIN    DD *
 INCLUDE DATMOD(USNTRY)
 INCLUDE DBNTRY
 ENTRY USNTRY
 NAME ALXURT(R)
/*
//
```

DATACOM Overview and Mapping Considerations

In this section:

Data Structures

Mapping Structures in the Server

DATACOM is a flat file database management system. The DATACOM DATA DICTIONARY stores and manages descriptive data about a data source.

To access DATACOM data sources, you describe them to server, using server terminology and attributes. The descriptions are kept in a Master File and an associated Access File.

The Master File tells the server how to interpret the DATACOM data structure. It identifies the DATACOM:

- Elements
- Fields (for each of those elements)
- Field type and field length (as DATACOM stores them)

The Access File creates a bridge between the server and DATACOM. It translates server terminology into DATACOM syntax by identifying the appropriate DATACOM:

- Database IDs
- Logical files
- Native key fields

- Element security information

When the server receives a request, it:

- Looks for the Master File describing that data source.
- Finds the suffix DATACOM in the Master File. This tells the server the requested data is in a DATACOM data source.
- Loads the Adapter for DATACOM module. The adapter module then opens the Access File, in which it finds the DATACOM User Requirements Table (URT) name, the database ID(s), and file name(s).

The Adapter for DATACOM opens the URT and builds calls to DATACOM. The adapter retrieves the data and passes it to the server. The server then creates the answer set.

For details on describing data sources to a server and on Master Files and Access Files, and examples of DATACOM logical files, fields, and elements, see *Mapping Structures in the Server* on page 7-9.

Data Structures

A DATACOM structure is a collection of one or more logical data sources, identified by a unique 3-byte ID number. Each logical data source contains data records with a defined set of fields, elements, and keys, and is identified by a unique, alphanumeric 3-byte table name.

A data record consists of one or more elements. An element is the smallest logical unit of data a program can request; it has a unique 5-byte name. Elements consist of one or more contiguous fields and may have an associated security code. Because a field may belong to more than one element, the elements themselves may overlap.

DATACOM locates records through keys. Each data source may have up to 999 keys (not necessarily contiguous) consisting of 1-180 fields which total a length of 180 bytes. Each key is identified by a 5-byte name and a 3-byte numeric ID. The use of keys in DATACOM's Compound Boolean Selection Facility is transparent to the user.

The table below is the structure of the sample PERSON data source. It contains two keys, three elements with fields that overlap, and seven unique fields.

EMPLOYEE NUMBER	**EMPLOYEE NAME**	**STREET ADDRESS**	**CITY ADDRESS**	**STATE ADDRESS**	**ZIP CODE**	**SOCIAL SECURITY NUMBER**
ELEMENT #1						
	ELEMENT #2 IDEMP					

EMPLOYEE NUMBER	EMPLOYEE NAME	STREET ADDRESS	CITY ADDRESS	STATE ADDRESS	ZIP CODE	SOCIAL SECURITY NUMBER
			ELEMENT #3 ADEMP			
						ELEMENT #4 EMDTA

Note: The DATACOM DATA DICTIONARY listing of the PERSON data source does not identify the specific fields in each element. For this information, refer to the DATA DICTIONARY Element Field Report.

The Employee Record Element (EMDTA) contains all seven fields in the data source.

The Employee Address Element (ADEMP) contains six fields that represent the employee ID and address. These are:

```
NUMBER
NAME
STREET_ADDRESS
CITY_ADDRESS
STATE_ADDRESS
ZIP_CODE_LOC
```

The Employee Identification Element (IDEMP) contains the NUMBER and NAME fields. All fields within each element are contiguous.

Mapping Structures in the Server

Example:

Using the DATA DICTIONARY Element Field Report for EMDTA

Reference:

Element Field Report Headings

DATACOM terms equate to server terms as follows:

DATACOM Master File Definition	Server Master File Attributes
File	SEGMENT
Element	ALIAS for groups of fields
Field	FIELD
Field Length (LNGTH)	ACTUAL (DATACOM Format) USAGE (Server Format)

When you describe a DATACOM data source in a Master File:

- You can choose one or more elements from a DATACOM data source to build a single server segment. You do not have to specify all the elements in a particular data source.
- You must define each field in the element in the order in which it appears in the DATACOM data source. A DATACOM field becomes a server field.
- You can define two or more DATACOM elements that contain overlapping fields. However, you must define each overlapping field to the server with a unique name each time it appears in an element.
- Each DATACOM field you define to the server must have the 5-byte element name to which the DATACOM field belongs, together with a unique qualifier.
- For each DATACOM field, the DATACOM field length becomes the ACTUAL field length.
- The number of decimal positions that DATACOM indicates becomes the number of decimal positions in the USAGE format.

Example: Using the DATA DICTIONARY Element Field Report for EMDTA

The following example represents the DATA DICTIONARY Element Field Report for the Employee Record Element (EMDTA) from the PERSON data source. The table in *Element Field Report Headings* on page 7-10 explains the relevant column headings in the report.

```
ENTITY-TYPE:ELEMENT NAME:PERSONNEL.EMPLOYEE (001)PROD DESC:EMPLOYEE RE
AUTHOR:JOHN DOE CONTROLLER:HR DESIGNER COPY-VERSION:
FILE-NAME: PERSON-MASTER DBNAME: PMF ID: 002 DESC: PERSONNEL MASTER FILE

LV C FIELD-NAME..PAREN-NAME.DISPL LNGTH T J S DEC RPFAC
DESCRIPTION....VALUE
ALC-NAME COMPILER-NAME......LANGUAGE-COMMENT

01 S NUMBER 0 5 N R N 00001 EMPLOYEE NUMBER
EMNUMBER EM-IDENTIFICATION-NUMBER
01 S NAME 5 24 C L N 00001 EMPLOYEE NAME
EMNAME EM-IDENTIFICATION-NAME
01 S STREET-ADDRESS 29 24 C L N 00001 EMPLOYEE STREET ADDRESS
EMADDR EM-STREET-ADDRESS
01 S CITY-ADDRESS 53 15 C L N 00001 EMPLOYEE CITY
EMCITY EM-CITY-ADDRESS
01 S STATE-ADDRESS 68 2 C L N 00001 EMPLOYEE STATE CODE
EMSTATE EM-STATE-ADDRESS
01 S ZIP-CODE-LOC 70 5 C L N 00001 EMPLOYEE ZIP CODE
EMZIP EM-ZIP-CODE-LOC
01 S SOCIAL-SECURITY 75 5 D R Y 00001 EMPLOYEE SOCIAL SECURITY
EMSSN EM-SOCIAL-SECURITY-NUMBER
```

Reference: Element Field Report Headings

Column Heading	Definition
LV	COBOL copybook level corresponding to the field.

C	Field class:
	
	S= Simple
	C= Compound
	V= Value
	F=Filler

FIELD-NAME	DATACOM field name.
PARENT-NAME	PARENT field to which the field belongs if the field class is C or V, or if this is a redefined field.
DISPL	Byte displacement of the field in the data source.

Column Heading	Definition
LNGTH	Field length in bytes.

T	Field Type:
	
	B=Binary
	C=Character
	D=Packed decimal
	H=2-character hexadecimal field
	N=Zoned decimal
	T=PL/1 bit representation

J	Justification:
	
	L=Left
	R=Right

S	Signed field:
	
	Y=Yes
	N=No

DEC	Number of decimal places for a numeric field.
RPFAC	Number of times the field can repeat.
DESCRIPTION	Description of the field.
VALUE	Value of a COBOL copybook VALUE CLAUSE when the Field Class is V.

The following Master File and Access File describe the Employee Record Element from the DATACOM PERSON data source.

Master File

```
FILENAME=PERSON,SUFFIX=DATACOM
  SEGNAME=PERSON,SEGTYPE=S,$
    FIELD=EMP_NO ,ALIAS=EMDTA.EN   ,USAGE=P6  ,ACTUAL=Z5 ,$
    FIELD=NAME   ,ALIAS=EMDTA.NAME ,USAGE=A24 ,ACTUAL=A24,$
    FIELD=STREET ,ALIAS=EMDTA.STR  ,USAGE=A24 ,ACTUAL=A24,$
    FIELD=CITY   ,ALIAS=EMDTA.CITY ,USAGE=A15 ,ACTUAL=A15,$
    FIELD=STATE  ,ALIAS=EMDTA.STAT ,USAGE=A2  ,ACTUAL=A2 ,$
    FIELD=ZIP    ,ALIAS=EMDTA.ZIP  ,USAGE=A5  ,ACTUAL=A5 ,$
    FIELD=SSNO   ,ALIAS=EMDTA.SSNO ,USAGE=P9L ,ACTUAL=P5 ,$
```

Access File

```
TRACE=NO,USERTABLE=URTLOAD,$
SEGNAME=PERSON,DBID=002,TABLENAME=PMF,KEYNAME=EMPNO,$
```

Managing DATACOM Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the DATACOM data types.

For the Adapter for DATACOM to access DATACOM files, you must describe each file you use in a Master and Access File. The logical description of an DATACOM file is stored in a Master File, which describes the field layout. The physical attributes of the DATACOM file, such as the database number, file number, descriptors, and security are stored in the Access File.

Creating Synonyms

Synonyms define unique names (or aliases) for each DATACOM table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File, which represent the server's metadata.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The right pane displays selection options for the adapter (Step 1 or 4).

4. Enter the following parameter:

Parameter	**Description**
Data Dictionary ID	Enter the ID of the data dictionary that stores and manages descriptive data for the selected database(s).

5. Click *Next*. The Database Selection for Datacom pane (Step 2 of 4) opens. The entries you made on the previous pane are displayed at the top of this pane.
6. From the drop-down list of databases, select a database.
7. Click the *Filter Database* check box if you wish to filter the list of available databases. A Name input box is added to the pane.

   Type a string for filtering the list of databases, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all databases whose names begin with the letters ABC; %ABC to select databases whose names end with the letters ABC; %ABC% to select databases whose names contain the letters ABC at the beginning, middle, or end.
8. Click *Next*. The Selection of Table for Datacom pane (Step 3 of 4) opens. The entries you made on the previous pane (Database and filtering criteria) are displayed at the top of this pane.
9. From the drop-down list of tables, select a table from the selected database.
10. Click *Next*. The Selection of Elements for Datacom pane (Step 4 of 4) opens. The entries you made on the previous pane (Table and filtering criteria) are displayed at the top of this pane.
11. In the URT input box, enter the name of a user requirement table. This table defines the database and files that a user can access. Note that this entry is not required for creating a synonym, however, data cannot be accessed without it. For more information about the user requirements table, see *Controlling Data Access With a User Requirements Table* on page 7-4.
12. If appropriate, enter the following additional parameters:

Parameter	**Description**
Select Application	Select an application directory. The default value is baseapp.

Parameter	**Description**
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

13. Complete your selection by choosing elements. An element is the smallest unit of data. An element contains fields that can belong to more than one element. For security information regarding elements, see *How to Specify the Element Logical Record Security* on page 7-21.

To select all elements in the list, select the check box to the left of the *Element Name* column heading.

To select specific elements, select the corresponding check boxes.

14. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

A status window displays the message:

```
All Synonyms Created Successfully
```

15. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.

Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Master Files for DATACOM

In this section:

File Attributes

Segment Attributes

Field Attributes

There are three types of Master File declarations.

Each declaration must begin on a separate line. A declaration consists of attribute-value pairs separated by commas. A declaration can span as many lines as necessary, as long as no single keyword-value pair spans two lines.

Do not use system or reserved words as names for files, segments, fields, or aliases. Specifying a reserved word generates syntax errors.

File Attributes

Each Master File begins with a file declaration. The file declaration has two attributes, FILENAME and SUFFIX:

```
FILE[NAME]=filename, SUFFIX=DATACOM [,$]
```

where:

`filename`

Identifies the Master File. The file name can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents.

`DATACOM`

Is the value for the Adapter for CA-DATACOM/DB.

Segment Attributes

Each table described in a Master File requires a segment declaration. The segment declaration consists of at least two attributes, SEGNAME and SEGTYPE:

```
SEGNAME=segname, SEGTYPE=type [,$]
```

where:

segname

Is the segment name that serves as a link to the actual DATACOM table name. It can consist of a maximum of 8 alphanumeric characters. It may be the same as the name chosen for FILENAME, the actual table name, or an arbitrary name.

The SEGNAME value in the Master File must be identical to the SEGNAME value specified in the Access File.

type

Indicates the type of segment sequencing:

`S` indicates the segments are logically sequenced in low to high order.

`U` indicates a unique segment.

Field Attributes

Each row in a table may consist of one or more columns. These columns are described in the Master File as fields with the following primary field attributes:

`FIELDNAME`

Identifies the name of a field.

`ALIAS`

Identifies the DATACOM 5-byte element short name to which the field belongs, along with a unique qualifier.

`USAGE`

Identifies how to display a field on reports.

`ACTUAL`

Identifies the datatype and length in bytes for a field.

You can get values for these attributes from the DATACOM DATA DICTIONARY Element Field Report.

The syntax for a field declaration is:

```
FIELD[NAME]=fieldname, ALIAS=elementname.qualifier,
[USAGE=]display_format, [ACTUAL=]storage_format ,$
```

where:

`fieldname`

Is the unqualified name of the field. You must describe the fields in the order in which they appear in the DATACOM element. You can find the DATACOM field names, their relative position within the element, and the DATACOM formats in the DATA DICTIONARY Element Field Report. This value must be unique within the Master File. The name can consist of a maximum of 48 alphanumeric characters (including any file name and segment name qualifiers and qualification characters you may later prefix to them in your requests). The name must begin with a letter. Special characters and embedded blanks are not recommended.

Note that some elements contain overlapping fields. You must define overlapping fields for those elements in which they appear, and assign a unique field name each time.

It is not necessary to describe all the columns of the DATACOM table in your Master File.

`elementname`

Is the DATACOM 5-byte element short name, followed by a period, to which the field belongs. You can find the name in the DATACOM DATA DICTIONARY listing of Master Files.

`qualifier`

Is the qualifier that is separated from the element name by a period. The qualifier is used to make the ALIAS unique and provide the field with an alternate identification. The total length of the ALIAS, including the element name and qualifier, cannot exceed 12 characters.

`display_format`

Is the display format. The value must include the field type and length and may contain edit options.

The data type of the display format must be identical to that of the ACTUAL format. For example, a field with an alphanumeric USAGE data type must have an alphanumeric ACTUAL data type.

Fields or columns with decimal or floating point data types must be described with the correct scale (s) and precision (p). Scale is the number of positions to the right of the decimal point. Precision is the total length of the field.

For the server, the total display length of the field or column *includes* the decimal point and negative sign. In SQL, the total length of the field or column *excludes* the decimal point and negative sign.

For example, a column defined as DECIMAL(5,2) would have a USAGE attribute of P7.2 to allow for the decimal point and a possible negative sign.

`storage_format`

Is the storage format of the DATACOM data type and length, in bytes, for the field. You can find this under LNGTH in the field report.

Access Files for DATACOM

How to:

Use the Header Logical Record

Use the Segment Logical Record

Specify the Element Logical Record Security

Each Master File must have a corresponding Access File. The file name of the Access File must be the same as that used for the Master File.

The Access File serves as a link between the server and DATACOM. It stores such information as the URT name, DATACOM Database ID(s), DATACOM table name(s), keyname(s), and element security information. In addition, the Access File provides you with the TRACE parameter, a useful debugging tool.

The Access File contains three types of logical records:

- Header record
- Segment record
- Element record

Each consists of a list of keyword and value pairs, separated by commas and terminated by a comma and dollar sign (,$). The list is free form and may span several lines; keywords may be in any order.

Syntax: How to Use the Header Logical Record

The header record contains the TRACE and USERTABLE attributes. The syntax is:

```
TRACE={NO|YES}, USERTABLE=name,$
```

where:

`YES`

Indicates a complete trace of executed DATACOM commands. It is recommended that you use TRACE only for debugging purposes.

`NO`

Indicates no trace. NO is the default value.

`name`

Is the name of the load module that supplies the DATACOM User Requirements Table (URT) information to be dynamically loaded. The URT must contain all the database IDs and file specified in the Master File, or you will not have access to the records.

Syntax: How to Use the Segment Logical Record

The segment record contains these attributes:

```
SEGNAME=segname, DBID=dbid, TABLENAME=tablename, [KEYFLD=keyfld,
IXFLD=ixfld,] KEYNAME=keyname,$
```

where:

segname

Identifies the segment name. The Master File SEGMENT attribute is the same as the SEGNAME attribute in the Access File.

dbid

Identifies the numeric, 3-byte DATACOM database ID.

tablename

Identifies the 3-byte DATACOM table name to which the segment identified by SEGNAME corresponds. You can find this name in the DATA DICTIONARY Element Field Report, under DBNAME.

keyfld

Is the server field name from the parent segment of a cross-reference; it is mandatory for everything but the root segment. You may also specify multiple field names, concatenated with the forward slash symbol (/) without blanks. The keyword value must be contiguous, and on the same line. The key list can span multiple lines.

ixfld

Is the server field name in the cross-referenced segment that establishes the cross-reference. It is mandatory for all but the root segment. The value(s) of these field(s) must have USAGE and ACTUAL formats comparable to the KEYFLD.

You may also specify multiple field names, concatenated with the forward slash symbol (/) without blanks. The keyword value may not span more than one line. The key list can span multiple lines.

keyname

Is the native key. This is in the DATA DICTIONARY Indented Report. In that report, MN stands for mandatory native.

Note: KEYNAME does not have to be the native key. If you wish to use a different keyname, that is acceptable. However, be aware that you may not get all the records.

Syntax: How to Specify the Element Logical Record Security

You must specify an element logical record if an element is defined to DATACOM with a security code.

```
ELEMENTNAME=element, SECURITY=security,$
```

where:

element

The five-byte name of the element.

security

The two-position hexadecimal value of the DATACOM security code.

Describing Multi-file Structures for DATACOM

In this section:

You can describe many different DATACOM data sources in one Master File and Access File pair. However, in the adapter, there must be at least one field in common between any parent and descendant pair of data sources. Each set of related fields must also have comparable USAGE and ACTUAL formats.

Each time you add a descendant segment, you must specify the PARENT attribute in the segment record. This will identify hierarchical relationships between the data sources.

In addition, you must specify the relationship between the fields in the Access File with the KEYFLD and IXFLD attributes:

- The value of the KEYFLD can be a simple field, a DEFINE field, or a list of fields.
- The value of IXFLD can be a simple field, or a list of fields.
- The length, and the format, of KEYFLD and IXFLD must be the same.

There are advantages to defining multiple data sources in a single structure:

- The multi-file structure creates a view of the DATACOM data source.
- You can describe up to 64 separate but related DATACOM logical files as segments in a single Master File. This will allow you to issue a request from any or all of the 64 segments defined in a single Master File without issuing a JOIN command.

To illustrate this concept, we will use the DATACOM data sources PERSON and PAYROLL.

The PAYROLL data source contains information about the wages, commissions, and taxes for each person in the PERSON data source.

Master File

```
FILENAME=PAYROLL,SUFFIX=DATACOM
  SEGNAME=PAYROLL,SEGTYPE=S,$
    FIELD=ENUM      ,ALIAS=PYIDT.EN     ,USAGE=P6    ,ACTUAL=Z5 ,$
    FIELD=ACTCODE   ,ALIAS=PYIDT.ACODE  ,USAGE=A1    ,ACTUAL=A1 ,$
    FIELD=ACTSTATUS ,ALIAS=PYIDT.ASTATU ,USAGE=A1    ,ACTUAL=A1 ,$
    FIELD=CURRATE   ,ALIAS=FIGSS.CRATE  ,USAGE=P10.2 ,ACTUAL=Z8 ,$
    FIELD=YTDWAGES  ,ALIAS=FIGSS.YWAGES ,USAGE=P10.2 ,ACTUAL=Z8 ,$
    FIELD=YTDCOMM   ,ALIAS=FIGSS.YCOMM  ,USAGE=P10.2 ,ACTUAL=Z8 ,$
    FIELD=YTDTAXES  ,ALIAS=FIGSS.YTAXES ,USAGE=P10.2 ,ACTUAL=Z8 ,$
```

Access File

```
TRACE=NO,USERTABLE=URTLOAD,$
SEGNAME=PAYROLL,DBID=002,TABLENAME=PAY,KEYNAME=EMPNO,$
```

Multi-file Master File

Consider an application that contains both the PERSON and PAYROLL data sources. It is reasonable to generate reports that identify the current salary rate for all employees, or a list of all the employees who have a particular salary rate.

For demonstration purposes, we will define both these DATACOM logical files in one Master File. In all but the top (or root) segment, you must specify the PARENT attribute. PARENT establishes the relationship between two segments.

```
PARENT=segname
```

where:

segname

The name of the child segment's parent. If you do not specify a PARENT attribute, it will default to the child segment's immediate predecessor as the parent.

The following example shows the new multi-file Master File with the PARENT attribute added to the descendent segment. The filename PERSPAY identifies the entire Master File.

```
FILENAME=PERSPAY,SUFFIX=DATACOM
  SEGNAME=PERSON,SEGTYPE=S,$
    FIELD=EMP_NO ,ALIAS=EMDTA.EN   ,USAGE=P6  ,ACTUAL=Z5  ,$
    FIELD=NAME   ,ALIAS=EMDTA.NAME ,USAGE=A24 ,ACTUAL=A24 ,$
    FIELD=STREET ,ALIAS=EMDTA.STR  ,USAGE=A24 ,ACTUAL=A24 ,$
    FIELD=CITY   ,ALIAS=EMDTA.CITY ,USAGE=A15 ,ACTUAL=A15 ,$
    FIELD=STATE  ,ALIAS=EMDTA.STAT ,USAGE=A2  ,ACTUAL=A2  ,$
    FIELD=ZIP    ,ALIAS=EMDTA.ZIP  ,USAGE=A5  ,ACTUAL=A5  ,$
    FIELD=SSNO   ,ALIAS=EMDTA.SSNO ,USAGE=P9L ,ACTUAL=P5  ,$
  SEGNAME=PAYROLL,SEGTYPE=S,PARENT=PERSON,$
    FIELD=ENUM      ,ALIAS=PYIDT.EN     ,USAGE=P6    ,ACTUAL=Z5 ,$
    FIELD=ACTCODE   ,ALIAS=PYIDT.ACODE  ,USAGE=A1    ,ACTUAL=A1 ,$
    FIELD=ACTSTATUS ,ALIAS=PYIDT.ASTATU ,USAGE=A1    ,ACTUAL=A1 ,$
    FIELD=CURRATE   ,ALIAS=FIGSS.CRATE  ,USAGE=P10.2 ,ACTUAL=Z8 ,$
    FIELD=YTDWAGES  ,ALIAS=FIGSS.YWAGES ,USAGE=P10.2 ,ACTUAL=Z8 ,$
    FIELD=YTDCOMM   ,ALIAS=FIGSS.YCOMM  ,USAGE=P10.2 ,ACTUAL=Z8 ,$
    FIELD=YTDTAXES  ,ALIAS=FIGSS.YTAXES ,USAGE=P10.2 ,ACTUAL=Z8 ,$
```

Multi-file Access File

The Access File for a multi-file structure must have a segment record for each logical segment you define in the Master File.

Each segment record, other than the root, must contain the KEYFLD and IXFLD attributes in addition to the Access File attributes described earlier. These new attributes identify the field(s) in an embedded cross-reference. The common field(s) serve as the cross-referenced link between the two data sources.

The PAYROLL data source is the child, or cross-referenced, segment in the following example. This is important because you identify cross-referencing fields by specifying KEYFLD and IXFLD in the cross-referenced segment record.

`KEYFLD`

The field name in the parent or host segment, or a list of up to five fields separated by a forward slash symbol (/).

`IXFLD`

The field name in the child or cross-referenced segment, or a list of up to five fields separated by a forward slash symbol (/).

In the example, the field EMP_NO in the PERSON data source is related to the field ENUM in the PAYROLL data source. Both have the same USAGE and ACTUAL data format and length (P6 and Z5); this is a server requirement.

Add the segment record for the PAYROLL data source to the Access File, including the KEYFLD and IXFLD attributes:

```
TRACE=NO,USERTABLE=URTLOAD,$
SEGNAME=PERSON,DBID=002,TABLENAME=PMF,KEYNAME=EMPNO,$
SEGNAME=PAYROLL,DBID=002,TABLENAME=PAY,KEYNAME=EMPNO,
  KEYFLD=EMP_NO,IXFLD=ENUM,$
```

Using Multiple Fields to Cross-reference Data Sources

With the Adapter for DATACOM, you can use up to five fields to establish a relationship or cross-reference between data sources.

For example, matching permutations of values of Z/Y/X/W/V in the KEYFLD and IXFLD attributes can be used to create new cross-referenced fields in the Access File. This is sometimes referred to as using concatenated keys.

To illustrate how to use concatenated keys, we will alter the PERSPAY multi-file Master and Access Files as shown in the following example:

1. Replace the field EMP_NO in the PERSON segment with the LAST_NAME and FIRST_NAME fields.
2. In the PAYROLL segment, replace the field ENUM with LN and FN.
3. Separate each field that is part of the cross-reference with a forward slash symbol (/).

```
FILENAME=PERPAY,SUFFIX=DATACOM
  SEGNAME=PERSON,SEGTYPE=S,$
    FIELD=LAST_NAME  ,ALIAS=EMDTA.LN   ,USAGE=A15 ,ACTUAL=A15 ,$
    FIELD=FIRST_NAME ,ALIAS=EMDTA.FN   ,USAGE=A10 ,ACTUAL=A10 ,$
    FIELD=NAME       ,ALIAS=EMDTA.NAME ,USAGE=A24 ,ACTUAL=A24 ,$
    FIELD=STREET     ,ALIAS=EMDTA.STR  ,USAGE=A24 ,ACTUAL=A24 ,$
    FIELD=CITY       ,ALIAS=EMDTA.CITY ,USAGE=A15 ,ACTUAL=A15 ,$
    FIELD=STATE      ,ALIAS=EMDTA.STAT ,USAGE=A2  ,ACTUAL=A2  ,$
    FIELD=ZIP        ,ALIAS=EMDTA.ZIP  ,USAGE=A5  ,ACTUAL=A5  ,$
    FIELD=SSNO       ,ALIAS=EMDTA.SSNO ,USAGE=P9L ,ACTUAL=P5  ,$
  SEGNAME=PAYROLL,SEGTYPE=S,PARENT=PERSON,$
    FIELD=LN        ,ALIAS=PYIDT.LNAME  ,USAGE=A15   ,ACTUAL=A15 ,$
    FIELD=FN        ,ALIAS=PYIDT.FNAME  ,USAGE=A10   ,ACTUAL=A10 ,$
    FIELD=ACTCODE   ,ALIAS=PYIDT.ACODE  ,USAGE=A1    ,ACTUAL=A1  ,$
    FIELD=ACTSTATUS ,ALIAS=PYIDT.ASTATU ,USAGE=A1    ,ACTUAL=A1  ,$
    FIELD=CURRATE   ,ALIAS=FIGSS.CRATE  ,USAGE=P10.2 ,ACTUAL=Z8  ,$
    FIELD=YTDWAGES  ,ALIAS=FIGSS.YWAGES ,USAGE=P10.2 ,ACTUAL=Z8  ,$
    FIELD=YTDCOMM   ,ALIAS=FIGSS.YCOMM  ,USAGE=P10.2 ,ACTUAL=Z8  ,$
    FIELD=YTDTAXES  ,ALIAS=FIGSS.YTAXES ,USAGE=P10.2 ,ACTUAL=Z8  ,$
```

These multiple fields will now become the KEYFLD and IXFLD attributes in the Access File.

Separate each field from the next with the forward slash symbol (/), and add these attributes to the PAYROLL segment record in the Access File:

```
TRACE=NO,USERTABLE=URTLOAD,$
SEGNAME=PERSON,DBID=002,TABLENAME=PMF,KEYNAME=LNAME,$
SEGNAME=PAYROLL,DBID=002,TABLENAME=PAY,KEYNAME=LNAME,
KEYFLD=LAST_NAME/FIRST_NAME,IXFLD=LN/FN,$
```

DATACOM Data Dictionary Master and Access Files

Example:

Data Dictionary PERSON-MASTER Indented Report

Data Dictionary Element Field Report for EMDTA

PERSON Master File and Access File

Data Dictionary PAYROLL-MASTER Indented Report

Data Dictionary Element Field Report for PAYROLL

PAYROLL Master File and Access File

PERSPAY Master File and Access File

You can describe many different DATACOM data sources in one Master File and Access File pair. However, in the adapter, there must be at least one field in common between any parent and descendant pair of data sources. Each set of related fields must also have comparable USAGE and ACTUAL formats.

This topic contains sample Master Files and Access Files, and examples of DATACOM logical files, fields, and elements from the DATACOM Data Dictionary Database listing.

Example: Data Dictionary PERSON-MASTER Indented Report

```
AREA PEOPLE 001 P PERSONNEL AREA PMF
FILE PERSON-MASTER 001 P PERSONNEL MASTER FILE PMF 002
RECORD PERSONNEL 001 P PERSONNEL RECORD
KEY PERSONNEL.NUMBER 001 P EMPLOYEE NUMBER KEY EMPNO 001 MN
KEY PERSONNEL.STATE-ZIP 001 P EMPLOYEE STATE ZIP KEY STZIP 002
ELEMENT PERSONNEL.EMPLOYEE 001 P EMPLOYEE RECORD ELEMENT EMDTA
ELEMENT PERSONNEL.FULL ADDRESS 001 P EMPLOYEE ADDRESS ELEMEN ADEMP
ELEMENT PERSONNEL.IDENTIFICATION 001 P EMPLOYEE IDENTIFICATION IDEMP
FIELD PERSONNEL.CITY-ADDRESS 001 P EMPLOYEE CITY
FIELD PERSONNEL.NAME 001 P EMPLOYEE NAME
FIELD PERSONNEL.NUMBER 001 P EMPLOYEE NUMBER
FIELD PERSONNEL.SOCIAL-SECURITY 001 P EMPLOYEE SOCIAL SECURITY NUMBER
FIELD PERSONNEL.STREET-ADDRESS 001 P EMPLOYEE STREET ADDRESS
FIELD PERSONNEL.ZIP-CODE-LOC 001 P EMPLOYEE ZIP CODE
```

Example: Data Dictionary Element Field Report for EMDTA

```
ENTITY-TYPE:ELEMENT NAME:PERSONNEL.EMPLOYEE (001)PROD DESC:EMPLOYEE RE
AUTHOR:JOHN DOE CONTROLLER:HR DESIGNER COPY-VERSION:
FILE-NAME: PERSON-MASTER DBNAME: PMF ID: 002 DESC: PERSONNEL MASTER FILE
LV C FIELD-NAME..PAREN-NAME.DISPL LNGTH T J S DEC RPFAC
DESCRIPTION.......VALUE
ALC-NAME COMPILER-NAME......LANGUAGE-COMMENT
01 S NUMBER 0 5 N R N 00001 EMPLOYEE NUMBER
EMNUMBER EM-IDENTIFICATION-NUMBER
01 S NAME 5 24 C L N 00001 EMPLOYEE NAME
EMNAME EM-IDENTIFICATION-NAME
01 S STREET-ADDRESS 29 24 C L N 00001 EMPLOYEE STREET ADDRESS
EMADDR EM-STREET-ADDRESS
01 S CITY-ADDRESS 53 15 C L N 00001 EMPLOYEE CITY
EMCITY EM-CITY-ADDRESS
01 S STATE-ADDRESS 68 2 C L N 00001 EMPLOYEE STATE CODE
EMSTATE EM-STATE-ADDRESS
01 S ZIP-CODE-LOC 70 5 C L N 00001 EMPLOYEE ZIP CODE
EMZIP EM-ZIP-CODE-LOC
01 S SOCIAL-SECURITY 75 5 D R Y 00001 EMPLOYEE SOCIAL SECURITY
EMSSN EM-SOCIAL-SECURITY-NUMBER
```

Example: PERSON Master File and Access File

Master File

```
FILENAME=PERSON,SUFFIX=DATACOM
  SEGNAME=PERSON,SEGTYPE=S,$
    FIELD=EMP_NO  ,ALIAS=EMDTA.EN   ,USAGE=P6  ,ACTUAL=Z5  ,$
    FIELD=NAME    ,ALIAS=EMDTA.NAME ,USAGE=A24 ,ACTUAL=A24 ,$
    FIELD=STREET  ,ALIAS=EMDTA.STR  ,USAGE=A24 ,ACTUAL=A24 ,$
    FIELD=CITY    ,ALIAS=EMDTA.CITY ,USAGE=A15 ,ACTUAL=A15 ,$
    FIELD=STATE   ,ALIAS=EMDTA.STAT ,USAGE=A2  ,ACTUAL=A2  ,$
    FIELD=ZIP     ,ALAS=EMDTA.ZIP   ,USAGE=A5  ,ACTUAL=A5  ,$
    FIELD=SSNO    ,ALIAS=EMDTA.SSNO ,USAGE=P9L ,ACTUAL=P5  ,$
```

Access File

```
TRACE=NO,USERTABLE=URTLOAD,$
SEGNAME=PERSON,DBID=002,TABLENAME=PMF,KEYNAME=EMPNO,$
```

Example: Data Dictionary PAYROLL-MASTER Indented Report

```
AREA PEOPLE 001 P PAYROLL AREA PAY
FILE PAYROLL-MASTER 001 P PAYROLL MASTER FILE PAY 001
RECORD PAYROLL 001 P PAYROLL RECORD

KEY PAYROLL.NUMBER 001 P EMPLOYEE NUMBER KEY EMPNO 001 MN
ELEMENT PAYROLL.ACTIVITY-CODE 001 P EMPLOYEE NUMBER ELEMENT PYIDT
ELEMENT PAYROLL.FIGURES 001 P EMPLOYEE PAY FIGURES FIGSS
ELEMENT PAYROLL.RECORD 001 P EMPLOYEE PAYROLL RECORD PAYRC
FIELD PAYROLL.ACTIVITY-CODE 001 P EMPLOYEE CODE
FIELD PAYROLL.ACTIVITY-STATUS 001 P EMPLOYEE PAY STATUS
FIELD PAYROLL.CURRENT-RATE 001 P EMPLOYEE RATE
FIELD PAYROLL.NUMBER 001 P EMPLOYEE NUMBER
FIELD PAYROLL.YTD-COMMISSION 001 P EMPLOYEE COMMISSION
FIELD PAYROLL.YTD-TAX 001 P EMPLOYEE YEAR T DATE TAXES
FIELD PAYROLL.YTD-WAGES 001 P EMPLOYEE YTD WAGES
```

Example: Data Dictionary Element Field Report for PAYROLL

```
ENTITY-TYPE:ELEMENT NAME:PAYROLL.RECORD (001)PROD DESC:EMPLOYEE PAYROLL
RECORD ELEMEN

FILE-NAME: PAYROLL-MASTER DBNAME: PAY ID: 001 DESC: PAYROLL MASTER FILE

LV C FIELD-NAME..PARENT-NAME.DISPL LNGTH T J S DEC RPFAC
DESCRIPTION...VALUE
ALC-NAME COMPILER-NAME......LANGUAGE-COMMENT

01 S NUMBER 0 5 N R N 00001 EMPLOYEE NUMBER
RCNUM RC-NUMBER
01 S ACTIVITY-CODE 5 1 C L N 00001 EMPLOYEE CODE
RCCODE RC-ACTIVITY-CODE
01 S ACTIVITY-STATUS 6 1 C L N 00001 EMPLOYEE PAY STATUS
RCSTATUS RC-ACTIVITY-STATUS
01 S CURRENT-RATE 7 8 N R N 00001 EMPLOYEE RATE
RCRATE RC-CURRENT-RATE
01 S YTD-WAGES 15 8 N R N 00001 EMPLOYEE YTD WAGES
RCWAGES RC-YTD-WAGE
01 S YTD-COMMISSION 23 8 N R N 00001 EMPLOYEE COMMISSION
RCCOMM RC-YTD-COMMISSION
01 S YTD-TAX 31 8 N R N 00001 EMPLOYEE YEAR T DATE TAXES
RCTAX RC-YTD-TAX
```

Example: PAYROLL Master File and Access File

Master File

```
FILENAME=PAYROLL,SUFFIX=DATACOM
  SEGNAME=PAYROLL,SEGTYPE=S,$
    FIELD=ENUM       ,ALIAS=PYIDT.EN     ,USAGE=P6     ,ACTUAL=Z5 ,$
    FIELD=ACTCODE    ,ALIAS=PYIDT.ACODE  ,USAGE=A1     ,ACTUAL=A1 ,$
    FIELD=ACTSTATUS  ,ALIAS=PYIDT.ASTATU ,USAGE=A1     ,ACTUAL=A1 ,$
    FIELD=CURRATE    ,ALIAS=FIGSS.CRATE  ,USAGE=P10.2  ,ACTUAL=Z8 ,$
    FIELD=YTDWAGES   ,ALIAS=FIGSS.YWAGES ,USAGE=P10.2  ,ACTUAL=Z8 ,$
    FIELD=YTDCOMM    ,ALIAS=FIGSS.YCOMM  ,USAGE=P10.2  ,ACTUAL=Z8 ,$
    FIELD=YTDTAXES   ,ALIAS=FIGSS.YTAXES ,USAGE=P10.2  ,ACTUAL=Z8 ,$
```

Access File

```
TRACE=NO,USERTABLE=URTLOAD,$
SEGNAME=PAYROLL,DBID=002,TABLENAME=PAY,KEYNAME=EMPNO,$
```

Example: PERSPAY Master File and Access File

Master File

```
FILENAME=PERSPAY,SUFFIX=DATACOM
  SEGNAME=PERSON,SEGTYPE=S,$
    FIELD=EMP_NO     ,ALIAS=EMDTA.EN    ,USAGE=P6   ,ACTUAL=Z5  ,$
    FIELD=NAME       ,ALIAS=EMDTA.NAME  ,USAGE=A24  ,ACTUAL=A24 ,$
    FIELD=STREET     ,ALIAS=EMDTA.STR   ,USAGE=A24  ,ACTUAL=A24 ,$
    FIELD=CITY       ,ALIAS=EMDTA.CITY  ,USAGE=A15  ,ACTUAL=A15 ,$
    FIELD=STATE      ,ALIAS=EMDTA.STAT  ,USAGE=A2   ,ACTUAL=A2  ,$
    FIELD=ZIP        ,ALIAS=EMDTA.ZIP   ,USAGE=A5   ,ACTUAL=A5  ,$
    FIELD=SSNO       ,ALIAS=EMDTA.SSNO  ,USAGE=P9L  ,ACTUAL=P5  ,$
  SEGNAME=PAYROLL  ,SEGTYPE=S,PARENT=PERSON,$
    FIELD=ENUM       ,ALIAS=PYIDT.EN     ,USAGE=P6     ,ACTUAL=Z5 ,$
    FIELD=ACTCODE    ,ALIAS=PYIDT.ACODE  ,USAGE=A1     ,ACTUAL=A1 ,$
    FIELD=ACTSTATUS  ,ALIAS=PYIDT.ASTATU ,USAGE=A1     ,ACTUAL=A1 ,$
    FIELD=CURRATE    ,ALIAS=FIGSS.CRATE  ,USAGE=P10.2  ,ACTUAL=Z8 ,$
    FIELD=YTDWAGES   ,ALIAS=FIGSS.YWAGES ,USAGE=P10.2  ,ACTUAL=Z8 ,$
    FIELD=YTDCOMM    ,ALIAS=FIGSS.YCOMM  ,USAGE=P10.2  ,ACTUAL=Z8 ,$
    FIELD=YTDTAXES   ,ALIAS=FIGSS.YTAXES ,USAGE=P10.2  ,ACTUAL=Z8 ,$
```

Access File

```
TRACE=NO,USERTABLE=URTLOAD,$
SEGNAME=PERSON,DBID=002,TABLENAME=PMF,KEYNAME=EMPNO,$
SEGNAME=PAYROLL,DBID=002,TABLENAME=PAY,KEYNAME=EMPNO,
KEYFLD=EMP_NO,IXFLD=ENUM,$
```

Data Retrieval Logic for DATACOM

This section describes the two types of DATACOM record retrieval commands generated by the Adapter for DATACOM:

- **Sequential Retrieval Commands: GSETL and GETIT.** These commands are issued by the adapter only for the root of the accessed subtree, and only if the request did not specify any record selection criteria on that segment.
- **Compound Boolean Selection Commands: SELFR and SELNR.** These commands are issued for DATACOM records when there are available selection criteria.

GSETL and GETIT

When the adapter determines sequential retrieval needs to be done for the root segment, it issues a GSETL command. It uses the KEYNAME, specified in the Access File, and establishes the starting position at the beginning (lowest value) of the key. Only one command is issued per request.

GETIT commands follow the GSETL command. They retrieve the element(s) for each root record indexed by the key. The adapter will request only the elements necessary to satisfy the request. If there are no sort fields in the request, the answer set will be produced, in ascending order, by the key. GETIT commands will be issued repeatedly until DATACOM issues a return code of 19 (End of Table).

As long as the KEYNAME in the Access File is the Native Key, the adapter will retrieve all records which correspond to a DATACOM table.

SELFR and SELNR

The adapter uses two types of selection criteria to construct the SELFR call to DATACOM for a record:

1. All the values supplied in WHERE statements that mention a field in the segment and have any of these types of operators:

```
EQ         GT         NE
IS         LE         CONTAINS
GE         LT         FROM...TO
```

   Note: The SELFR request does not use selection on defined fields.

2. DBA value selection criteria on the segment.

The adapter generates a SELFR command, using all available selection criteria on the segment. This builds a list of records which match the selection criteria and returns the first record. The list is built in the temporary index area of the DATACOM data source.

For a descendant record with SEGTYPE=U, the SELFR retrieves the unique descendant. No SELNR command is issued. Otherwise, the SELFR command is followed by SELNR command(s) which retrieve the records listed in the temporary index. A SELNR is issued repeatedly until DATACOM issues a return code of 14 (End/Beginning of Set).

To illustrate the sequence of calls the adapter must make to DATACOM to service a multi-segment request, we will use the PERSPAY data source.

```
FILENAME=PERSPAY,SUFFIX=DATACOM
  SEGNAME=PERSON,SEGTYPE=S,$
    FIELD=EMP_NO ,ALIAS=EMDTA.EN   ,USAGE=P6  ,ACTUAL=Z5  ,$
    FIELD=NAME   ,ALIAS=EMDTA.NAME ,USAGE=A24 ,ACTUAL=A24 ,$
    FIELD=STREET ,ALIAS=EMDTA.STR  ,USAGE=A24 ,ACTUAL=A24 ,$
    FIELD=CITY   ,ALIAS=EMDTA.CITY ,USAGE=A15 ,ACTUAL=A15 ,$
    FIELD=STATE  ,ALIAS=EMDTA.STAT ,USAGE=A2  ,ACTUAL=A2  ,$
    FIELD=ZIP    ,ALIAS=EMDTA.ZIP  ,USAGE=A5  ,ACTUAL=A5  ,$
    FIELD=SSNO   ,ALIAS=EMDTA.SSNO ,USAGE=P9L ,ACTUAL=P5  ,$
  SEGNAME=PAYROLL,SEGTYPE=S,PARENT=PERSON,$
    FIELD=ENUM      ,ALIAS=PYIDT.EN     ,USAGE=P6    ,ACTUAL=Z5 ,$
    FIELD=ACTCODE   ,ALIAS=PYIDT.ACODE  ,USAGE=A1    ,ACTUAL=A1 ,$
    FIELD=ACTSTATUS ,ALIAS=PYIDT.ASTATU ,USAGE=A1    ,ACTUAL=A1 ,$
    FIELD=CURRATE   ,ALIAS=FIGSS.CRATE  ,USAGE=P10.2 ,ACTUAL=Z8 ,$
    FIELD=YTDWAGES  ,ALIAS=FIGSS.YWAGES ,USAGE=P10.2 ,ACTUAL=Z8 ,$
    FIELD=YTDCOMM   ,ALIAS=FIGSS.YCOMM  ,USAGE=P10.2 ,ACTUAL=Z8 ,$
    FIELD=YTDTAXES  ,ALIAS=FIGSS.YTAXES ,USAGE=P10.2 ,ACTUAL=Z8 ,$
```

In the request:

```
SELECT NAME EMP_NO SSNO YTDWAGES YTDCOMM YTDTAXES
FROM PERSPAY
WHERE ACTSTATUS = 'A'
```

PERSON is a referenced segment; it is the root of the accessed subtree. NAME, EMP_NO, and SSNO are the names of fields on this segment.

However, there are no available selection criteria for the PERSON segment (ACTSTATUS is a field on the PAY segment). Therefore, the adapter will first issue a GSETL command for the PAYROLL MASTER File (PMF), using the EMP_NO Native Key. The GSETL command is followed by a GETIT command to retrieve the first root record, the EMDTA element.

Since server always processes from top-to-bottom, left-to-right, all the related descendants of this first root record must be retrieved before proceeding to the next root record.

Next, to generate the SELFR call to retrieve PAY data source records related to the PERSON parent, two pieces of selection criteria will be used: the value of EMP_NO (the KEYFLD) from the PMF record, and the selection criteria on ACTSTATUS.

The first SELFR call to DATACOM for the first root record will retrieve a PAY record with ENUM equal to EMP_NO, and ACTSTATUS equal to 'A'. Subsequent SELNR calls (SEGTYPE=S) may retrieve other records with those same values.

After DATACOM returns a 14 for the PAY data source, a second GETIT command will be generated for a PMF record. The value of this record's EMP_NO field is used, with ACTSTATUS EQ A, to generate a new set of SELFR/SELNR calls to retrieve related PAY records.

This process will be repeated until DATACOM returns a 19 for the GETIT command on the root, signifying all records have been retrieved and processed. If the request were:

```
SELECT NAME EMP_NO SSNO YTDWAGES YTDCOMM YTDTAXES
FROM PERSPAY
WHERE ACTSTATUS = 'A'
WHERE STATE = 'CT' OR 'RI' OR 'MA' OR 'VT' OR 'NH' OR 'ME'
```

SELFR/SELNR commands would be issued, instead of GSETL/GETIT commands with the five STATE values (STATE is a field on the PERSON segment). The process however, would be the same.

Finally, keep in mind:

- The adapter will retrieve all descendant records of one parent occurrence before it will retrieve the next parent record.
- The higher you put your selection criteria in the hierarchical structure, the more efficient processing will be.

CHAPTER 8

Using the Adapter for DB2

Topics:

The Adapter for DB2 allows applications to access DB2 data sources. The adapter converts data or application requests into native DB2 statements and returns optimized answer sets to the requesting program.

The adapter supports the execution of DB2 stored procedures and DB2 Cube Views.

Preparing the DB2 Environment

The Adapter for DB2 minimally requires the installation of the DB2 Client for the CLI type adapter or DB2 load library for the CAF adapter. The DB2 Client allows you to connect to a local or remote DB2 data source server.

For the server running on the OS/390 or z/OS environment utilizing the CLI interface FMID JDB7717, the DB2 component must be installed on the MVS system.

Procedure: How to Set Up the Environment on Microsoft Windows

On Microsoft Windows, the DB2 environment is set up during the installation of DB2.

Procedure: How to Set Up the Environment on UNIX

1. Specify the DB2 data source instance to access using the UNIX environment variable $DB2INSTANCE. For example:

   ```
   DB2INSTANCE=db2
   export DB2INSTANCE
   ```

2. Specify the location of the DB2 instance you wish to access using the UNIX environment variable $INSTHOME. For example, to set the home directory for the DB2 software to /usr/db2710, specify:

   ```
   INSTHOME=/usr/db2710
   export INSTHOME
   ```

3. Specify the path to the DB2 shared library using the UNIX environment variable $LD_LIBRARY_PATH. For example:

   ```
   LD_LIBRARY_PATH=$INSTHOME/sqllib/lib:$LD_LIBRARY_PATH
   export LD_LIBRARY_PATH
   ```

Note: If the server is running with security on, the LD_LIBRARY_PATH variable is ignored. In this case, you must use IBI_LIBPATH.

Procedure: How to Set Up the Environment on OS/390 and z/OS

Two types of Adapters for DB2 are available in this environment. Each one requires different pre-configuration steps prior to using the Web Console.

- **DB2/CAF:** Utilizes embedded SQL using the IBM/DB2 for OS/390 Call Attach Facility to access the DB2 RDBMS. The DB2 Bind is run by choosing the Yes radio button on the Web Console. This will execute a shell script $EDAHOME/bin/genidb2.sh, which is created during the installation. If you do not have permission to run the bind, choose *Create JCL* only on the Web Console. This option will create a member called GENDB2. If you do not have access to the Web Console, a job, GENIDB2, which is shipped with the server, must be edited to reflect your installation and submitted.

  Note: The GENIDB2 JCL will automatically be submitted by ISETUP if the DB2/CAF option is selected from within that procedure.

- **DB2/CLI:** Utilizes IBM/DB2 for OS/390 Call Level Interface calls to access the DB2 RDBMS.

  The Adapter for DB2/CLI makes use of the DB2 Plan DSNACLI to ensure that the GRANT EXECUTE command is issued for all users who will be using the adapter.

Procedure: How to Set Up the Environment on OS/400

To create tables with one-part names, a CURLIB must be set. Thereafter, the location can be anywhere in the library path. The first table found is the one used.

Accessing a Remote Database Server

Using the standard rules for deploying the DB2 Client, the server supports connections to:

- Local DB2 data source servers.
- Remote DB2 data source servers.

When using DB2/CLI to connect to a remote DB2 data source, the DB2 client catalog must contain an entry for the node where the remote data source resides and an entry for the remote database name.

XA Support

Read/write applications accessing DB2 data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see Appendix A, *XA Support*.

Configuring the Adapter for DB2

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Example:

In order to connect to the DB2 database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf*), or in a group profile (if supported on your platform).

You can declare connections to more than one DB2 database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the DB2 Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

   The Adapter for DB2/CLI configuration screen displays the following fields:

Attribute	Description
Datasource	DB2 database name that will be used for this connection. For OS/390 and z/OS, this is the DB2 location name as specified in the DB2 communications data source.
Security	There are three methods by which a user can be authenticated when connecting to an DB2 database server: **Explicit.** The user ID and password are explicitly specified for each connection and passed to DB2, at connection time, for authentication. **Password Passthru.** The user ID and password received from the client application are passed to DB2, at connection time, for authentication. This option requires that the server be started with security off. **Trusted.** The adapter connects to DB2 as a Windows login using the credentials of the Windows user impersonated by the server data access agent.
User	Authorization ID by which the user is known to DB2.
Password	Password associated with the user ID. The password is stored in encrypted form.

The Adapter for DB2/CAF Adapter configuration screen displays the following fields:

Attribute	Description
SSID	DB2 SSID that is to be accessed, specified in the GENIDB2 job. This value must be upper case.
Plan	Plan that was bound to DB2 via the GENIDB2 job. This value must be upper case.
Execute DB2 BIND Command	Indicates whether to bind the program. Yes is the default value and adds the following fields to the configuration window: DSNCLST Library Name, Owner, Isolation Level.
DSNCLST Library Name	Name of the DSNCLST library. For example: DSN710.SDSNCLST This attribute is available only if you choose to bind your program.
Owner	Authorization ID of the Owner of the Plan. This attribute is available only if you choose to bind your program
Isolation Level	Isolation level. CS is the default value. This attribute is available only if you choose to bind your program.
Grant	Grants plan execution to public.

4. Select the level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default.

   If you wish to define a user profile (user.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

5. When you have completed the fields, click *Configure* to generate the adapter and build a DB2 plan.

6. Click *Grant* to grant plan execution to public.

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to DB2, at connection time, for authentication.

```
ENGINE [DB2] SET CONNECTION_ATTRIBUTES [datasource]/userid,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to DB2, at connection time, for authentication. This option requires that the server be started with security off.

```
ENGINE [DB2] SET CONNECTION_ATTRIBUTES [datasource]/
```

Trusted authentication. The adapter connects to DB2 as a Windows login using the credentials of the Windows user impersonated by the server data access agent.

```
ENGINE [DB2] SET CONNECTION_ATTRIBUTES [datasource]/,
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`datasource`

Is the name of the DB2 data source you wish to access.

`userid`

Is the primary authorization ID by which you are known to DB2.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allows the application to access the DB2 database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the DB2 database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

The following SET CONNECTION_ATTRIBUTES command connects to a local DB2 database server using operating system authentication:

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES /,
```

DB2 CURRENT SQLID (OS/390 and z/OS)

How to:

Reset CURRENT SQLID

DB2 accepts two types of IDs, the primary authorization ID and one or more optional secondary authorization IDs. It also recognizes the CURRENT SQLID setting.

Any interactive user or batch program that accesses a DB2 subsystem is identified by a primary authorization ID. A security system, such as RACF, normally manages the ID. During the process of connecting to DB2, the primary authorization ID may be associated with one or more secondary authorization IDs (usually RACF groups). Each site controls whether it uses secondary authorization IDs. For more information about using the adapter in conjunction with external security packages, see the server manual for your platform.

The primary authorization ID is the same ID passed to the server at connect time. This user ID is then used to connect to the DB2 subsystem.

The DB2 Data Source Administrator may grant privileges to a secondary authorization ID that are not granted to the primary ID. Thus, secondary authorization IDs provide the means for granting the same privileges to a group of users. (The DBA associates individual primary IDs with a secondary ID and grants the privileges to the secondary ID.)

The DB2 CURRENT SQLID may be the primary authorization ID or any associated secondary authorization ID. At the beginning of the session, the CURRENT SQLID is the primary authorization ID.

Syntax: How to Reset CURRENT SQLID

You can reset the CURRENT SQLID in a stored procedure or a profile using the following adapter command

```
ENGINE [DB2] SET CURRENT SQLID = 'sqlid'
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`sqlid`

Is the primary or secondary authorization ID, which must be enclosed in single quotation marks as shown. All DB2 security rules are respected.

The CURRENT SQLID is the default owner ID for DB2 objects, such as tables or indexes, created with dynamic SQL commands. The CURRENT SQLID is also the sole authorization ID for GRANT and REVOKE commands. It must have all the privileges needed to create objects as well as GRANT and REVOKE privileges. In addition, the CURRENT SQLID is the implicit owner for unqualified table names.

Other types of requests, such as SQL SELECT, INSERT, UPDATE, or DELETE requests, automatically search for the necessary authorization using the combined privileges of the primary authorization ID and all of its associated secondary authorization IDs, regardless of the DB2 CURRENT SQLID setting.

The CURRENT SQLID setting remains in effect until the thread to DB2 is disconnected, when it reverts to the primary authorization ID.

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [DB2] SET DEFAULT_CONNECTION [connection]
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Controlling Connection Scope (OS/390 and z/OS)

The SET AUTODISCONNECT and AUTOCLOSE commands control the persistence of connections when using the adapter.

Controlling Connection Scope with AUTOCLOSE

SET AUTOCLOSE initiates the DB2 Call Attachment Facility (CAF) CLOSE operation. It determines how long a thread (the connection between the application program in the user's address space and the DB2 application plan) is open. The thread is not the same as the address space connection to DB2; that connection is controlled by the AUTODISCONNECT setting.

In the server, an application program is one of the following:

- The dynamic Adapter for DB2.
- A CALLPGM subroutine using embedded SQL (non-CLI).

Generally speaking, each program has a corresponding plan or package.

A site that installs a DB2 subsystem determines the maximum number of concurrent users (threads) the subsystem will support. Since each user requires enough virtual storage for their application plan, this setting controls the amount of storage the site wants to allocate to active DB2 users at any one time.

The CAF CLOSE command deallocates the DB2 thread, releasing the virtual storage for the application plan. DB2 requires that an existing thread to a plan be closed before a thread to another plan is opened. If a thread is closed without a subsequent OPEN operation, the closed thread becomes "inactive"; the user is still connected to DB2, but not to a particular application plan. The user (task) still owns the thread; it is not available to other users. To release the thread, the user must disconnect completely from DB2.

Note: The term pseudo-conversational describes the type of transaction processing provided when you use AUTOCLOSE ON COMMIT.

Controlling Connection Scope with AUTODISCONNECT

AUTODISCONNECT completely detaches the user's address space (or task) from DB2. This differs from CLOSE because after a CLOSE, the task is still connected to the DB2 subsystem and can open a thread to another plan. After a DISCONNECT, the task must reestablish its connection to DB2 before performing any data source work. The tasks that frequently issue the DISCONNECT command are connected to DB2 for shorter periods of time, allowing other tasks to connect and acquire threads as needed. However, there is significant system overhead associated with frequently connecting and disconnecting, and the possibility exists that no thread will be immediately available when the task attempts to reconnect.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Control Connection Scope

```
ENGINE [DB2] SET {AUTOCLOSE|AUTODISCONNECT} ON {FIN|COMMIT}
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`AUTOCLOSE`

Issues the DB2 Call Attach Facility (CAF) CLOSE operation.

`AUTODISCONNECT`

Issues the DB2 Call Attach Facility (CAF) DISCONNECT operation.

`FIN`

Disconnects automatically only after the server session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing DB2 Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the DB2 data types.

Creating Synonyms

How to:

Example:

Reference:

Synonyms define unique names (or aliases) for each DB2 table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

   Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

   - **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.

   - **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.

5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.

6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.

7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value. See *Controlling the Mapping of Large Character Data Types* for more information.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS DB2 [AT connection|AT '']
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

DB2

Indicates the Data Adapter for DB2.

AT *connection*

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

AT ' '

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local database server.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS DB2 AT DB1
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=DB2 ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4  ,MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25 ,MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4  ,MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DB1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Name of the table or view. This value can include a location or owner name as follows: `TABLENAME=[`*`location.`*`][`*`owner.`*`]`*`tablename`* **Note:** Location is valid only with DB2 CAF and specifies the subsystem location name.
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.

`DBSPACE`	Optional keyword that indicates the storage area for the table. For example:
	`datasource.tablespace`
	`DATABASE` *`datasource`*

`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following table lists how the server maps DB2 data types. Note that you can:

- Control the mapping of large character data types.
- Control the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

DB2 Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
CHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 254
VARCHAR (*n*)	A*n* TX50	A*n* TX	*n* is an integer between 1 and 32672
BLOB	BLOB	BLOB	Up to 2 gigabytes
CLOB	TX50	TX	Up to 2 gigabytes
SMALLINT	I6	I4	
INTEGER	I11	I4	Maximum precision is 11
BIGINT	P20	P10	Available on UNIX & Microsoft Windows only
DECIMAL (*p,s*)	P6	P8	*p* is an integer between 1 and 31 *s* is an integer between 0 and *p*
REAL	F9.2	F4	Maximum precision is 9
FLOAT	D20.2	D8	Maximum precision is 20
DATE	YYMD	DATE	
TIME	HHIS	HHIS	
TIMESTAMP	HYYMDm	HYYMDm	Supported as Read-only
LONG VARCHAR (*n*) in (1...32700)			Not supported
GRAPHIC			Not supported
VARGRAPHIC			Not supported

DB2 Data Type	Data Type		Remarks
	USAGE	**ACTUAL**	
LONG VARGRAPHIC			Not supported
DATALINK			Not supported

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of the DB2 data type VARCHAR. By default, the server maps this data type as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

The following table lists data type mappings based on the value of LONGCHAR:

DB2 Data Type	Remarks	LONGCHAR ALPHA		BLOB		LONGCHAR TEXT	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**	**USAGE**	**ACTUAL**
VARCHAR (*n*)	*n* is an integer between 1 and 32768	A*n*	A*n*	BLOB	BLOB	TX50	TX

Syntax: How to Control the Mapping of Large Character Data Types

```
ENGINE [DB2] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`ALPHA`

Maps the DB2 data type VARCHAR as alphanumeric (A). ALPHA is the default value.

`TEXT`

Maps the DB2 data type VARCHAR as text (TX). Use this value for WebFOCUS applications.

`BLOB`

Is equivalent to ALPHA. That is, it maps the DB2 data type VARCHAR as alphanumeric (A).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the DB2 data types VARCHAR. By default, the server maps this data type as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

DB2 Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
VARCHAR (*n*)	*n* is an integer between 1 and 32768	A*n*V	A*n*V	A*n*	A*n*

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE [DB2] SET VARCHAR {ON|OFF}
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the DB2 data type VARCHAR as variable-length alphanumeric (A*n*V).

`OFF`

Maps the DB2 data type VARCHAR as alphanumeric (A). OFF is the default value.

BLOB Activation (OS/390 and z/OS)

DB2 data types that support the *for bit data* attribute including VARCHAR(n) where n > 256 and LONG VARCHAR, can be supported in the server as Binary Large Objects (BLOBs). This support is for both read and write access.

Syntax: How to Activate BLOB

To activate this support, you must issue the following command in the one of the supported server profiles

```
ENGINE [DB2] SET CONVERSION LONGCHAR BLOB
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`BLOB`

Activates long binary support. ALPHA is the default value.

BLOB Read/Write Support (OS/390 and z/OS)

For DB2 data types VARCHAR (>256) and LONG VARCHAR which have the *for bit data* attribute, the server provides read and write support using three server remote procedures routines. These routines are:

Routine	Used to...
EDABS	Send binary image data to the server.
EDABE	Mark the end of the binary image.
EDABK	Purge the binary image from server storage.

The sequence of operations for the client application is:

1. Converts every binary byte of the image into 2 bytes of hexadecimal data and stores the result in an internal buffer. If the image is large, this could be split into manageable pieces.
2. Sends the converted binary bytes to the server using remote procedure EDABS. The server converts the hex data back to binary and stores the image ready for INSERT/UPDATE into DB2.
3. Repeats steps 1 and 2 until the complete image is sent to the server in hex format. It then sends remote procedure EDABE to mark the end of the image.

4. The client application prepares an SQL INSERT/UPDATE using parameter markers for the columns of the row.
5. The client application issues a BIND for the columns using CHAR(16) for the image column.
6. The client application issues an EXECUTE USING command giving the data values for the row columns but using 'BLOB ' for the image. The row will be INSERTed/UPDATEd using the image buffer stored on the server.
7. Once the application has finished with the stored image on the server (and COMMITed the data), it should send the EDABK Remote Procedure to release server storage.

For full details and examples of how to maintain DB2 *for bit data* columns, see the *API Reference* and *Connector for ODBC* manuals.

Changing the Precision and Scale of Numeric Columns

How to:

Change the Precision and Scale of Numeric Columns

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Change the Precision and Scale of Numeric Columns

```
ENGINE [DB2] SET CONVERSION RESET
ENGINE [DB2] SET CONVERSION format RESET
ENGINE [DB2] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [DB2] SET CONVERSION format [PRECISION MAX]
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`REAL` which indicates that the command applies only to single precision floating point columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT and REAL data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	**MAX Precision**
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER fields to 7:

```
ENGINE DB2 SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE DB2 SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER fields to the default:

```
ENGINE DB2 SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE DB2 SET CONVERSION RESET
```

Customizing the DB2 Environment

In this section:

The Adapter for DB2 provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Improving Response Time

If you are using the Adapter for DB2 on OS/390 or z/OS, note that DB2 Version 3 supports parallel query I/O and Version 4 supports parallel query CPU to improve response. The adapter supports parallel processing if you issue the SET CURRENT DEGREE command prior to the request.

Syntax: How to Improve Response Time

```
ENGINE [DB2] SET CURRENT DEGREE {'1'|'ANY'}
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`1`

Invokes serial processing. 1 is the default value.

`ANY`

Invokes parallel processing for dynamic requests. If the thread to DB2 is closed during the session, the value resets to 1.

Designating a Default Tablespace

You can use the SET DBSPACE command to designate a default tablespace for tables you create. For the duration of the session, the adapter places these tables in the DB2 tablespace that you identify with the SET DBSPACE command. If the SET DBSPACE command is not used, DB2 uses the default tablespace for the connected user.

Syntax: How to Designate a Default Storage Space for Tables

```
ENGINE [DB2] SET DBSPACE {datasource.tablespace|DATABASE datasource}
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`datasource`

Is the data source name. DSNDB04 is the default value, which is a public data source.

`tablespace`

Is a valid table space name in the data source.

Note: This command will only affect CREATE FILE requests issued by Table Services. It does not affect Passthru CREATE TABLE commands.

Controlling the Types of Locks

You can use the SET ISOLATION command to specify the isolation level of transactions created by the adapter. The isolation level controls the types of locks for objects referenced in the requests executed within the transaction.

Syntax: How to Control the Lock Type

```
ENGINE [DB2] SET ISOLATION level
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`level`

Sets the DB2 isolation level which is mapped to the IBM RDBMS isolation level. If you do not specify an isolation level, the level is reset to the adapter default.

DB2 Isolation Level (CLI)	**IBM RDBMS Isolation Level**
RC (SQL_TXN_READ_COMMITTED)	CS (Cursor Stability)
SE (SQL_TXN_SERIALIZABLE_READ)	RR (Repeatable Read)
RR (SQL_TXN_REPEATABLE_READ)	RS (Read Stability)
RU (SQL_TXN_READ_UNCOMMITTED)	UR (Uncommitted Read)

`RC` releases shared locks as the cursor moves on in the table. Use for read-only requests. RC is the default value. Maps to CS (Cursor Stability).

`SE` locks the retrieved data until it is released by an SQL COMMIT WORK or SQL ROLLBACK WORK statement. Maps to RR (Repeatable Read).

`RR` maps to RS (Read Stability). For more information, see the *DB2 Command and Utility Reference*.

`RU` provides read-only access to records even if they are locked. However, these records may not yet be committed to the data source. Maps to UR (Uncommitted Read).

Note:

- The adapter does not validate the isolation level values. If you issue the SET ISOLATION command with an invalid value, the adapter will not return a message.
- To display the isolation level setting, issue the ENGINE [DB2] ? CONNECTINFO query command.

Overriding Default Parameters for Index Space

You can use the SET IXSPACE command to override the default parameters for the index space implicitly created by the CREATE FILE and HOLD FORMAT commands.

Syntax: How to Set IXSPACE

```
ENGINE [DB2] SET IXSPACE [index-spec]
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

index-spec

Is the portion of the DB2 CREATE INDEX statement that defines the parameters for the index. It can consist of up to 94 bytes of valid Oracle index space parameters. To reset the index space parameters to their default values, issue the SET IXSPACE command with no parameters.

Note: Refer to the DB2 documentation for more information on this command.

The long form of SQL Passthru syntax for commands exceeding one line is:

```
ENGINE DB2
SET IXSPACE index-spec
END
```

For example, to specify the NOSORT, NOLOGGING, and TABLESPACE portions of the DB2 CREATE INDEX statement, enter the following commands:

```
ENGINE DB2
SET IXSPACE NOSORT NOLOGGING
TABLESPACE TEMP
END
```

Note: This command will only affect CREATE INDEX requests issued by CREATE FILE and HOLD FORMAT DB2 commands. It does not affect Passthru CREATE INDEX commands, for example:

```
ENGINE DB2 SET IXSPACE TABLESPACE tablespace_name
TABLE FILE table_name
PRINT *
ON TABLE HOLD AS file_name FORMAT DB2
END
```

Activating NONBLOCK Mode

The Adapter for DB2 has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE [DB2] SET NONBLOCK {0|n}
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Controlling Column Names

You can use the SET NOCOLUMNTITLE command to control the column names in a report when executing a stored procedure.

Syntax: How to Control Column Names

```
ENGINE [DB2] SET NOCOLUMNTITLE {ON|OFF}
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this parameter value if you previously issued the SET SQLENGINE command.

`ON`

Uses generated column names (for example, E01, E02, and so on) instead of the column names returned by DB2.

`OFF`

Uses the column names returned by DB2. OFF is the default value.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [DB2] SET PASSRECS {ON|OFF}
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Using DB2 Alias

You can use the SET REMOTECAT command to take advantage of DB2 aliases. DB2 allows a table alias in one DB2 subsystem to access a physical table in another DB2 subsystem. The physical table can be located in either a local or remote subsystem. You can include the command in any of the supported server profiles.

Syntax: How to Use a DB2 Alias

```
ENGINE DB2 SET REMOTECAT ON
```

Setting End-User Information

How to:

Set End-User Information

Query End-User Information for DB2 Running on OS/390

Query End-User Information for DB2 Running on UNIX or Windows

In the UNIX, Windows, and OS/390 or z/OS environments, you can set new values for end-user information that is passed to DB2 when the next SQL request is processed. This information includes the:

- Client user ID
- Application program name
- Workstation name
- Accounting string

You can then query the information using SELECT statements. On OS/390 or z/OS, you can also view end-user information natively by submitting the following command:

```
-DISPLAY THREAD(*) DETAIL
```

Syntax: How to Set End-User Information

```
ENGINE DB2 SET CLIENT_APPLNAME application_name
ENGINE DB2 SET CLIENT_USERID userid
ENGINE DB2 SET CLIENT_WRKSTNNAME workstation
ENGINE DB2 SET CLIENT_ACCTNG account
```

where:

`application_name`

Is the name of an application program.

`userid`

Is a client's user ID.

`workstation`

Is the name associated with the user workstation.

`account`

Is an accounting string associated with the client user.

Syntax: How to Query End-User Information for DB2 Running on OS/390

Once end-user information is defined, you can query it using the following syntax:

```
ENGINE DB2
SELECT CURRENT CLIENT_APPLNAME,CURRENT CLIENT_USERID,CURRENT
CLIENT_WRKSTNNAME, CURRENT CLIENT_ACCTNG
FROM SYSIBM.SYSDUMMY1;
END
```

Syntax: How to Query End-User Information for DB2 Running on UNIX or Windows

Once end-user information is defined, you can query it using the following syntax:

```
ENGINE DB2
SELECT CLIENT APPLNAME, CLIENT USERID, CLIENT WRKSTNNAME, CLIENT ACCTNG
FROM SYSIBM.SYSDUMMY1;
END
```

Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [DB2] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`DB2`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

`setting`

Is the optimization setting. Valid values are as follows:

`OFF` instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

`ON` instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

`FOCUS` passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

`SQL` passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql DB2 set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql DB2 set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

How to:

Optimize Requests Containing Virtual Fields

Example:

Using IF-THEN_ELSE Optimization Without Aggregation

Using IF-THEN_ELSE Optimization With Aggregation

Using IF-THEN_ELSE Optimization With a Condition That is Always False

Reference:

SQL Limitations on Optimization of DEFINE Expressions

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [DB2] SET OPTIFTHENELSE {ON|OFF}
```

where:

`DB2`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL DB2 SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL DB2 SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL DB2 SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Specifying the Block Size for Retrieval Processing

How to:

Specify Block Size for Array Retrieval

Specify Block Size for Insert Processing

Suppress the Bulk Insert API

Reference:

Bulk Insert API Behavior

The Adapter for DB2 supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Specify Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE [DB2] SET FETCHSIZE n
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Syntax: How to Specify Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE [DB2] SET INSERTSIZE n
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Syntax: How to Suppress the Bulk Insert API

```
ENGINE [DB2] SET FASTLOAD [ON|OFF]
```

where:

`DB2`

Indicates the Adapter for DB2. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Uses the Bulk Insert API. ON is the default.

`OFF`

Suppresses the use of the Bulk Insert API.

Reference: Bulk Insert API Behavior

You can use DataMigrator with the Bulk Insert API for DB2 (Windows and UNIX).

Errors that occur during the load (such as duplication) can cause the batch of rows to be rejected as a whole.

Improving Efficiency With Aggregate Awareness

Aggregate awareness substantially improves the efficiency of queries.

For details about this feature, see Appendix B, *Aggregate Awareness Support*.

Syntax: How to Set Aggregate Awareness

```
SET AGGREGATE_AWARENESS {FRESHONLY|OLD_OK|OFF}
```

where:

`FRESHONLY`

Sets different values for the parameters associated with each RDBMS.

`OLD_OK`

Sets different values for the parameters associated with each RDBMS.

`OFF`

If no option is selected, the behavior of the target RDBMS is determined by the database configuration options.There is no default for this setting.

For details about adapter-specific settings, see *Usage Notes for Aggregate Awareness* on page B-3.

Using DB2 Cube Views

The Adapter for DB2 supports Cube Views as an object type. You can specify Cubes as the object type for which you want to create synonyms.

Before you can use the Adapter for DB2 with Cube Views, you must define DB2 connections for the database(s) containing the DB2 cube views.

For details, see *How to Create a Synonym From the Web Console* on page 8-13 or *How to Create a Synonym Manually* on page 8-16.

Mapping Metadata for DB2 Cubes Views

Synonyms are generated for Cubes (not Cube Models). These synonyms reflect the structure of the cube and its attributes and measures, rather than the structure of the underlying DB2 tables and their columns.

The Master File synonym component has several segments:

- The root segment corresponds to the Cube Facts.
- The children segments correspond to the Cube Hierarchies.

The fields in the root segment reflect the Cube Measures. Field names and titles are generated based on measure business names, while aliases are generated based on measure names.

- The PROPERTY attribute for these fields contains the word Measure (or Calculated Measure for calculated measures).
- The REFERENCE attribute contains the aggregated expression for measures that are based on simple columns (not expressions) and have aggregations that match those available in the DML.

Example: Sample DB2 Cube View Master File

```
FILENAME=BASEAPP/COMPLEX_MEASURES__C_, SUFFIX=DB2CV   , $
  SEGMENT=SALESCF, SEGTYPE=S0, $
    FIELDNAME=4ADDS, ALIAS=4adds, USAGE=I11, ACTUAL=I4, MISSING=ON,
      TITLE='4adds',
      REFERENCE=CNT.4ADDS, PROPERTY=MEASURE,  $
    FIELDNAME=POWEROFADDS, ALIAS=powerofadds, USAGE=D21.2, ACTUAL=D8,
MISSING=ON,
      TITLE='powerofadds',
      PROPERTY=MEASURE,  $
    FIELDNAME=PROFIT_CONST, ALIAS=profit*const, USAGE=D21.2, ACTUAL=D8,
MISSING=ON,
      TITLE='profit*const',
      PROPERTY=MEASURE,  $
    FIELDNAME=RANDOM, ALIAS=random, USAGE=I11, ACTUAL=I4, MISSING=ON,
      TITLE='random',
      REFERENCE=CNT.RANDOM, PROPERTY=MEASURE,  $
    FIELDNAME=3_COLS_WITH_MIXTUREOF_FUNC, ALIAS='3 cols with mixtureof
FUNC', USAGE=D21.2, ACTUAL=D8, MISSING=ON,
      TITLE='3 cols with mixtureof FUNC',
      PROPERTY=MEASURE,  $
    FIELDNAME=ROUND_OF_ANOTHER_MEAS, ALIAS='round of another meas',
USAGE=D21.2, ACTUAL=D8, MISSING=ON,
      TITLE='round of another meas',
      PROPERTY=MEASURE,  $
  SEGMENT=PRODUCT__CH_, SEGTYPE=U, PARENT=SALESCF, $
    FIELDNAME=PRODUCT_GROUP_ID, ALIAS=product_group_ID, USAGE=I11,
ACTUAL=I4, MISSING=ON,
      TITLE='product_group_ID',
      WITHIN='*product (CH)', $
    FIELDNAME=PRODUCT_LINE_ID, ALIAS=product_line_ID, USAGE=I11,
ACTUAL=I4, MISSING=ON,
      TITLE='product_line_ID',
      WITHIN=PRODUCT_GROUP_ID, $
    FIELDNAME=PRODUCT_LINE_NAME, ALIAS='12  product_line_ID
product_line_name', USAGE=A25V, ACTUAL=A25V, MISSING=ON,
      TITLE='product_line_name',
      REFERENCE=PRODUCT_LINE_ID, PROPERTY=CAPTION,  $
  SEGMENT=STORE__CH_, SEGTYPE=U, PARENT=SALESCF, $
    FIELDNAME=STORE_LOCATION_ID, ALIAS='**** store_location_ID_1',
USAGE=I11, ACTUAL=I4, MISSING=ON
      TITLE='store_location_ID',
      WITHIN='*store (CH)', $
    FIELDNAME=STORE_ADDRESS, ALIAS='store_location_ID_1
_store_address', USAGE=A25V, ACTUAL=A25V, MISSING=ON,
      TITLE='store address',
      REFERENCE=STORE_LOCATION_ID, PROPERTY=UDA,  $
    FIELDNAME=STORE_CITY, ALIAS='store_location_ID_1   _store_city',
USAGE=A45, ACTUAL=A45, MISSING=ON,
      TITLE='store_city',
      REFERENCE=STORE_LOCATION_ID, PROPERTY=UDA,  $
    FIELDNAME=STORE_ID, ALIAS=store_ID, USAGE=I11, ACTUAL=I4, MISSING=ON,
TITLE='store_ID',
      WITHIN=STORE_LOCATION_ID, $
```

Calling a DB2 Stored Procedure Using SQL Passthru

How to:

Invoke a Stored Procedure

Example:

Invoking a Stored Procedure

Sample Stored Procedure

DB2 stored procedures are supported using SQL Passthru. These procedures need to be developed within DB2 using the CREATE PROCEDURE command.

The adapter supports stored procedures with input, output, and inout parameters.

The output parameter values that are returned by stored procedures are available as result sets. These values form a single-row result set that is transferred to the client after all other result sets returned by the invoked stored procedure. The names of the output parameters (if available) become the column titles of that result set.

Note that only the output parameters (and the returned value) referenced in the invocation string are returned to the client. As a result, users have full control over which output parameters have their values displayed.

The server supports invocation of stored procedures written according to rules that are specific to each adapter.

Syntax: How to Invoke a Stored Procedure

```
SQL [DB2]EX procname [parameter_specification1]
[,parameter_specification2]...
END
```

where:

`DB2`

Is the ENGINE suffix.

`procname`

Is the name of the stored procedure. It is the fully or partially qualified name of the stored procedure in the native RDBMS syntax.

`parameter_specification`

Input, output, and in-out parameters are supported. Use the variation required by the stored procedure:

`input`

Is a literal (for example, 125, 3.14, 'abcde'). You can use reserved words as input. Unlike character literals, reserved words are not enclosed in quotation marks (for example, NULL). Input is required.

`output`

Is represented as a question mark (?). You can control whether output is passed to an application by including or omitting this parameter. If omitted, this entry will be an empty string (containing 0 characters).

`in-out`

Consists of a question mark (?) for output and a literal for input, separated by a slash: /. (For example: ?/125, ?/3.14, ?/'abcde'.) The out value can be an empty string (containing 0 characters).

Example: Invoking a Stored Procedure

In this example, a user invokes a stored procedure, edaqa.test_proc01, supplies input values for parameters 1, 3, 5 and 7, and requests the returned value of the stored procedure, as well as output values for parameters 2 and 3.

Note that parameters 4 and 6 are omitted; the stored procedure will use their default values, as specified at the time of its creation.

```
SQL DB2 EX edaqa.test_proc01 125,?,?/3.14,,'abc',,'xyz'
END
```

Example: Sample Stored Procedure

This stored procedure uses out and inout parameters:

```
CREATE PROCEDURE EDAQA.PROCP3 (   OUT chSQLSTATE_OUT   CHAR(5),
                                  OUT intSQLCODE_OUT   INT,
                                  INOUT l_name char(20),
                                  INOUT f_name char(20))
    RESULT SETS 1
    LANGUAGE SQL
------------------------------------------------------------------------
-- SQL Stored Procedure
------------------------------------------------------------------------
P1: BEGIN

    -- Declare variable
    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
    DECLARE SQLCODE INT DEFAULT 0;

    -- Declare cursor
    DECLARE cursor1 CURSOR WITH RETURN FOR

        SELECT
           EDAQA.NF29005.SSN5 AS SSN5,
           EDAQA.NF29005.LAST_NAME5 AS LAST_NAME5,
           EDAQA.NF29005.FIRST_NAME5 AS FIRST_NAME5,
           EDAQA.NF29005.BIRTHDATE5 AS BIRTHDATE5,
           EDAQA.NF29005.SEX5 AS SEX5
        FROM
           EDAQA.NF29005
        WHERE
           (
             ( EDAQA.NF29005.LAST_NAME5 =  l_name )
        AND
             ( EDAQA.NF29005.FIRST_NAME5 = f_name )
           );

     -- Cursor left open for client application
    OPEN cursor1;

    SET chSQLSTATE_OUT = SQLSTATE;
    SET intSQLCODE_OUT = SQLCODE;
    SET l_name = 'this is first name';
    SET f_name = 'this is last name';

END P1   @
```

CHAPTER 9

Using the Adapter for Enterprise Java Beans

Topics:

- Preparing the Web Application Server Environment
- Configuring the Adapter for Enterprise Java Beans
- Managing Enterprise Java Beans Metadata

The Adapter for Enterprise Java Beans (EJB) allows applications to access Enterprise Java Beans data sources. The adapter converts application requests into native Enterprise Java Beans statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

The Adapter for Enterprise Java Beans supports access to Entity and Session Enterprise Java Beans deployed under BEA WebLogic and IBM WebSphere Application Servers.

Preparing the Web Application Server Environment

The Web Application Server environment is set up during the installation of the corresponding Web Application Server or Client software.

Note: Enterprise Java Beans must be deployed on the Web Application Server.

Using the standard rules for deploying the Web Application Server or Client software, the server supports connections to:

- BEA WebLogic Application Server
- IBM WebSphere Application Server

Java must exist in your environment in order for you to access Weblogic EJBs. Before you configure the Adapter for Enterprise Java Beans, you must indicate where Java resides on your machine. This involves specifying your Java home directory (JDK_HOME) and the directory that contains the Weblogic.jar file. The weblogic.jar file must be added to the CLASSPATH to provide access to the Weblogic.ejbc file and certain J2EE class files contained in this jar file.

Syntax: How to Define Your Java Home Directory JDK_HOME

```
JDK_HOME = path
export JDK _HOME
```

where:

path

Is the Java home directory.

Syntax: How to Add the weblogic.jar File to Your Path

```
CLASSPATH=$CLASSPATH:path_to_Weblogic.jar
export CLASSPATH
```

where:

`path`

Is the directory in which the weblogic.jar file resides.

Configuring the Adapter for Enterprise Java Beans

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

There are two methods by which a user can be authenticated when connecting to an Enterprise Java Beans database server:

- **Explicit.** User IDs and passwords are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.
- **Password Passthru.** User IDs and passwords received from the client application are passed to the Web Application Server for authentication.

When a client connects to the iWay Server, the user ID and password are passed to the Web Application Server for authentication, and are not authenticated by the iWay Server. To implement this type of authentication, start the iWay Server with security turned off. The server allows the client connection, then stores an encrypted form of the client's connection message. The encrypted message can be used for connection to a Web Application Server at any time during the duration of the server session.

Declaring Connection Attributes for the Web Application Server

How to:

Declare Connection Attributes for the Web Application Server From the Web Console

Declare Connection Attributes for the Web Application Server Manually

Example:

Declaring Connection Attributes for the BEA WebLogic Application Server

Declaring Connection Attributes for the IBM WebSphere Application Server

The SET CONNECTION_ATTRIBUTES command allows you to declare a connection to one Web Application Server, and to supply the attributes necessary for connecting to the server.

You can connect to more than one Web Application Server by issuing multiple SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query referencing that connection is issued (see *Selecting a Web Application Server to Access* on page 9-7). You can include SET CONNECTION_ATTRIBUTES commands in an RPC or a server profile. The profile can be encrypted.

If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The first SET CONNECTION_ATTRIBUTES command sets the default Web Application Server database server to be used.
- If more than one SET CONNECTION_ATTRIBUTES command declares the same Web Application Server, the connection information is taken from the last SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes for the Web Application Server From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *Procedures* group folder, then expand the *EJB* adapter folder and click a connection. The Add EJB Weblogic or Websphere to Configuration pane opens.
3. Enter the following parameters:

Field	Description
Connection name	Arbitrary connection name to be used to reference the connection.
URL	URL of the machine where the Web Application Server is running.

Field	**Description**
Security	There are two methods by which a user can be authenticated when connecting to a Enterprise Java Beans database server: **Explicit.** The user ID and password are explicitly specified for each connection and passed to Enterprise Java Beans, at connection time, for authentication. **Password Passthru.** The user ID and password received from the client application are passed to Enterprise Java Beans, at connection time, for authentication. This option requires that the server be started with security off.
User	User name for the Web Application Server authenticated login.
Password	Password that identifies the entered user name.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Tip: Once the adapter is configured, the connection name appears in the navigation pane under *Configured*. If you left-click either the adapter name or the connection name, an appropriate set of options becomes available.

Syntax: How to Declare Connection Attributes for the Web Application Server Manually

For BEA WebLogic Application Server with Explicit authentication:

```
ENGINE [EJB] SET CONNECTION_ATTRIBUTES connection_name
't3://hostname:port'/userid,password
```

For BEA WebLogic Application Server with Password Passthru authentication:

```
ENGINE [EJB] SET CONNECTION_ATTRIBUTES connection_name
't3://hostname:port'
```

For IBM WebSphere Application Server with Explicit authentication:

```
ENGINE [EJB] SET CONNECTION_ATTRIBUTES connection_name
'iiop://hostname:port'/userid,password
```

For IBM WebSphere Application Server with Password Passthru authentication:

```
ENGINE [EJB] SET CONNECTION_ATTRIBUTES connection_name
'iiop://hostname:port'
```

where:

`EJB`

Indicates the Adapter for Enterprise Java Beans.

`connection_name`

Is any name used as a connect descriptor to a Web Application Server across the network.

`hostname`

Is the host name of the computer where the Web Application Server is running.

`port`

Is the TCP/IP port number on which the Web Application Server is listening for incoming connections.

`userid`

Is the authorization ID known to the Web Application Server.

`password`

Is the password associated with the authorization ID.

Example: Declaring Connection Attributes for the BEA WebLogic Application Server

The following SET CONNECTION_ATTRIBUTES command uses Password Passthru authentication to connect to BEA1, the BEA WebLogic Application Server running on the machine with hostname UNXSOL28:

```
ENGINE EJB SET CONNECTION_ATTRIBUTES BEA1 't3://UNXSOL28:7001'
```

The following SET CONNECTION_ATTRIBUTES command uses Explicit authentication to connect to MYBEA, a local BEA WebLogic Application Server:

```
ENGINE EJB SET CONNECTION_ATTRIBUTES MYBEA
't3://localhost:7001'/USERA,PWDA
```

Example: Declaring Connection Attributes for the IBM WebSphere Application Server

The following SET CONNECTION_ATTRIBUTES command uses Password Passthru authentication to connect to IBMCON1, the IBM WebSphere Application Server running on the machine with hostname AIX43:

```
ENGINE EJB SET CONNECTION_ATTRIBUTES IBMCON1 'iiop://AIX43:900'
```

The following SET CONNECTION_ATTRIBUTES command uses Explicit authentication to connect to MYWSP, a local IBM WebSphere Application Server:

```
ENGINE EJB SET CONNECTION_ATTRIBUTES MYWSP
'iiop://localhost:900'/USERB,PWDB
```

Selecting a Web Application Server to Access

After you have used the SET CONNECTION_ATTRIBUTES command to declare which Web Application Servers will be accessed, select a specific Web Application Server from the list of declared servers. You can do this in one of two ways:

- Include the CONNECTION= attribute in the Access File of the table specified in the current EJB query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION= attribute is automatically included in the Access File. This attribute supersedes the default connection.
- Do not include the CONNECTION= attribute in the Access File. The connection_name value specified in the *first* SET CONNECTION_ATTRIBUTES command is used.

Controlling Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections each of the connections you want to establish.

A connection occurs at the first interaction with the declared Web Application Server.

Syntax: How to Control the Connection Scope

```
ENGINE EJB SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the Web Application Server.

Managing Enterprise Java Beans Metadata

In this section:

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Enterprise Java Beans data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Java Bean (EJB) table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File, which represent the server's metadata.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter. See *Configuring the Adapter for Enterprise Java Beans* on page 9-3 for more information.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter (Only one connection can be selected at any given time.) The Select Synonym Candidates for EJB WebSphere or Weblogic (Step 1 of 2) pane opens.

4. Click the *Filter by Full or Partial JNDI Name* check box to filter the collections for which you wish to create synonyms. To create synonyms for all collections leave this box blank.

5. Click *Select JNDI*. The Create Synonym (Step 2 of 2) pane opens.

6. Enter the following parameters, as appropriate:

Parameter	**Description**
Select Application	Select an application directory. baseapp is the default value.
Prefix/Suffix	If you have EJBs with identical names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll EJBs, assign the prefix HR to distinguish the synonyms for the human resources EJBs. Note that the resulting synonym name cannot exceed 64 characters. If all EJBs have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

7. Select the EJBs for which you wish to create synonyms:
   - To select all EJBs in the list, select the check box to the left of the *Default Synonym Name* column heading.
   - To select specific EJBs, select the corresponding check boxes.
8. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.
9. Click *Create Synonym*.

   Synonyms are created and added under the specified application directory.

   A status window displays the message:

   ```
   All Synonyms Created Successfully
   ```
10. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR jndi_name DBMS EJB AT connection_name
END
```

where:

appname

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the EJB JNDI name (maximum 64 characters for Windows and UNIX Server platforms).

`jndi_name`

Is the JNDI name for the EJB.

`EJB`

Indicates the Adapter for Enterprise Java Beans.

`AT` *`connection_name`*

Is the connection name as previously specified in a SET CONNECTION_ATTRIBUTES command. When the server creates the synonym, this connection name becomes the value for CONNECTION= in the Access File.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.
- CREATE SYNONYM creates a Master File and Access File which represent the server metadata.

Example: Using CREATE SYNONYM

Use the following syntax to create a synonym for the variable “Cities.”

```
CREATE SYNONYM CITIES FOR CitiesEJB DBMS EJB AT BEACON1
END
```

CitiesEJB interfaces:

```
public interface CitiesEJBRemote extends javax.ejb.EJBObject {
     public String getData() throws RemoteException;
     public int getIndex() throws RemoteException;
     public void setData(int num) throws RemoteException;
     public void setIndex(int ind) throws RemoteException;
 }

public interface CitiesEJBHomeRemote extends javax.ejb.EJBHome {
       public CitiesEJBRemote create()
         throws RemoteException,CreateException;
       public CitiesEJBRemote findByPrimaryKey(Integer primaryKey)
         throws FinderException, RemoteException;
}
```

Generated Master File cities.mas

```
FILENAME=CITIESG, SUFFIX=EJB     , $
$ ==========================================================================
$ MFD generated for JNDI:CitiesEJB
$==========================================================================
  SEGMENT=PRIMARY_KEY, SEGTYPE=S0, $
$==========================================================================
$ constructors and parameters for PRIMARY_KEY:Integer
$==========================================================================
   GROUP=, ALIAS=K1, USAGE=A4, ACTUAL=A4, $
    FIELDNAME=K1F1, ALIAS=c0p0, USAGE=I11, ACTUAL=A4, $
  SEGMENT=OBJ01_CITIESEJBREMOTE, SEGTYPE=S0, PARENT=PRIMARY_KEY, $
$==========================================================================
$ OBJECT01, CitiesEJBRemote
$==========================================================================
    FIELDNAME=DATA, ALIAS=Data, USAGE=A256V, ACTUAL=A256V, MISSING=ON, $
    FIELDNAME=INDEX, ALIAS=Index, USAGE=I11, ACTUAL=A4, MISSING=ON, $
```

Generated Access File cities.acx

```
JNDINAME=CitiesEJB,
CONNECTION=MYCON,$
SEGNAME=PRIMARY_KEY,OBJECTNAME=Integer,$
SEGNAME=OBJ01_CitiesEJBRemote,OBJECTNAME=CitiesEJBRemote,$
```

The Master File root segment describes EJB Primary Keys. Each Primary Key is represented by a GROUP. Each KEY component is represented by an elementary FIELD in the GROUP. Only one active Primary Key is allowed in the request.

All other segments in the Master File describe the hierarchy of the OBJECTS that can be returned by EJB. Each GET Method of the OBJECT is represented by an elementary FIELD in the segment.

Reference: Access File Keywords

Keyword	Description
`JNDINAME`	EJB JNDI name.
`CONNECTION=`*`connection_name`*	Indicates access to the specified Web Application Server.
Absence of `CONNECTION=`	Indicates access to the default Web Application Server.

Data Type Support

The following chart provides information about the default mapping of Enterprise Java Beans data types to server data types:

EJB Data Type	**Data Type**	
	ACTUAL	**USAGE**
VOID	A1	A1
BYTE	A4	I4
SHORT	A4	I6
INT	A4	I11
LONG	A20	P20
FLOAT	A12	F12.2
DOUBLE	A20	D20.2
CHAR	A1	A1
STRING	A256V	A256V
BOOLEAN	A5	A5

CHAPTER 10

Using the Adapter for Essbase

Topics:

The Adapter for Essbase allows applications to access Essbase data sources. The adapter converts application requests into native Essbase statements and returns optimized answer sets to the requesting application.

The Adapter for Essbase provides read-only access to Hyperion Cubes from a server running on Microsoft Windows. Essbase can only be accessed using an API supplied by the vendor. It is not possible to gain access using Direct SQL (Direct Passthru in server terminology) or the server's Adapter for ODBC.

Preparing the Essbase Environment

The Adapter for Essbase minimally requires the installation of the Essbase Client. The Essbase Client allows you to connect to a local or remote Essbase Database Server.

Procedure: How to Prepare the Environment on Microsoft Windows

On Microsoft Windows, the Essbase environment is set up during the installation of Essbase.

Procedure: How to Prepare the Environment on UNIX

Using the UNIX environment variable $ARBORPATH, specify the location of the Essbase Server you wish to access.

Example: Specifying ARBORPATH on UNIX

To set the home directory for the Essbase Client to /usr/essbase, specify:

```
ARBORPATH=/usr/essbase
export ARBORPATH
```

Configuring the Adapter for Essbase

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

When connecting to an Essbase Server, you must use Explicit authentication. This method requires that user IDs and passwords be explicitly stated in the SET CONNECTION_ATTRIBUTES command. You can include these commands in the server global profile, edasprof.prf, for all users.

When you access an Essbase OLAP Server using the iWay Server, you are identified by a user ID and password. The Database Administrator can assign management privileges to a specific user, where privileges:

- Are constant for all applications and databases.
- Apply to an application.
- Apply to a database.

For more information on security, see the *Hyperion Essbase* documentation.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, select *Data Adapters*.
2. Expand the *Add* folder, expand the *OLAP* group folder, then expand the *Essbase* folder and click a connection. The Add Essbase to Configuration pane opens.
3. Supply values for the following parameters:

Field	Description
Server	Machine name where Essbase is running.
User	User name by which you are known to Essbase.
Password	Password associated with the user name.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

   Tip: Once the adapter is configured, the connection name appears in the navigation pane under *Configured*. If you left-click either the adapter name or the connection name, an appropriate set of options becomes available.

Syntax: How to Declare Connection Attributes Manually

For Explicit authentication,

```
ENGINE [ESSBASE] SET CONNECTION_ATTRIBUTES [connection_name]/userid,password
```

where:

`ESSBASE`

Indicates the Adapter for Essbase. You can omit this value if you previously issued the SET SQLENGINE command.

`connection_name`

Is the server name used as a connect descriptor to the Essbase Server across the network.

`userid`

Is the primary authorization ID by which you are known to Essbase.

`password`

Is the password associated with the primary authorization ID.

Managing Essbase Metadata

In this section:

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Essbase data type.

Creating Synonyms

How to:

Create a Synonym From the Web Console

Create a Synonym Manually

Use Default Field Names

Example:

Using CREATE SYNONYM (Master File)

Using CREATE SYNONYM (Access File)

Reference:

Managing Synonyms

CHECKNAMES for Special Characters and Reserved Words

Access File Keywords

Synonyms define unique names (or aliases) for each Essbase *application.database* combination that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

For details on how to create a synonym through the Web Console, see the *Server Configuration and Operation* manual for your platform.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The Select Database from the Application for ESSBASE pane (Step 1 of 3) opens.

4. Select a database(s) from the list below the *Select Dimension* button.

5. If you wish to filter your selection(s), choose any of the following options:

Option	**Description**
Filter Dimensions	This check box enables you to retrieve a filtered list of Essbase Dimensions from which to choose a Scenario Dimension and/or a Measure Dimension. A *Name* input box is added to the selection pane. Type a string for filtering the dimension names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, type ABC% to select dimensions whose names begin with the letters ABC; %ABC to select dimensions whose names end with the letters ABC; %ABC% to select dimensions whose names contain the letters ABC at the beginning, middle, or end.
Measure	This option enables you to limit the number of fields in the Master File. • Choose *Yes* (the default) to change the Accounts tagged dimension without having to change the outline in the Essbase Database Server. With this setting, an actual field name is generated for every member of the chosen dimension in the Master File rather than producing the dimension in terms of generations. **Note:** If you select Yes, when you click *Select Dimension* in step 6, two drop-down lists will be added to the pane—*Select Measures* and *Select Scenarios*. • Choose *No* to limit the number of fields in the Master File. With this setting, all of the dimensions are interpreted as non-Accounts dimensions in the Master File and represented as generations. For related information, see *Limiting the Number of Fields in a Master File* on page 10-23. **Note:** If you select No, when you click *Select Dimension* in step 6, only the *Select Scenarios* drop-down list appears in the pane. You cannot apply additional selection criteria for Measures. For additional information and syntax, see *Describing the Measures Dimension in the Master File* on page 10-22.

Option	**Description**
Aggregate	This option enables you to control summing on non-aggregated fields in an Essbase request. • Choose *On* to enable summing on non-aggregated fields. • Choose *Off* to disable summing on non-aggregated fields. For additional information and syntax, see *Preventing Aggregation of Non-Consolidating Members* on page 10-33.

6. Click *Select Dimension*. The Choose Measures and Scenarios (Step 2 of 3) pane opens. Any selection criteria you specified in in the previous pane are reflected at the top of this pane. In addition, the dimensions that meet the specified criteria are listed in the Select Scenarios, and, if shown, the Select Measures, drop-down lists, which appear at the bottom of the pane.

7. If you wish to create a synonym based on a *normal dimension* (that is, one in which field descriptions are generated for the accounts dimension segment), simply click *Create Synonym*.

Tip: If you wish to exercise control over several synonym field name processing options, go to step 10 in this procedure *before* you click *Create Synonym*.

8. If you wish to create a synonym based on a *pseudo accounts dimension* (that is, one in which a scenario dimension is generated in place of the accounts dimension segment), click the *Choose Scenario* check box, and then:

a. Optionally, from the Select Scenario drop down list, specify a dimension from which you can later choose a member. (If you choose *None* (the default), no pseudo accounts dimension is generated.)

If you do not want to use all members of the chosen dimension, click the *Filter* check box and, in the *Name* input box, type a string for filtering the scenario names, inserting the wildcard character (%) as needed at the beginning and/or end of the string.

b. Click *Get Scenario*. The Select Scenario (Step 3 of 3) pane opens.

c. From the list of Scenario Names at the bottom of the pane, choose the member(s) you would like to use to build the scenario dimension in the Master File.

To select *all* members in the list, click the check box to the left of the *Scenario Name* column heading. (A message indicates that the selection of *all* members in the list may take some time. Click *OK* if you wish to proceed.)

To select *specific* members, click the check boxes to the left of those names.

   d. Click *Create Synonym*.

      Tip: If you wish to exercise control over several synonym field name processing options, go to step 10 in this procedure *before* you click *Create Synonym*.

9. If you selected Measures/Yes (the default) in step 5, and you wish to generate a field name in the Master File for every member of a chosen Measures dimension:

   a. Select a measure from the Select Measures drop-down list.

      Note that *None* (the default) generates the field name for every member in the default Accounts tagged dimension in the Essbase outline.

   b. Click *Create Synonym*.

   Tip: If you wish to exercise control over several synonym field name processing options, go to step 10 in this procedure *before* you click *Create Synonym*.

10. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

    When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

11. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

12. If appropriate, choose or enter the following synonym processing parameters:

Parameter	**Description**
Synonym Name	Type the name of the synonym.
Select Application	Select an application directory. baseapp is the default value.

Parameter	Description
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

13. At whichever point you click *Create Synonym*, synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.

Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '|', '&', ' (', ') ', ' <', '> ', '' ', ' =', ''''

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR application.database DBMS ESSBASE
  AT connection_name
  [CHECKNAMES][UNIQUENAMES]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for one *application.database* combination.

application

Is the Essbase application that contains the database (Cube).

database

Is the database name (Cube) within the application.

ESSBASE

Indicates the Adapter for Essbase.

AT *connection_name*

Is the *connection_name* as previously specified in a SET CONNECTION_ATTRIBUTES command. When the server creates the synonym, this connection_name becomes the value for CONNECTION= in the Access File.

CHECKNAMES

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

UNIQUENAMES

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.
- The Master File that is produced contains a segment for each dimension in the Essbase outline. For each generation within a dimension, a field description is produced. The WITHIN keyword is used to describe the hierarchy inherent in each dimension structure.

Example: Using CREATE SYNONYM (Master File)

Use the following syntax to create a synonym for the application.database Sample.Basic:

```
CREATE SYNONYM sample FOR Sample.Basic DBMS ESSBASE AT UNXSOL26
END
```

Generated Master File sample.mas

```
FILENAME=SAMPLE, SUFFIX=ESSBASE, $
SEGMENT=BASIC, SEGTYPE=S0, $

$ DIMENSION: Scenario

FIELD=SCENARIO,
WITHIN=*Scenario,
ALIAS='Gen1,Scenario',
USAGE=A8,ACTUAL=A8, $

FIELD=GEN2_SCENARIO,
WITHIN=SCENARIO,
ALIAS='Gen2,Scenario',
USAGE=A10,ACTUAL=A10, $
```

Syntax: How to Use Default Field Names

The default field names that are used are taken from the generation names, if available, in the outline; otherwise, they take the format

```
FIELDNAME=generationnumber_dimensionname
```

where:

generationnumber

Is the generation number within the dimension (for example, GEN2).

dimensionname

Is the name of the dimension.

For example, if the outline has a SCENARIO dimension, the first field name would be FIELDNAME=Scenario to represent the highest generation, the second field name would be FIELDNAME=GEN2_SCENARIO, and so on down the dimension hierarchy.

Example: Using CREATE SYNONYM (Access File)

Generated Access File sample.acx

```
SEGNAME=BASIC, SERVER=unxsol26, DBNAME=Basic, APPLNAME=Sample, $
```

Reference: Access File Keywords

Keyword	Description
`SERVER`	Essbase Server name.
`DBNAME=`*`database_name`*	Indicates access to the specified database within an application.
`APPLNAME=`*`application_name`*	Indicates access to the specified application, which can contain one or more databases.

Parent/Child Support

Example:

Using the Parent/Child View in the Master File

The parent/child view in the Master File enables you to make requests using a member without having to know the generation to which the member belongs. The fields that provide this functionality are:

`DIMENSION_MEMBER`

Is the declaration for the field that contains the dimension members. It represents members of product at all levels of the dimension.

`DIMENSION_CAPTION`

Is the declaration for the field that contains the label displayed on reports for DIMENSION_MEMBER (this is the Essbase alias).

`DIMENSION_PARENT`

Is the declaration for the field that contains the parent of DIMENSION_MEMBER.

`DIMENSION_PARENTCAP`

Is the declaration for the field that contains the label displayed on reports for DIMENSION_PARENT (this is the Essbase alias).

The following is a sample of the parent/child view in the Master File:

```
FIELD=SCENARIO_MEMBER
WITHIN=*Scenario2
ALIAS=Scenario
USAGE=A10, ACTUAL=A10, $
FIELD=SCENARIO_CAPTION,
PROPERTY=CAPTION, REFERENCE=SCENARIO_MEMBER,
USAGE=A10,ACTUAL=A10, $
FIELD=SCENARIO_PARENT,
PROPERTY=PARENT_OF, REFERENCE=SCENARIO_MEMBER,
USAGE=A10,ACTUAL=A10, $
FIELD=SCENARIO_PARENTCAP,
PROPERTY=CAP_PARENT, REFERENCE=SCENARIO_MEMBER,
USAGE=A10,ACTUAL=A10, $
```

Example: Using the Parent/Child View in the Master File

In the first request, Actual is explicitly identified with the generation GEN2_SCENARIO:

```
TABLE FILE SAMPLE
PRINT PROFIT
WHERE GEN2_SCENARIO EQ Actual
END
```

In the second request, which takes advantage of a parent/child view, Actual is represented as a member of the Scenario dimension; it is not necessary to know which generation it falls under.

```
TABLE FILE SAMPLE
PRINT PROFIT
WHERE SCENARIO_MEMBER EQ Actual
END
```

Both requests yield the same output:

```
PROFIT = 105,522.00
```

Support for User-Defined Attributes

A User-Defined Attribute (UDA) in Essbase enables you to select and report on data based on a common characteristic. You can include User-Defined Attributes in report requests. For each dimension with UDAs in an Essbase outline, CREATE SYNONYM generates a UDA field name under a segment called UDA in the Master File.

Example: Reporting From a UDA Segment

The Sample Master File contains the following UDA segment:

```
$ DIMENSION: UDA
SEGMENT=UDA, SEGTYPE=U, PARENT=BASIC,$
FIELD=MARKET_UDA,
ALIAS=Market,
USAGE=A13,ACTUAL=A13, $
```

When referencing UDAs in a request, you must also reference a member of the dimension that contains the UDA. In this example, STATE is a member of the MARKET Dimension in the Master File.

```
TABLE FILE SAMPLE
PRINT SALES BY STATE
WHERE MARKET_UDA EQ 'New Market'
END
```

The output displays the values only for the states that have the UDA 'New Market' as a common characteristic:

```
STATE           SALES
---------   ---------
Colorado    38,240.00
Louisiana   11,316.00
Nevada      29,156.00
```

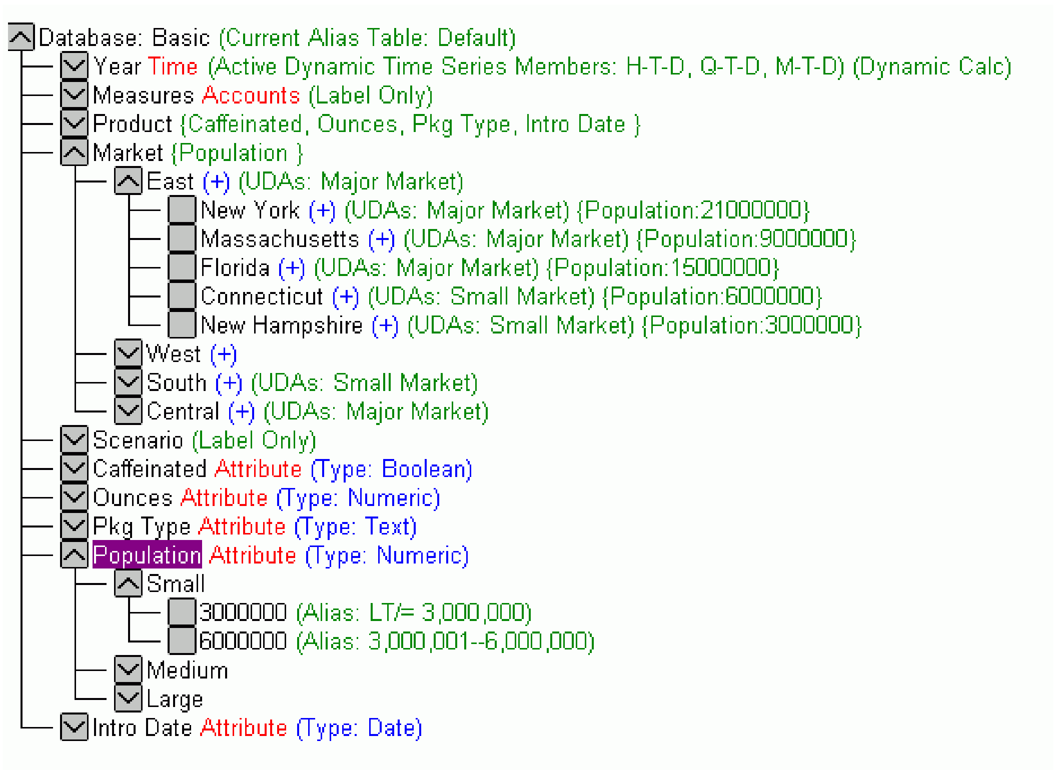
Describing Attribute Dimensions in the Master File

Example:

Generating Attribute Dimensions With the CREATE Synonym Command

Reporting Against Attribute Tagged Dimensions

An attribute dimension is identified by the word Attribute, which appears next to the Dimension name in the Essbase outline. Attribute dimensions are usually associated with standard Essbase dimensions. For example, in the following outline, Population is an Attribute dimension associated with the standard dimension Market.



```
Database: Basic (Current Alias Table: Default)
  Year Time (Active Dynamic Time Series Members: H-T-D, Q-T-D, M-T-D) (Dynamic Calc)
  Measures Accounts (Label Only)
  Product {Caffeinated, Ounces, Pkg Type, Intro Date }
  Market {Population }
    East (+) (UDAs: Major Market)
      New York (+) (UDAs: Major Market) {Population:21000000}
      Massachusetts (+) (UDAs: Major Market) {Population:9000000}
      Florida (+) (UDAs: Major Market) {Population:15000000}
      Connecticut (+) (UDAs: Small Market) {Population:6000000}
      New Hampshire (+) (UDAs: Small Market) {Population:3000000}
    West (+)
    South (+) (UDAs: Small Market)
    Central (+) (UDAs: Major Market)
  Scenario (Label Only)
  Caffeinated Attribute (Type: Boolean)
  Ounces Attribute (Type: Numeric)
  Pkg Type Attribute (Type: Text)
  Population Attribute (Type: Numeric)
    Small
      3000000 (Alias: LT/= 3,000,000)
      6000000 (Alias: 3,000,001--6,000,000)
    Medium
    Large
  Intro Date Attribute (Type: Date)
```

The standard dimension serves as the base dimension for the associated attribute dimensions.

Example: Generating Attribute Dimensions With the CREATE Synonym Command

The CREATE SYNONYM command generates both a Master File that contains an ATTR (attribute) segment, which defines attribute tagged dimensions, and an Access File that contains the actual members of the attribute tagged dimensions.

The following is a portion of the Master File SAMPLE. Notice that Market is the base dimension with which the attribute dimension Population is associated.

```
FILENAME=SAMPLE, SUFFIX=ESSBASE ,$
  SEGMENT=BASIC, SEGTYPE=S0, $
>.
>.
>.
$  DIMENSION: Market
    FIELDNAME=MARKET, ALIAS='Gen1,Market', USAGE=A6, ACTUAL=A6,
      WITHIN='*Market', $
    FIELDNAME=REGION, ALIAS=Region, USAGE=A7, ACTUAL=A7,
      WITHIN='MARKET', $
    FIELDNAME=STATE, ALIAS=State, USAGE=A13, ACTUAL=A13,
      WITHIN='REGION', $
    FIELDNAME=MARKET_MEMBER, ALIAS=Market, USAGE=A13, ACTUAL=A13,
      WITHIN='*Market2', $
    FIELDNAME=MARKET_CAPTION, USAGE=A13, ACTUAL=A13,
      REFERENCE=MARKET_MEMBER, PROPERTY=CAPTION,  $
    FIELDNAME=MARKET_PARENT, USAGE=A13, ACTUAL=A13,
      REFERENCE=MARKET_MEMBER, PROPERTY=PARENT_OF,  $
    FIELDNAME=MARKET_PARENTCAP, USAGE=A13, ACTUAL=A13,
      REFERENCE=MARKET_MEMBER, PROPERTY=CAP_PARENT,  $
>.
>.
>.
$  DIMENSION: ATTR
  SEGMENT=ATTR, SEGTYPE=U, PARENT=BASIC, $
    FIELDNAME=CAFFEINATED, ALIAS=Product, USAGE=A17, ACTUAL=A17, $
    FIELDNAME=OUNCES, ALIAS=Product, USAGE=A9, ACTUAL=A9, $
    FIELDNAME='PKG TYPE', ALIAS=Product, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=POPULATION, ALIAS=Market, USAGE=A15, ACTUAL=A15, $
    FIELDNAME='INTRO DATE', ALIAS=Product, USAGE=A21, ACTUAL=A21, $
```

The corresponding Access File contains the member names of the attribute tagged dimension Population. Next to each member is the name of the associated base dimension, Market. GEN3 refers to the generation within the base dimension (Market) that you must reference when forming an Essbase request. For an illustration, see *Reporting Against Attribute Tagged Dimensions* on page 10-19).

```
SEGNAME=BASIC, SERVER=edaknw2, DBNAME=Basic, APPLNAME=Sample, $
 TIMEDIM=Year, $
 MEASURE=Measures, $
 MEMBER=COGS, AGGREGATE=NO, $
 MEMBER=TOTAL_EXPENSES, AGGREGATE=NO, $
 MEMBER=PROFIT_%, AGGREGATE=NO, $
 MEMBER=PROFIT_PER_OUNCE, AGGREGATE=NO, $
 .
 .

 .
 ATTRIBUTE=Small, BASE=Market, GEN=3, $
 ATTRIBUTE=Small_3000000, BASE=Market, GEN=3, $
 ATTRIBUTE=Small_6000000, BASE=Market, GEN=3, $
 ATTRIBUTE=Medium, BASE=Market, GEN=3, $
 ATTRIBUTE=Medium_9000000, BASE=Market, GEN=3, $
 ATTRIBUTE=Medium_12000000, BASE=Market, GEN=3, $
 ATTRIBUTE=Medium_15000000, BASE=Market, GEN=3, $
 ATTRIBUTE=Medium_18000000, BASE=Market, GEN=3, $
 ATTRIBUTE=Large, BASE=Market, GEN=3, $
 ATTRIBUTE=Large_21000000, BASE=Market, GEN=3, $
 ATTRIBUTE=Large_24000000, BASE=Market, GEN=3, $
 ATTRIBUTE=Large_27000000, BASE=Market, GEN=3, $
 ATTRIBUTE=Large_30000000, BASE=Market, GEN=3, $
 ATTRIBUTE=Large_33000000, BASE=Market, GEN=3, $
 .
 .
 .
```

Example: Reporting Against Attribute Tagged Dimensions

This example uses data defined in the sample Master and Access Files in *Generating Attribute Dimensions With the CREATE Synonym Command* on page 10-18.

The following request references the Population attribute dimension in the ATTR segment of the Master File. The request also references a member of the Population attribute dimension, 3000000. Market is the base dimension for the Population attribute dimension, as defined in the Access File. The BY phrase in the request references STATE, which falls within the 3rd Generation (GEN=3) of the Market dimension. This reference is consistent with the following line in the Access File

```
ATTRIBUTE=Small_3000000, BASE=Market, GEN=3, $
```

which names the Base dimension (Market) and identifies the generation (GEN3) that you must reference within that Base dimension when you create an Essbase request using the specified member (3000000) of the Population attribute dimension.

```
TABLE FILE BASIC
PRINT PROFIT
BY STATE
WHERE POPULATION EQ '3000000'
END
```

The output is:

```
STATE            Profit
------------- --------
Iowa          9,061.00
Nevada        4,039.00
New Hampshire 1,125.00
New Mexico      330.00
Utah          3,155.00
```

The next request generates a message because the generation of the referenced field, REGION, falls within the 2nd generation of the base dimension, Market. Therefore, it does not match the generation specified in the Access File (GEN=3) when using the specified member (3000000).

```
TABLE FILE BASIC
PRINT PROFIT
BY REGION
WHERE POPULATION EQ '3000000'
END
```

As a result, the request displays the following message:

```
(FOC43271) Small_3000000 is not an associated attribute of any requested
column
```

Describing Scenario Dimensions in the Master File

How to:

Generate a Pseudo Scenario Dimension

Scenario dimensions in Essbase can be represented in two ways to the server: as *normal* dimensions or as *pseudo* account dimensions. If they are to be described as normal dimensions, the CREATE SYNONYM command generates field descriptions for the dimension. If you wish to describe them as pseudo in order to generate the Scenario dimension in place of the accounts dimension segment, use the SET SCENARIO command. You must issue the SET SCENARIO command before the synonym is created.

Syntax: How to Generate a Pseudo Scenario Dimension

If you wish to generate the Scenario dimension to be used in place of the accounts dimension segment, issue

```
ENGINE ESSBASE SET SCENARIO [DIM dimension_name] {member_name|ALL} FOR synonym
```

where:

dimension_name

Is the Scenario dimension name. Scenario is the default value. If the default name is used in the outline, you can omit DIM *dimension_name*.

When used in the SET SCENARIO command, *dimension_name* is case-sensitive and must match the case in the Essbase outline.

member_name

Specifies a member name in the Scenario dimension to be used to generate a field for every account member intersection. When used in the SET SCENARIO command, member_name is case-sensitive and must match the case in the Essbase outline.

ALL

Indicates that all members of the Scenario dimension are used to generate the Master File. ALL is the default value.

synonym

Is an alias for the data source (it can be a maximum of 64 characters).

Note: You can issue the SET SCENARIO command multiple times to specify that a number of members from the Scenario dimension be used in the generation of the Master File. Once this command has been issued, you can use the CREATE SYNONYM command to generate the Master File name.

For each Scenario/Accounts member intersection, a field is generated in the Master File. If this multiplicity effect causes the Master File that is generated to be invalid (due to the total length of the fields), the CREATE SYNONYM command fails and displays the following message:

```
Total actual or usage exceeds 32768 To cut down,try SET SCENARIO
```

If you receive this message, you will need to issue fewer SET SCENARIO commands to limit the scope of the Master File with regards to the number of Scenario dimensions used.

Describing the Measures Dimension in the Master File

The SET MEASURE command enables you to set or change the Accounts tagged dimension without having to change the outline in the Essbase Database Server. With this setting, iWay produces an actual field name for every member of the named dimension in the Master File rather than producing the dimension in terms of generations. You must issue the SET MEASURE command before the synonym is created.

Syntax: How to Set or Change the Measures Dimension in the Master File

To set the Accounts tag for a dimension, issue

```
ENGINE ESSBASE SET MEASURE dimension_name FOR synonym
```

where:

dimension_name

Is the name of the dimension to be interpreted as an Accounts tagged dimension when generating a synonym.

Generates a Master File in which the Accounts Tagged dimension is represented as generations.

synonym

Is an alias for one *application.database* combination.

Example: Using SET MEASURE

The following command displays the actual member names of the Scenario dimension (for ACTUAL, BUDGET, or VARIANCE), rather than displaying fields like SCENARIO or GEN2_SCENARIO, in the Master File.

```
ENGINE ESSBASE SET MEASURE Scenario FOR SAMPLE
```

Reference: Limiting the Number of Fields in a Master File

To address the 3,072 field limit in the Master File, SET MEASURE can be set to NONE. With this setting, all of the dimensions are interpreted as non-Accounts dimensions in the Master File and represented as generations.

In addition, a segment called DATA is added to the Master File. This segment contains a field called DATA_VALUES, which enables you to display the values of the Measures dimension although the actual member names are not present in the Master File. This field should be used in conjunction with a GEN_MEASURES, MEASURE_MEMBER, or a generation field from the measures dimension.

Syntax: How to Limit the Number of Fields in a Master File

```
ENGINE ESSBASE SET MEASURE NONE FOR SAMPLE
```

where:

`NONE`

Generates a Master File in which the Accounts tagged dimension is represented as generations.

Example: Reporting Against a Master File With a Limited Number of Fields

If SET MEASURE has been set to NONE, you can display the values of the Measures dimension although the actual member names are not present in the Master File. The following is a sample DATA segment, which includes the field DATA_VALUES, against which you can report.

```
$ DIMENSION: DATA
  SEGMENT=DATA ,SEGTYPE=U, PARENT=BASIC, $
  FIELD=DATA_VALUE, ALIAS=DATA_VALUE,
  USAGE=D20.2,ACTUAL=D8,MISSING=ON,
    TITLE=DATA_VALUE $
TABLE FILE SAMPLE
PRINT DATA_VALUE
BY PRODUCT
WHERE GEN2_MEASURE EQ 'Sales'
END
```

The output is:

```
PRODUCT                 DATA_VALUE $
-------------           -------------------
Product                 400,855.00
```

Changing the Default Usage Format of the Accounts Dimension

All dimension descriptions in the Master File, except for the accounts dimension, have the USAGE format of alphanumeric. The default format of the accounts dimension is D20.2. You can change this default using the SET CONVERSION command.

Syntax: How to Change the Default Usage Format of the Accounts Dimension

```
ESSBASE SET CONVERSION format PRECISION n m
```

where:

format

Possible values are:

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`DECIMAL` which indicates that the command applies only to decimal columns.

n

Is the precision. It must be a valid number representing the maximum value for precision for the data type.

m

Is a valid number representing the maximum value for scale for the data type.

If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set *m* to 0 (zero).

Customizing the Essbase Environment

In this section:

The Adapter for Essbase provides several parameters for customizing the environment and optimizing performance. The following topics provide an overview of customization options.

Adapter Functionality

The Adapter for Essbase functionality supports:

- JOINs.
- SUM and PRINT commands with Accounts/Measures dimensions.
- Reporting on parent/child views.

Reference: JOIN Support

The Adapter for Essbase does not support direct JOINs from or to a cube. Therefore, SQL joins using a WHERE clause are not supported.

To achieve a joined request, use the FOCUS MATCH FILE command or create a HOLD file and use the HOLD file in the JOINed request.

Reference: SUM and PRINT Support

The Essbase Server does not support the use of SUM or PRINT on an Accounts dimension member without reference to another non-Accounts dimension member in the request.

For example, since this request refers only to an Accounts dimension member,

```
TABLE FILE SAMPLE
PRINT SALES
END
```

the following message displays:

```
(FOC43244) No active dimension found in the request.
```

Reference: Parent/Child View Support

- You cannot reference PARENT or PARENTCAP from the parent/child view without referencing MEMBER or CAPTION.

  For example, the following request is incorrect since it does not reference MEMBER or CAPTION.

  ```
  TABLE FILE SAMPLE
  PRINT SALES
  BY PRODUCT_PARENT
  WHERE PRODUCT_PARENT EQ '100'
  END
  ```

  The following message displays:

  ```
  (FOC43248) Can't reference PRODUCT's PARENT,PARENTCAP without MEMBER
  or CAPTION.
  ```

  The correct request is:

  ```
  TABLE FILE SAMPLE
  PRINT SALES
  BY PRODUCT_PARENT
  BY PRODUCT_MEMBER
  WHERE PRODUCT_PARENT EQ '100'
  END
  ```

  The output is:

  ```
  PRODUCT_PARENT     PRODUCT_MEMBER              Sales
  ---------------    ---------------    --------------
  100                100-10                  62,824.00
                     100-20                  30,469.00
                     100-30                  12,841.00
  ```

- You cannot reference fields from the Generation view (GEN1_PRODUCT or GEN2_PRODUCT) and from the parent/child view (PRODUCT_MEMBER or PRODUCT_CAPTION) in one request if the fields from these different views are from the same dimension.

  For example, in the following request

  ```
  TABLE FILE SAMPLE
  PRINT SALES
  BY PRODUCT_MEMBER
  BY FAMILY
  WHERE PRODUCT_MEMBER EQ '100'
  END
  ```

  FAMILY is a member of the Product dimension and part of the Generation view so the following message is generated:

  ```
  (FOC43247) Can't reference fields from Product2 and Product at the
  same time.
  ```

  The correct request is:

  ```
  TABLE FILE SAMPLE
  PRINT SALES
  BY PRODUCT_MEMBER
  BY SCENARIO
  WHERE PRODUCT_MEMBER EQ '100'
  END
  ```

  The output is:

  ```
  PRODUCT_MEMBER  SCENARIO         Sales
  --------------  --------         -----
  100             Scenario    106,134.00
  ```

Specifying ALIAS Names

In Essbase, you can assign more than one name to a member or a shared member. For example, in the *Sample.Basic* database, the PRODUCT dimension contains members that can be identified by both product codes, such as 200, or by more descriptive names, such as COLA. By default, the output values for members specified in a request are the member values. The server allows you to take advantage of the more descriptive member names in the Essbase outline when formulating a request.

Syntax: How to Specify ALIAS Names in a Request

```
ENGINE ESSBASE SET USEALIASNAME {ON|OFF} [FOR synonym]
```

where:

`ON`

Allows the use of ALIAS names in a request.

`OFF`

Does not allow the use of ALIAS names in a request. OFF is the default value.

`synonym`

Is an alias for the data source (it can be a maximum of 64 characters).

Note: If the SET USEALIASNAME command is used without FOR *synonym*, the setting applies to all Master Files. If the *synonym* option is used, the setting applies only to that Master File. You can issue multiple commands to control the use of the ALIAS name at the Master File level.

In an Essbase script, the reporting keyword OUTALTNAMES is equivalent to ALIAS.

Specifying ALIAS and Member Names in One Request

How to:

Specify ALIAS and Member Names in a Request

Example:

Generated Master File sample.mas

Using ALIAS and Member Names in an Essbase Request

To specify the member (or shared member) and the ALIAS name in a report, you must create a synonym using the PARMS 'ALIASFIELD' option. The Master File generated using this option will contain an ALIAS field for each field in the generation view.

Syntax: How to Specify ALIAS and Member Names in a Request

```
CREATE SYNONYM app/synonym FOR application.database DBMS ESSBASE AT
connection_name PARMS 'ALIASFIELD'
END
```

where:

`connection_name`

Is the server name used as a connect descriptor to the Essbase Server across the network.

`ALIASFIELD`

Is the parameter at the end of the CREATE SYNONYM statement that creates an additional ALIAS_FIELDNAME for each field of the generation view in the Master File.

Example: Generated Master File sample.mas

```
FILENAME=SAMPLE, SUFFIX=ESSBASE, $
SEGMENT=BASIC, SEGTYPE=S0, $
.
.
.
$  DIMENSION: Product
    FIELDNAME=PRODUCT, ALIAS='Lev2,Product', USAGE=A7, ACTUAL=A7,
      WITHIN='*Product', $
    FIELDNAME=FAMILY, ALIAS=Family, USAGE=A4, ACTUAL=A4,
      WITHIN=PRODUCT, $
    FIELDNAME=FAMILY_ALIAS, ALIAS=Family_ALIAS, USAGE=A11, ACTUAL=A11,
      REFERENCE=FAMILY, PROPERTY=CAPTION,  $
    FIELDNAME=SKU, ALIAS=SKU, USAGE=A6, ACTUAL=A6,
      WITHIN=FAMILY, $
    FIELDNAME=SKU_ALIAS, ALIAS=SKU_ALIAS, USAGE=A18, ACTUAL=A18,
      REFERENCE=SKU, PROPERTY=CAPTION,  $
```

Note: The SET USEALIASNAME ON/OFF command is ignored when the Master File is created using this option.

For information about the SET USEALIASNAME command, see *Specifying ALIAS Names* on page 10-28.

Example: Using ALIAS and Member Names in an Essbase Request

```
TABLE FILE SAMPLE
PRINT SALES PROFIT
BY SKU
BY SKU_ALIAS
WHERE SKU_ALIAS EQ 'Cola'
END
```

The output is:

```
SKU       SKU_ALIAS    Sales        Profit
---       ---------    -------      --------
100-10    Cola         62,824.00    22,777.00
```

Specifying ALIAS Tables

In Essbase, aliases are generally stored in one or more tables as part of the database outline. Once the SET command is active, screening conditions must use the ALIAS value, not the member value.

If multiple ALIAS tables exist for an outline, you can use the SET ALIASTABLE command to specify which ALIAS table to use for a query.

Syntax: How to Specify an ALIAS Table

```
ENGINE ESSBASE SET ALIASTABLE {aliastablename|RESET} FOR synonym
```

where:

aliastablename

Is the ALIAS table name to be used.

RESET

Sets the alias table back to the current default for the outline in Essbase.

synonym

Is the Master File name.

Note: This SET ALIASTABLE command is used in conjunction with the SET USEALIASNAME command. If that command is not issued, the SET ALIASTABLE command is ignored. For more information, see *Specifying ALIAS Names* on page 10-28.

Setting the Maximum Number of Rows Returned

The default number of rows that can be returned from an Essbase cube using the server is 10,000. This can be controlled by the MAXROWS setting, which can be set in any supported server profile.

Syntax: How to Set the Maximum Number of Rows Returned

To limit or extend the number of rows returned, use the syntax

```
ENGINE ESSBASE SET MAXROWS n [FOR synonym]
```

where:

n

Is a valid number that either limits or extends the number of rows to be returned. The default number of rows returned without this setting is 10,000. The maximum number for this setting is 999999999.

synonym

Is an alias for the data source (it can be a maximum of 64 characters).

Time Series Reporting

By using Time Series and Accounts tags within your Essbase outline, you can tell Essbase how to calculate your accounts data. When you tag a dimension as Time, Essbase knows that this is the dimension on which to base the time periods for the Accounts tags.

The server supports all eight levels of period-to-date reporting within Essbase. See the *Hyperion Essbase Database Administrator's Guide* for further information.

In order to retrieve values from members in an Accounts dimension using time-series reporting, you must refer to at least one member of the Time dimension using the WHERE clause.

Example: Using Time Series Reporting

```
SELECT PROFIT,PRODUCT,QTD FROM BASIC WHERE QTD = 'Mar' ORDER BY PRODUCT,QTD
```

Where PROFIT is a member of the Accounts dimension, PRODUCT is a member of a non-Accounts dimension, and QTD is a member of the Time dimension. The above request returns the following row:

```
Product Q-T-D(Mar)           24703.00
```

Note: All results reference the Dynamic Time series member with dashes between letters.

Summing on Non-Aggregated Fields

How to:

Turn Aggregation ON/OFF For Non-Aggregated Fields

Example:

Using RESTRICTSUM on Non-Aggregated Fields

Any two-pass calculated members that are found in the accounts (or Scenario) dimension and members with the (-), (\), (*), and (%) consolidation properties in the Essbase outline, have the following attributes in the associated Access File:

```
MEMBER=membername, AGGREGATE=NO, $
```

Due to the nature of two-pass calculation members, a summation request (SQL SUM... or SUM in TABLE) may produce incorrect values. Therefore, if the member has the AGGREGATE=NO attribute in the Access File, a SUM action in the request against the member results in the following message:

```
(FOC43241) Aggregation is requested for non aggregatable member(s)
```

The server allows SUM to be used on non-aggregated fields in the Access File.

Syntax: How to Turn Aggregation ON/OFF For Non-Aggregated Fields

```
ENGINE ESSBASE SET RESTRICTSUM [ON|OFF] FOR synonym
```

where:

ON

Does not allow the use of SUM on non-aggregated fields. ON is the default value.

OFF

Allows the use of SUM on non-aggregated fields.

synonym

Is an alias for the data source (it can be a maximum of 64 characters).

Example: Using RESTRICTSUM on Non-Aggregated Fields

Using this Access File

```
SEGNAME=BASIC, SERVER=unxsol26, DBNAME=Basic, APPLNAME=Sample, $
TIMEDIM=Year, $
MEASURE=Measures, $
MEMBER=COGS, AGGREGATE=NO, $
MEMBER=TOTAL_EXPENSES, AGGREGATE=NO, $
MEMBER=PROFIT_%, AGGREGATE=NO, $
MEMBER=PROFIT_PER_OUNCE, AGGREGATE=NO, $
```

issue the following request:

```
TABLE FILE SAMPLE
SUM COGS
BY QUARTER
END
```

If RESTRICTSUM ON (default) is set in the profile, the following message is displayed and no output is generated:

```
(FOC43241) Aggregation is requested for non aggregatable member(s)
```

If you set RESTRICTSUM OFF in the profile, SUM is permitted on non-aggregated fields, producing this output:

```
QUARTER                      COGS
-------                 ---------
Qtr1                    42,877.00
Qtr2                    45,362.00
Qtr3                    47,343.00
Qtr4                    43,754.00
```

Preventing Aggregation of Non-Consolidating Members

How to:

Prevent Aggregation of Non-Consolidating Members

Member consolidation properties determine how children roll up into their parents in the Essbase outline. The members with the (~) as a consolidation property in the Essbase outline, are *not* rolled up in the database. The SET AGGREGATE NOOP (No Operation) setting only affects those Essbase members in the Accounts dimension with the (~) consolidation property. If AGGREGATE NOOP is set to OFF, SUM is not permitted on Essbase members with the (~) consolidation property and those members appear in the Access File as:

```
MEMBER=ADDITIONS, AGGREGATE=NO, $
```

Syntax: How to Prevent Aggregation of Non-Consolidating Members

You must turn set AGGREGATE NOOP to OFF before you create the synonym.

```
ENGINE ESSBASE SET AGGREGATE NOOP {ON|OFF}
```

where:

`ON`

Allows the use of SUM on NOOP Essbase members. With this setting, Essbase members are not added to the Access File. ON is the default value.

`OFF`

Does not allow the use of SUM on NOOP Essbase members. Members with the (~) consolidation property are added to the Access File.

Note: Any member of the Accounts dimension in the Essbase outline tagged with (-), (/), (*), or (%) as a consolidation property automatically appears in the Access File with the AGGREGATE=NO setting.

Suppressing Shared Members

How to:

Suppress Shared Members in Essbase

Example:

Suppressing Shared Members

Shared Members in Essbase store pointers to data that is stored in the real member. Therefore, although the data is shared between the two members, it is only stored one time. You can exclude shared members from reports using the SUPSHARE setting.

Syntax: How to Suppress Shared Members in Essbase

```
ENGINE ESSBASE SET SUPSHARE [ON|OFF]
```

where:

`ON`

Excludes shared members in a report.

`OFF`

Includes shared members in a report. OFF is the default value.

Example: Suppressing Shared Members

The following request and output illustrates the affect of turning SUPSHARE OFF and ON.

```
TABLE FILE SAMPLE
PRINT SALES BY FAMILY BY SKU
AND COLUMN-TOTAL
WHERE FAMILY EQ 'Diet' OR '200'
END
```

With SUPSHARE OFF, the output is:

```
FAMILY  SKU          SALES
------  ------  ---------
200     200-10  41,537.00
        200-20  38,240.00
        200-30  17,559.00
        200-40  11,750.00
Diet

        100-20  30,469.00
        200-20  38,240.00
        300-30  36,969.00
==========================
TOTAL   214,764.00
```

With SUPSHARE ON, the output is:

```
FAMILY  SKU          SALES
------  ------  ---------
200     200-10  41,537.00
        200-20  38,240.00
        200-30  17,559.00
        200-40  11,750.00
Diet

        100-20  30,469.00
        300-30  36,969.00
==========================
TOTAL   176,524.00
```

Notice that member 200-20 is displayed twice in the first report, but only once in the second report. Turning SUPSHARE ON prevents a shared member from being duplicated in a report request.

Suppressing Zero Values

Essbase stores zero values that you can exclude from reports using the SUPZEROS setting. This setting applies to an entire row. If one member in a row has a value of zero, and the other values in the same row have values other than zero, the zero value will not be suppressed.

Syntax: How to Suppress Rows With Zero Values

```
ENGINE ESSBASE SET SUPZEROS [ON|OFF]
```

where:

`ON`

Suppresses an entire row that contains 0 values.

`OFF`

Does not suppress zero values. OFF is the default value.

Example: Suppressing Zero Values

The following output illustrates the affect of turning SUPSHARE OFF and ON.

With SUPZEROS OFF, the output is:

```
STATE             SALES       PROFIT
---------   ---------    ---------
Colorado    38,240.00    42,877.00
Louisiana   0.00         0.00
Nevada      29,156.00    47,343.00
```

With SUPZEROS ON, the output is:

```
STATE             SALES      PROFIT
--------    ---------   ---------
Colorado    38,240.00   42,877.00
Nevada      29,156.00   47,343.00
```

Suppressing Missing Data

In Essbase, empty cells are known as missing or #MISSING data. You can suppress missing data during reporting using the SUPMISSING setting.

Syntax: How to Suppress Rows With Missing Data

```
ENGINE ESSBASE SET SUPMISSING [ON|OFF]
```

where:

`ON`

Suppresses an entire row that contains #MISSING data.

`OFF`

Does not suppress rows with #MISSING data. OFF is the default value.

Example: Suppressing Missing Data

The following output illustrates the affect of turning SUPMISSING OFF and ON.

With SUPMISSING OFF, the output is:

```
STATE            SALES      PROFIT
---------    ---------   ---------
Colorado     38,240.00   42,877.00
Louisiana    #MISSING    #MISSING
Nevada       29,156.00   47,343.00
```

With SUPMISSING ON, the output is:

```
STATE            SALES      PROFIT
--------     ---------   ---------
Colorado     38,240.00   42,877.00
Nevada       29,156.00   47,343.00
```

Suppressing Zero Values and Missing Data

You can use the SUPEMPTY setting to suppress an entire row that contains either zero values, missing (#MISSING) data, or a combination of the two.

Syntax: How to Suppress Rows With Zero Values or Missing Data

```
ENGINE ESSBASE SET SUPEMPTY [ON|OFF]
```

where:

`ON`

Suppresses an entire row that contains either zero values, #MISSING data, or a combination of the two.

`OFF`

Does not suppress rows that contain either zero values or #MISSING data. OFF is the default value.

Example: Suppressing Rows With Zero Values or Missing Data

The following output illustrates the affect of turning SUPEMPTY OFF and ON.

With SUPEMPTY OFF, the output is:

```
STATE            SALES     PROFIT
---------    ---------  ---------
Colorado     38,240.00  42,877.00
Louisiana    #MISSING   #MISSING
Nevada       0.00       0.00
```

With SUPEMPTY ON, the output is:

```
STATE            SALES     PROFIT
--------     ---------  ---------
Colorado     38,240.00  42,877.00
```

Substitution Variables

Substitution variables act as global placeholders for information that changes regularly. Each substitution variable configured in Essbase is assigned a value. You can change the value at any time, thus limiting the number of manual changes required in your scripts when you are running reports.

When you reference an Essbase substitution variable in a TABLE command, the substitution variable must be preceded with an up arrow (^) and enclosed in single quotation marks.

Example: Using a Substitution Variable in a Request

In this example, CurrMonth is the name of the substitution variable that was set in Essbase. The ^ and quotation marks are required.

```
TABLE FILE SAMPLE
PRINT Product
BY
PRODUCT
WHERE
MONTH EQ '^CurrMonth'
END
```

CHAPTER 11

Using the Adapter for CA-IDMS/DB

Topics:

The Adapter for CA-IDMS/DB for OS/390 and z/OS allows applications to access IDMS/DB data sources. The adapter converts application requests into native IDMS/DB statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Note: In the remainder of this topic, the following terms all refer to the Adapter for CA-IDMS/DB.

- CA-IDMS/DB
- IDMS/DB
- Adapter for IDMS/DB

Preparing the IDMS/DB Environment

Prior to configuring the Adapter for IDMS/DB using the Web Console, it is necessary to edit the ISTART JCL that is used to initiate the server.

Procedure: How to Edit the ISTART JCL

1. Allocate the IDMS.LOADLIB and IDMS.DBA.LOADLIB libraries to the server's STEPLIB DDNAME allocation.
2. Allocate DDNAME SYSCTL to the IDMS.SYSCTL data set.
3. Allocate DDNAME SYSIDMS to the IDMS.SYSIDMS data set.

Configuring the Adapter for IDMS/DB

Configure the adapter using the Web Console Adapter Add screen and click *Configure*.

Declaring Connection Attributes

The integrity of the IDMS/DB data source is not jeopardized because the adapter provides read-only access. Access to the data source is restricted by the security package in use on the OS/390 system (such as RACF) and by the CA-IDMS/DB security features.

Both CA-IDMS/DB and the server provide security mechanisms to ensure that users have access to only those objects for which they have authorization.

IDMS/DB Overview and Mapping Considerations

In this section:

CA-IDMS/DB databases, both network and LRF-based, correspond to hierarchical and relational data sources. The concepts discussed in this section affect the way IDMS/DB files are described to the server.

Mapping Concepts

IDMS/DB is a network database management system that is accessed by a subschema (the equivalent in the server is a Master File). IDMS/DB provides two methods of retrieving records within a subschema from an application program:

- Network access (DML)
- LRF-based access (LRF)

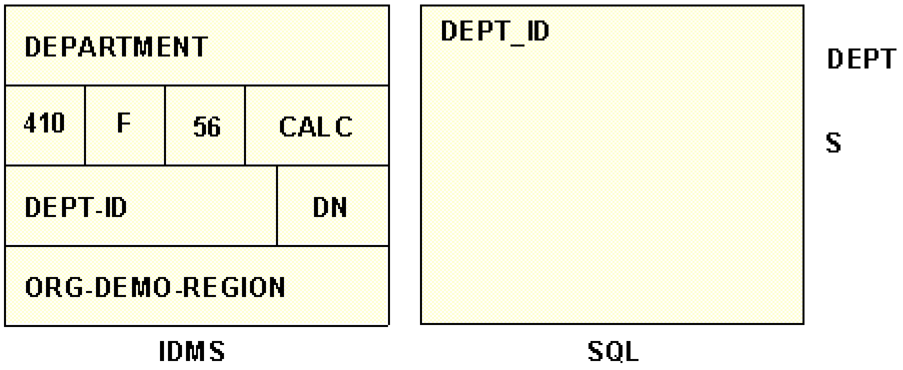
Data Management Language (DML) access is the traditional method of IDMS/DB database navigation. It is the network navigation facility. Each physical record is retrieved separately. An application program enters the IDMS/DB database at a particular IDMS/DB record type (the equivalent in the server is a segment) and searches the set connections to retrieve the required data. For mapping concepts that apply to network record types, see *Summary of Network Relationships* on page 11-16.

Logical Record Facility (LRF) access, available as of IDMS/DB Release 5.7, is the relational-like method of access. LRF provides software to dynamically create flat records from one or more network record types at run time. For mapping concepts that apply to LRF-based records, see *Logical Record Facility Concepts* on page 11-14.

Network Concepts

An IDMS/DB record type is described to the server as a segment. Since DML retrieval is record-oriented rather than field-oriented, the Master File must list the fields in the same order as they appear on the IDMS/DB record type. However, a Master File does not have to list every field of a particular record type; you may omit fields after a given field. For example, your description may list the first four fields of a 10-field IDMS/DB record type. You can use IDMS/DB field names in your Master File up to a maximum of 48 characters.

The following graphic illustrates how a record type is described as a segment. The record type DEPARTMENT contains the field-type DEPT-ID. Its corresponding server segment DEPT contains a field named DEPT_ID.



Repeating fields on an IDMS/DB record type are defined as OCCURS segments.

Network record types may be related to each other. These relationships may be physical (using set connections) or logical (achieved through an index or CALC field). The Adapter for IDMS/DB supports both physical and logical relationships.

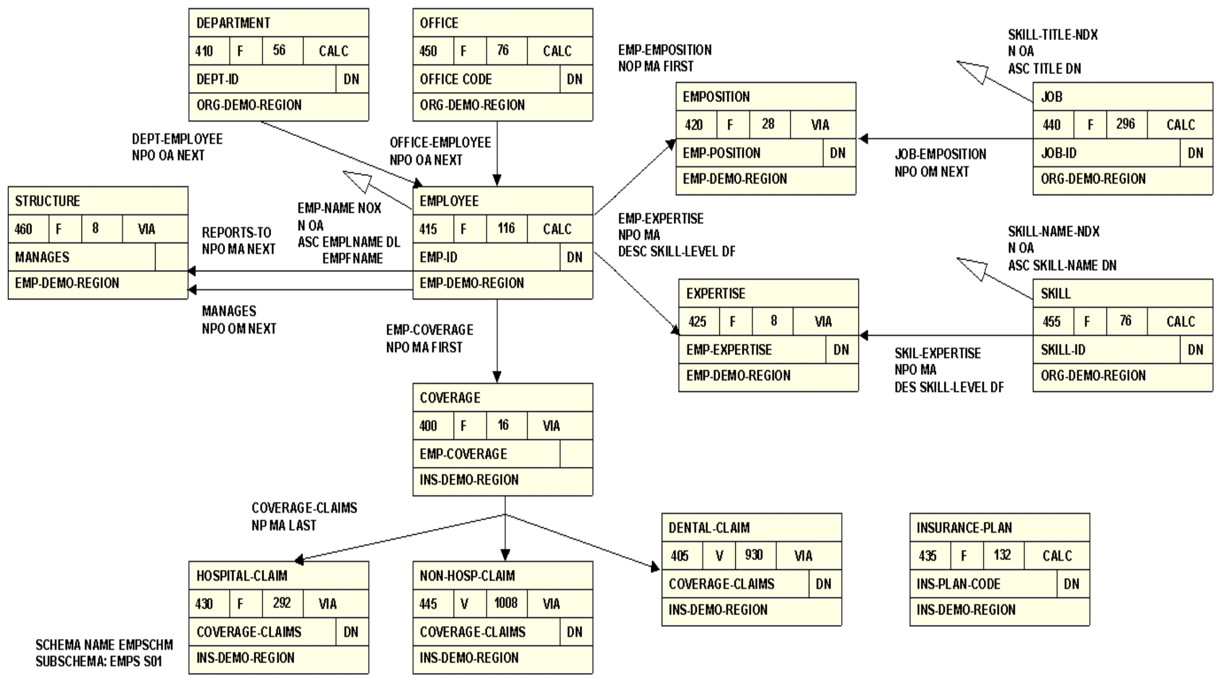
Set-Based Relationships

In an IDMS/DB database, physical relationships between record types are achieved with pointers that correspond to IDMS/DB sets. A set implements a one-to-many relationship between record types. The server equivalent of a set is the parent/descendant relationship between segments. In an IDMS/DB set, one record type acts as the owner (the one side of the relationship) and one or more record types act as the members (the many side of the relationship). A single IDMS/DB record type can participate in several set relationships as either the owner or the member.

The IDMS/DB representation of record types and set relationships within a database is called a Bachman diagram, also known as a data structure diagram. In a Bachman diagram, a record type is depicted as a box; a set, as a line with an arrow. Set names appear as labels beside the arrows. The box that the arrow points to is the member record type. Triangles indicate indexes. The next graphic is the Bachman diagram for the IDMS/DB network subschema EMPSS01. Sections of this diagram are referenced throughout these sections.

The following kinds of set-based relationships are depicted: simple set, common owner, common member, multi-member, bill-of-materials (simple and multi-tiered), and loop structures.

All relationships correspond to server structures and are explained following the diagram:



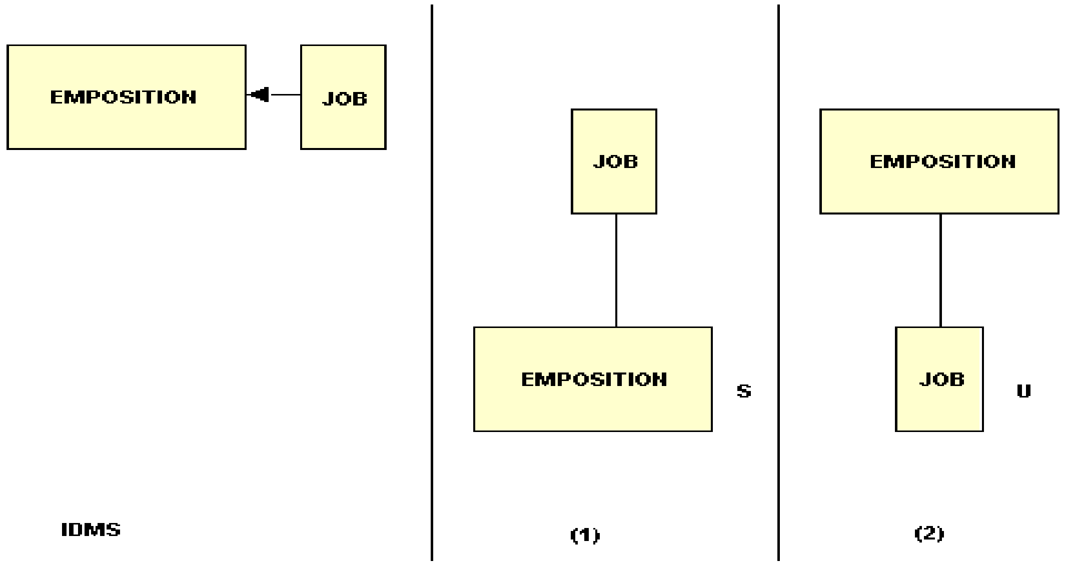
Simple Set

The graphic below illustrates the basic mapping principle of a simple set: an IDMS/DB record type corresponds to a segment; a set relationship corresponds to a parent/descendant relationship.

This figure also illustrates a second principle: an IDMS/DB structure can have more than one representation as a hierarchy. The type of parent/descendant relationship required in the Master File depends on whether the owner or the member record type is designated as the parent segment.

The JOB-EMPOSITION set has two possible representations:

1. It shows the JOB record type, the owner in the set, mapped to the server as the root segment. Since a member record type is multiply-occurring (for example, several instances of EMPOSITION records per JOB instance, indicating that many positions share one job title and description), the EMPOSITION record type is displayed as a non-unique descendant.
2. It depicts the reverse. The EMPOSITION record type is the root segment and JOB is the unique descendant, since an EMPOSITION instance can have only one owner (only one job title and description per position).

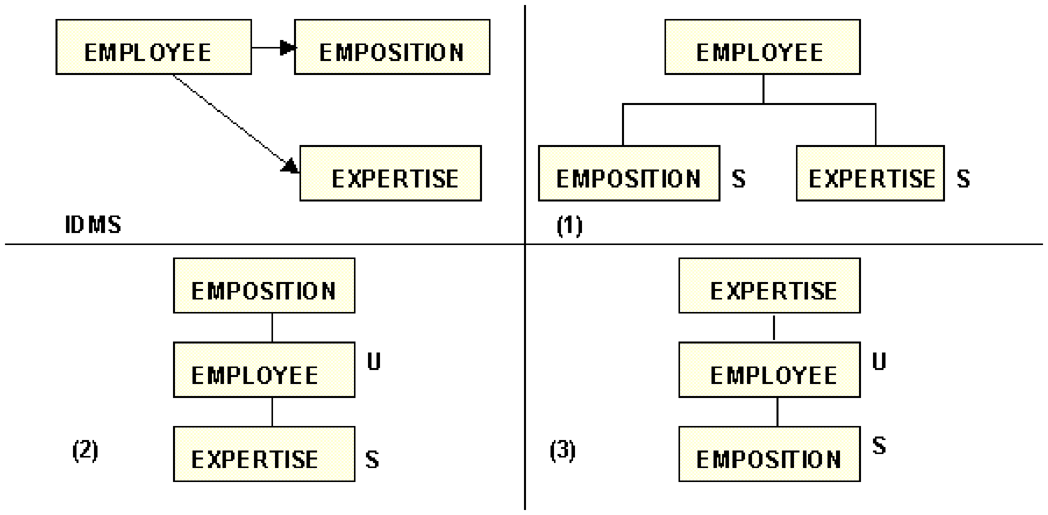


Common Owner

Given the rules in the previous example, consider a more complex scenario called a common owner. A common owner structure contains a record type that is the owner of two or more record types. Several representations are possible.

For example, the graphic below depicts three ways to describe the EMPLOYEE, EXPERTISE, and EMPOSITION structure in one Master File:

1. It shows the EMPLOYEE record type, the owner in both sets, mapped to the server as the root segment of EMPOSITION and EXPERTISE. Since an EMPLOYEE record can have many EMPOSITION and EXPERTISE records, both descendant records are non-unique.
2. It depicts the EMPOSITION record type as the root segment of EMPLOYEE, which acts as the parent of EXPERTISE. Since an EMPOSITION record can have only one owner, EMPLOYEE is a unique descendant; EXPERTISE is a non-unique descendant of EMPLOYEE.
3. It depicts the EXPERTISE record type as the parent of EMPLOYEE, which acts as the parent of EMPOSITION. EMPLOYEE is a unique descendant and EMPOSITION is a non-unique descendant of EMPLOYEE.

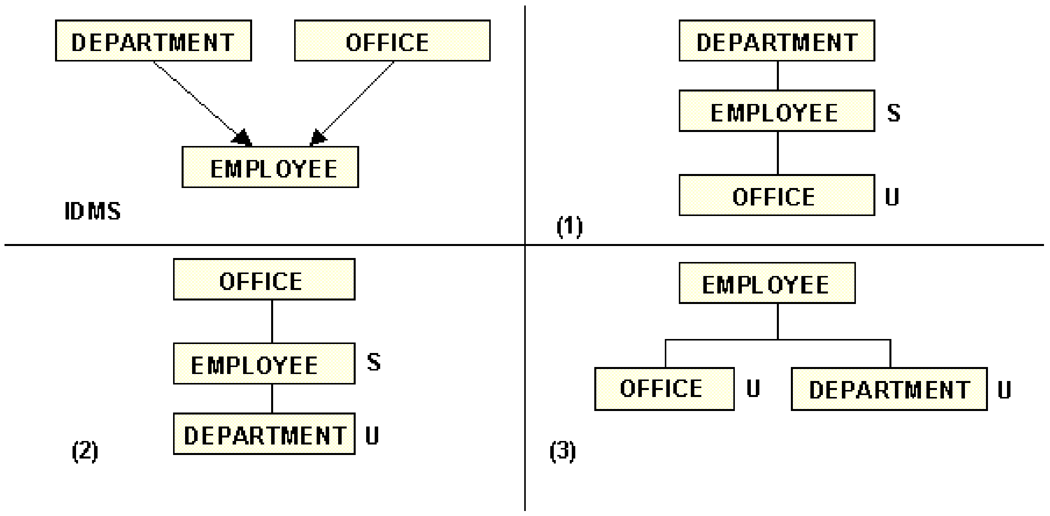


Common Member

When an IDMS/DB record type is a member of two or more sets, the association of the owner record type as the parent segment must be abandoned for one or more sets, because a segment can have only one parent. The graphic below displays the possible interpretations for this IDMS/DB configuration.

In the graphic, the EMPLOYEE record type is a common member in the DEPT-EMPLOYEE and the OFFICE-EMPLOYEE sets. This structure can be described to the server in three ways:

1. It shows DEPARTMENT as the root segment with EMPLOYEE as its non-unique descendant; and OFFICE is the unique descendant of EMPLOYEE.
2. It depicts the reverse: OFFICE is the root segment; EMPLOYEE is its non-unique descendant; and DEPARTMENT is the unique descendant of EMPLOYEE.
3. It shows the only other alternative: EMPLOYEE is the parent of OFFICE and DEPARTMENT. Both are unique, since an EMPLOYEE can belong to only one OFFICE and DEPARTMENT.



Notice that the rules for simple sets are still valid:

- If the owner record type is the parent segment, the member record type as a descendant segment is non-unique.
- If the member record type is the parent segment, the owner record type as a descendant segment is unique.
- A member or an owner record type may act as a root segment (the server ignores whether the root segment is unique or non-unique).

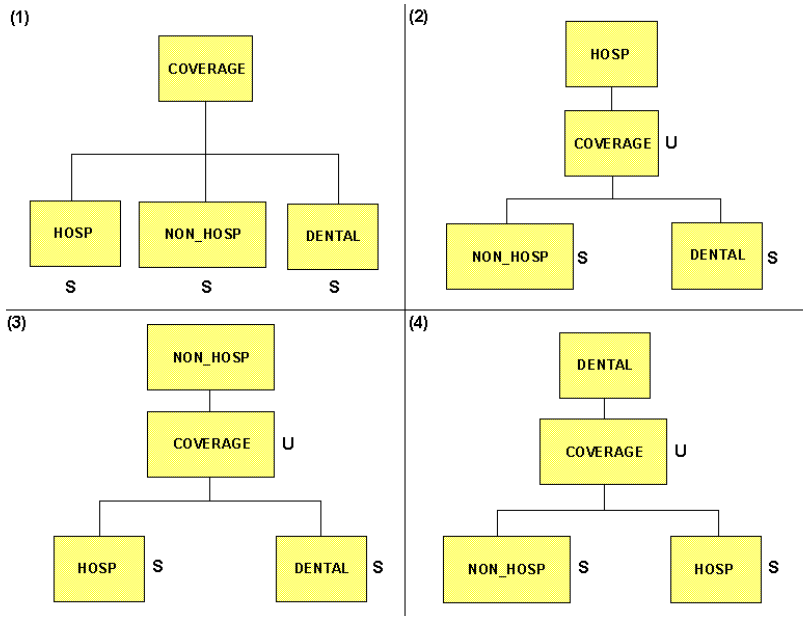
It may be helpful to think of the hierarchical depiction of a network structure as its navigational path. From an IDMS/DB standpoint, panel 1 of the previous graphic shows that the IDMS/DB database can be entered at the DEPARTMENT record type. The corresponding EMPLOYEE record occurrences for a DEPARTMENT record occurrence can be obtained by searching the DEPT-EMPLOYEE set. For each EMPLOYEE record occurrence, obtaining the owner in the OFFICE-EMPLOYEE set retrieves the corresponding OFFICE record occurrence. This is a three-segment retrieval hierarchy that maps to server descriptions.

Multi-Member

When there is more than one member record type, the set is called a multi-member set. A multi-member set is represented in the Master File exactly like a common owner set. The fact that the two relationships are based on the same set is stated in the Access File.

For example, to describe the COVERAGE record type and its three members of the COVERAGE-CLAIMS set, you may choose one of four ways as depicted in the following graphic:

1. The first panel shows the COVERAGE record type, owner of the multi-member set, as the root segment. Since several instances of CLAIMS can be reported against one insurance policy (COVERAGE), each member is a non-unique descendant.
2. The second panel depicts HOSPITAL-CLAIM as the parent of COVERAGE, and the other two member record types as descendants of COVERAGE. In each case, a claim can be reported against only one insurance policy. This explanation applies to panels 3 and 4 as well.



Bill-of-Materials

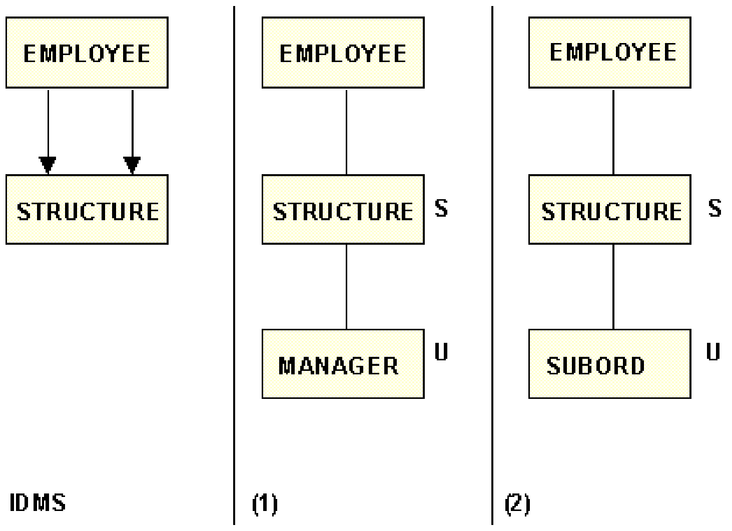
Bill-of-materials structures are classified as simple or multi-tiered. In this section, the simple version is discussed first.

An instance of two record types being linked by more than one set is called a bill-of-materials structure. This structure describes a many-to-many relationship between record occurrences of the same record type. The member record type is the junction record type between the two related owners.

For a simple bill-of-materials structure, the server requires that the owner record type be represented as two or more segments with different field names for the identical fields. This ensures that, at retrieval time, the field names specified in the user's request will provide the proper navigational path.

In the following graphic, the EMPLOYEE and STRUCTURE record types are connected by two sets. This simple bill-of-materials structure can be described two ways:

1. As an employee-to-manager relationship. The EMPLOYEE record type is the parent segment of the non-unique STRUCTURE segment using the REPORTS-TO set. The STRUCTURE record type, in turn, is the parent segment of the unique MANAGER segment using the MANAGES set (the MANAGER segment duplicates the EMPLOYEE segment and its fields are renamed).
2. As an employee-to-subordinate relationship. The EMPLOYEE record type is the parent segment of the non-unique STRUCTURE segment using the MANAGES set. The STRUCTURE record type is the parent segment of the unique SUBORD segment using the REPORTS-TO set (the SUBORD segment duplicates the EMPLOYEE segment and its fields are renamed).



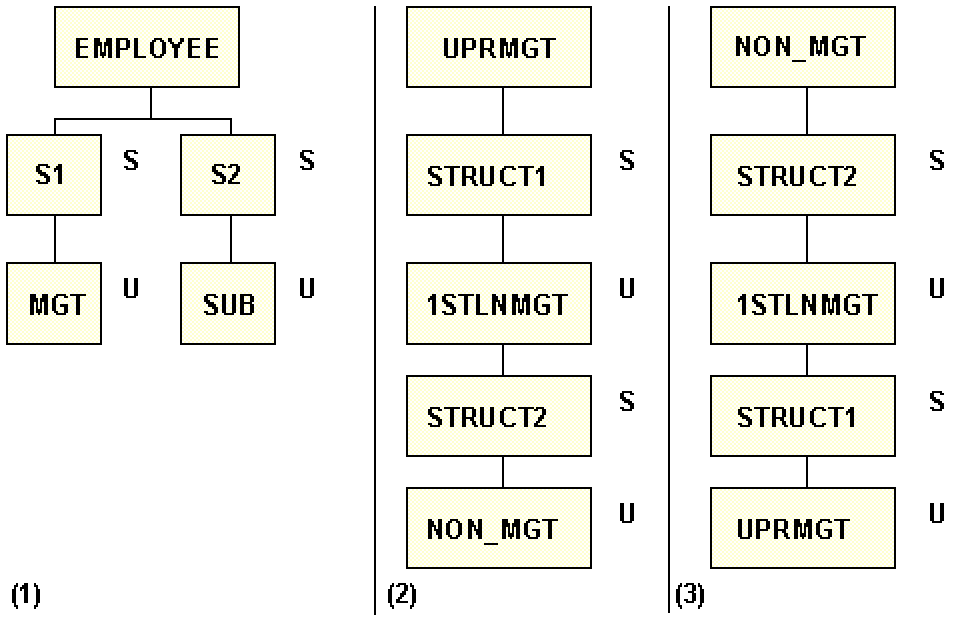
The previous graphic represents a two-tiered employee-to-employee relationship. Multi-tiered relationships are extended bill-of-materials structures. Multi-tiered relationships are created and used for different levels of answer sets. The number of levels or tiers should be kept to a minimum.

To determine the number of segments required to describe an n-tiered relationship, use this formula:

```
Number of segments = (2 x number of tiers) - 1
```

The following graphic is a three-tiered version of the previous graphic:

1. The first panel combines both views with EMPLOYEE as the parent of two non-unique descendants, S1 and S2. Both S1 and S2, in turn, are parents of a unique descendant, MGT and SUB, respectively (S1 and S2 describe junction records that point to MGT and SUB).
2. The second panel shows a three-tiered relationship between employees implemented in a five-segment single-path hierarchy. The segments UPRMGT (upper management), 1STLNMGT (first-line management), and NON_MGT (non-management) all describe the EMPLOYEE segment but have renamed fields (like S1 and S2 above, STRUCT1 and STRUCT2 contain renamed fields that point to descendant segments).
3. The third panel depicts the opposite of panel 2.

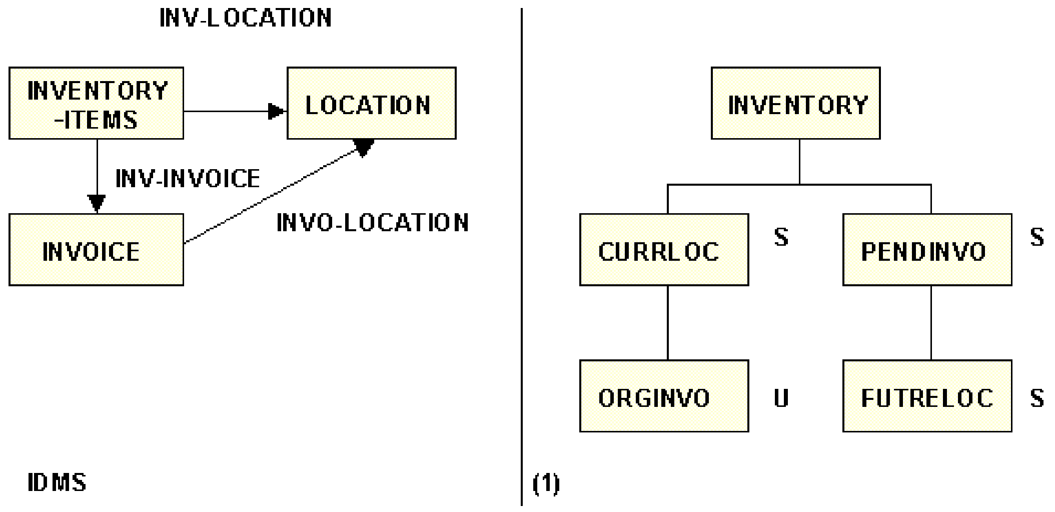


Loop Structures

Loop structures in IDMS/DB implement complicated relationships among record types. For the server depiction of a loop, you must select a record type to be the parent in the relationship.

Suppose you had a loop structure like the one below. An INVOICE record occurrence has an owner record occurrence (INVENTORY-ITEMS) only if the invoice order is pending or has not been delivered. The INVO-LOCATION set lists the location where an invoice item will be or was delivered.

In panel 1 in the following graphic, the server translates this structure as multi-tiered, with one tier of renamed segments. The INVENTORY-ITEMS record type, in this case, acts as the root INVENTORY. The LOCATION record type is mapped as the CURRLOC segment that lists the location of items currently in stock. The INVOICE record type is mapped as the PENDINVO segment that lists pending invoice orders. Segments ORGINVO and FUTRELOC are required segments that rename the data in record types INVOICE and LOCATION. The segment ORGINVO contains historical information for an inventory item. The segment FUTRELOC indicates where an ordered item will be delivered.



CALC-Based and Index-Based Relationships

Logical relationships, unlike physical ones, are based on the occurrence of the same data value in two different record types. To make the parent/descendant connection, the Adapter for IDMS/DB uses CALC fields or indices to locate the related record occurrence(s). The related fields are not required to have the same name in both record types, but the field format must be the same.

Note:

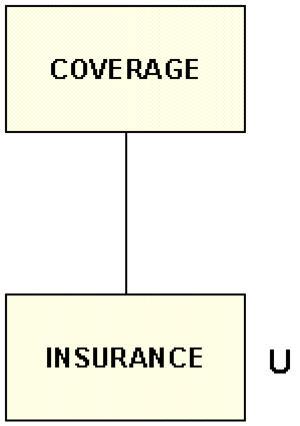
- WHEREs on CALC fields and indices are also used to generate CALC and index calls to IDMS/DB.
- CALC or index fields can also be GROUPNAMEs, consisting of fields contiguous in both parent and descendant segments. The format types and lengths of the GROUP and its fields must be comparable in both parent and descendant segments.

Like set-based descendants, CALC- and index-based descendants are unique or non-unique, but this depends largely on how the DUPLICATES parameter is specified in the Access File (the SEGTYPE parameter for the descendant must also reflect the DUPLICATES parameter; for example, CLCDUP=N, SEGTYPE=U). The server treats unique descendants in the same manner regardless of what underlies the parent/descendant relationship: set, index, or CALC field.

The COVERAGE and INSURANCE-PLAN record types in the graphic in *Set-Based Relationships* on page 11-5 illustrate a logical relationship. The field PLAN-CODE is common to both record types. The parent/descendant relationship can be described to the server if the:

- INSURANCE-PLAN record type has an index on PLAN-CODE.
- INSURANCE-PLAN record type is an IDMS/DB CALC record type with PLAN-CODE as the CALC key.

The graphic below depicts the server representation of the graphic in *Set-Based Relationships* on page 11-5 that uses the CALC field method. The server interprets COVERAGE as the parent segment and INSURANCE-PLAN as the descendant. Since the DUPLICATES parameter is set to N (CLCDUP=N), the INSURANCE segment is unique.



Logical Record Facility Concepts

The Adapter for IDMS/DB supports two kinds of LRF-based records:

- Logical Records (LR) are built from network record types. They are not physically stored as such; instead, they are defined within a subschema using DBA-supplied navigational paths. Logical Records are created at execution time from real database records-types based on DBA-supplied navigational paths and the SELECTs passed to LRF from the server.
- Automatic System Facility (ASF) records are created by end-users with the Automatic System Facility, a menu-driven front end to LRF. This Facility generates subschemas and navigational paths based on the user's menu selection.

The Adapter for IDMS/DB uses the IDMS/DB Logical Record Facility (LRF) to retrieve and create ASF and LR records. In general, the LRF-based records can contain information from several sources (both DML created and ASF generated) that are located in several database areas. All fields, records, and areas, however, must be defined to the same subschema.

LRF-based records may be related to each other using an embedded cross-reference. Your Master File can list up to 64 related LRF-based records. Access Files can list an unlimited number of subschemas. However, the server accesses up to 16 subschemas per request.

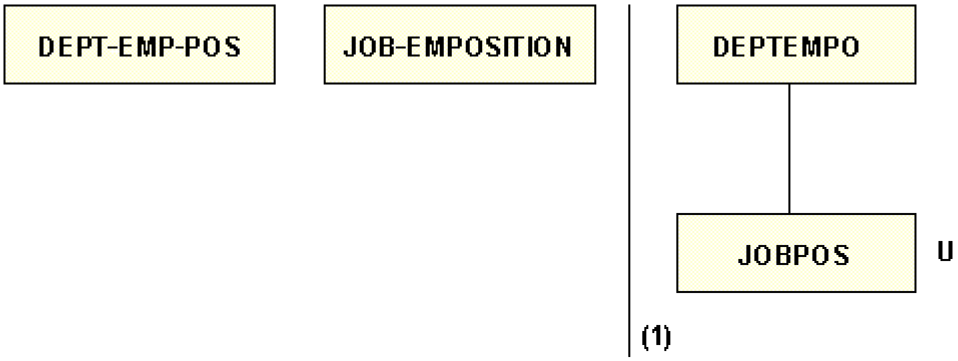
LRF Records as Descendants

How to:

Implement a Parent/Descendant Relationship With SELECT

With the Adapter for IDMS/DB, you may create a parent/descendant relationship between two LRF records if they share a common field or GROUP. Field lengths and formats must be the same in both segments. The shared fields are specified in the descendant segment declaration of the Access File as the values of the KEYFLD/IXFLD pair.

For example, in the graphic below, the LRF record DEPT-EMP-POS is represented as the root segment DEPTEMPO; the LRF record JOB-EMPOSITION as the unique segment JOBPOS. The related fields are POS_STRT_DR2 (the IXFLD value) and POS_STRT_DT1 (the KEYFLD value).



The restriction that the shared field in the descendant record be a CALC or an indexed field does not apply to LRF records. However, this kind of embedded cross-reference has one restriction: the IDMS/DB path group for an LRF record that acts as the descendant segment must contain a SELECT clause to process the implied WHERE clauses.

Syntax: How to Implement a Parent/Descendant Relationship With SELECT

SELECT clauses are maintained by your DBA and should already be available in your subschema. The following SELECT clauses can successfully implement a parent/descendant relationship. They are listed in descending order of efficiency:

```
1. SELECT FOR FIELDNAME EQ fieldname
2. SELECT USING INDEX indexname
3. SELECT FOR FIELDNAME fieldname
4. SELECT FOR ELEMENT elementname
5. SELECT
```

where:

`fieldname`

Is the joined-to IDMS/DB field name on the descendant record type.

`indexname`

Is the index set name to the joined-to field.

`elementname`

Is the physical record type on which the joined-to field is stored.

Note:

- If no specific SELECT clause exists, a null SELECT clause (item 5) must be available to process an answer set.
- LRF records that return partial record occurrences and user-defined path status should not be used. If the adapter encounters a user-defined path status, the status is interpreted as LR-ERROR and the retrieval process is halted with messages EDA967 and EDA949.
- Do not confuse the use of the word SELECT with the SQL SELECT statement. In the above example, the word SELECT is used to refer to the IDMS/DB internal meaning.

Summary of Network Relationships

As illustrated in the previous examples, there are four ways to implement a network relationship. Each can be described as a parent/descendant relationship:

- Owner/member
- Member/owner
- Field or GROUP/CALC field
- Field, GROUP/SPF, or Integrated Index

All four can be intermixed in one Master File. The actual underlying connection (set, CALC, index) between parent and descendant is not apparent to end users who specify field names. In addition, a Master File can list record types from multiple subschemas, databases, and dictionaries.

In server processing, the identity of the record type behind a segment is invisible. The retrieval technique is followed in all cases as if all segments were represented by distinct records. Within IDMS/DB, currencies are maintained by storing the DBKEYs of record types and retrieving the record occurrence again, when needed.

The top-to-bottom order in which segments are defined (the chains of parent/descendant relationships) corresponds to structural relationships among the IDMS/DB record types. This top-to-bottom order is logically significant; the left-to-right order, on the other hand, is not.

Managing IDMS/DB Metadata

This section explains the rules for making an IDMS/DB data source accessible to the server using the Master File. The Access File is an extension of Master File syntax that provides information to map DML record types and LRF-based records, including record and area names and, where needed, set or field name information.

Note: This section relates to the describing of IDMS/DB data sources for retrieval in the SQL language.

Master Files for IDMS/DB

In this section:

A Master File consists of file, segment, and field declarations. Rules for declarations are:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,$). Text appearing after the comma and dollar sign is treated as a comment.
- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas separate attribute=value pairs.

The following sections summarize the syntax for each type of declaration, and then describe each attribute.

File Attributes

Master Files begin with a file declaration that names the file and describes the type of data source; in this case, an IDMS/DB data source. The file declaration has two attributes, FILENAME and SUFFIX.

Syntax: How to Identify a Master File

```
FILE[NAME]=name, SUFFIX=IDMSR [,$]
```

where:

name

Is any 1- to 8-character name. On OS/390 and z/OS, the Master File is the member name within the PDS allocated to DDNAME MASTER.

`IDMSR`

Indicates that the Adapter for IDMS/DB is required for data retrieval.

Segment Attributes

Each IDMS/DB segment described in a Master File requires a segment declaration that consists of at least two attributes, SEGNAME and SEGTYPE.

- SEGNAME
- SEGTYPE
- PARENT
- CRFILE (Cross-referenced File)
- OCCURS and POSITION

SEGNAME

The SEGNAME value assigned to DML record types and LRF records are suggestive names -- not necessarily the IDMS/DB record names. The segment name may be a maximum of eight characters and must be unique within a given Master File. If the same IDMS/DB record type is viewed in two different contexts (for example, a loop structure), two segment declarations are required.

SEGTYPE

The SEGTYPE keyword indicates whether a segment occurs once (SEGTYPE=U) or many times (SEGTYPE=S). It is used as follows:

- For a root segment, SEGTYPE has no meaning and may be omitted entirely.
- For a descendant segment, SEGTYPE values can be S or U.
- For descendant segments with set-based relationships, SEGTYPE indicates whether the segment acts as an owner or a member. If the descendant segment is the owner record type, the SEGTYPE value is U. If the descendant is the member, the SEGTYPE value is S.
- For descendant segments with CALC-based or index-based relationships, SEGTYPE values may be S or U depending on the DUPLICATES (CLCDUP or IXDUP) value.
- For descendant segments with LRF-based relationships, SEGTYPE values may be S or U.

PARENT

The PARENT keyword is required for descendant segment declarations. The PARENT value names the descendant's parent segment. This keyword is not specified for the root declaration.

CRFILE (Cross-referenced File)

The CRFILE keyword is specified only for segments that are described remotely in another Master File. The field descriptions for these segments are, in effect, copied into the Master File at execution time. Remote descriptions are discussed in *Remote Descriptions* on page 11-25.

OCCURS and POSITION

The OCCURS and POSITION attributes are specified only when the segment corresponds to an intra-record structure, such as a COBOL OCCURS or OCCURS DEPENDING clause. These keywords are described in detail in *Intra-record Structures: OCCURS Segment* on page 11-26.

Field Attributes

Each segment consists of one or more fields. The Master File need not describe all fields from a segment. The attributes are:

- FIELDNAME
- ALIAS
- USAGE
- ACTUAL
- GROUP Fields
- IDMS/DB Database Key

To describe a field in the Master File, you must specify the primary attributes FIELDNAME, ALIAS, USAGE, and ACTUAL. These attributes are discussed in this section. Note that the adapter does not support the MISSING attribute.

FIELDNAME

How to:

Describe a Field in the Master File

The FIELDNAME keyword may contain any name — not necessarily an IDMS/DB field name. It can be a maximum of 48 characters. The name that you assign must be unique, because in the server, data is referenced through field names. If the same IDMS/DB record type viewed in different contexts underlies two or more segments, different field names must be specified in separate segment descriptions.

Due to the record-oriented nature of IDMS/DB, the fields are described for each segment in the same order as they appear in the subschema record area. Bytes skipped for field alignment or for fields you want to omit must be described as dummy fields of the appropriate length.

For example, a 1-byte alphanumeric field is followed by a double-precision floating point field:

```
SEGNAME=EMPSTATUS,$
  FIELD=STATUS_CODE,SCODE ,A1,A1,$
  FIELD=           ,      ,A7,A7,$
  FIELD=GROSS_SALES,GSALES,D12.2,D8,$
```

The same name can be used for all dummy fields; in the example above, blanks are used.

You do not need to describe all fields of the record type or LRF record; only an initial set, starting with the first field and continuing up to the last field of interest. For variable-length record types, treat the fixed-length portion as one segment and the variable portion as another segment, as described with the OCCURS attribute. For information about OCCURS segments, see *Intra-record Structures: OCCURS Segment* on page 11-26.

Syntax: How to Describe a Field in the Master File

```
FIELD[NAME]=field,[ALIAS=]alias,[USAGE=]display,[ACTUAL=]format ,$
```

where:

field

Is a 1- to 48-character field name.

alias

The ALIAS keyword is used for certain fields which require specific ALIAS values.

display

Is the display format for the field.

format

Is the definition of the IDMS/DB field format and length (n).

You can omit the ALIAS, USAGE, and ACTUAL keywords from the field declaration if the values are specified in the standard order (FIELD, ALIAS, USAGE, ACTUAL). For example, the following declarations are equivalent:

```
FIELD = YEAR, ALIAS=, USAGE=A2, ACTUAL=A2,$
FIELD = YEAR, ,A2, A2,$
```

ALIAS

The ALIAS keyword is used for certain fields which require specific ALIAS values:

- Database key fields -- the ALIAS value is DBKEY.
- ORDER fields -- the ALIAS value is ORDER.
- Fields on LRF-based records (LR or ASF) that correspond to the last fields of their underlying physical record types. Filler fields may be required if the last field does not end on a double-word boundary. The ALIAS for each filler field is a unique name with a maximum of eight characters counting the .END suffix. The adapter uses this suffix to correctly address LRF records for LRF calls.
- GROUP fields -- an ALIAS is always required or a message results.

The following is an example of a filler field in a segment that describes an LRF record. Suppose a record called JOB-EMPOSITION is defined to the IDMS/DB subschema with seven fields: the first three fields are derived from a physical (DML) record type called JOB; the last four fields, from the EMPOSITION physical record type. The Master File syntax for this LRF record looks like this:

```
FILENAME=JOBMAST, SUFFIX=IDMSR,$
  SEGNAME=JOBEMP, SEGTYPE=S,$
    FIELDNAME=JOBID       ,ALIAS=         ,USAGE=A4   ,ACTUAL=A4   ,$
    FIELDNAME=TITLE       ,ALIAS=         ,USAGE=A20  ,ACTUAL=A20  ,$
    FIELDNAME=DESCRIPTN   ,ALIAS=CMTS.END ,USAGE=A120 ,ACTUAL=A120,$
    FIELDNAME=START_DTE   ,ALIAS=         ,USAGE=A6YMD,ACTUAL=A6   ,$
    FIELDNAME=FINISH_DTE  ,ALIAS=         ,USAGE=A6YMD,ACTUAL=A6   ,$
    FIELDNAME=SALARY_GRADE,ALIAS=         ,USAGE=P4   ,ACTUAL=Z2   ,$
    FIELDNAME=SALARY      ,ALIAS=         ,USAGE=P10.2,ACTUAL=P5   ,$
    FIELDNAME=            ,ALIAS=FILL.END ,USAGE=A1   ,ACTUAL=A8   ,$
```

In this example, the filler field corresponds to the last field of the EMPOSITION record type, because LRF records must lie on a double-word boundary. The IDMS/DB filler field is stored to make the record length a multiple of 8. IDMS/DB filler fields on physical record types underlying LRF views must become filler fields. These filler fields can have any field name or contain a blank; its ALIAS must include the .END suffix. Since the JOB record type is 144 bytes (total of ACTUAL formats), it does not need a filler; the last field DESCRIPTN only requires an alias with an .END suffix.

USAGE

Master Files require you to specify field formats and lengths for USAGE and ACTUAL parameters.

The USAGE and ACTUAL keywords describe the data format of the field. The USAGE parameter defines the field length for the fields. Allow for the maximum possible number of characters or digits including decimal points. You may include valid edit options without increasing the length size.

Note: For network record types, the USAGE and ACTUAL formats of zoned CALC and zoned index fields must be described as alphanumeric (A).

ACTUAL

The ACTUAL parameter defines the field length for COBOL fields found in the IDMS/DB file. The number of internal storage bytes used by COBOL determines the field's actual length for these formats:

Alphanumeric (A)	Equals the number of characters described in the PICTURE clause.
Zoned decimal (Z)	Equals the number of characters described in the PICTURE clause.
Integer (I)	Equals 2 or 4, corresponding to decimal lengths of 1-4 or 5-9 in the PICTURE clause.
Floating-point (F)	Equals 4 bytes.
Double-precision (D)	Equals 8 bytes.
Packed decimal (P)	Equals (number of PICTURE digits / 2) + 1; excluding sign (S) or implied decimal (V).

Use the following chart as a guide for describing ACTUAL formats:

COBOL Format	COBOL PICTURE	Bytes of Storage	ACTUAL Format	USAGE Format
DISPLAY	X(4)	4	A4	A4
DISPLAY	S99	2	Z2	P3
DISPLAY	9(5)V9	6	Z6.1	P8.1
DISPLAY	99	2	A2	A2
COMP	S9	4	I2	I1
COMP	S9(4)	4	I2	I4
COMP	S9(5)	4	I4	I5
COMP	S9(9)	4	I4	I9
COMP-1	-	4	F4	F6
COMP-2	-	8	D8	D15
COMP-3	9	1	P1	P1

COBOL Format	COBOL PICTURE	Bytes of Storage	ACTUAL Format	USAGE Format
COMP-3	S9V99	2	P2	P5.2
COMP-3	9(4)V9(3)	4	P4	P8.3
FIXED BINARY(7) (COMP-4)	B or XL1	4	I4	I7

Note:

- For COMP-1 and COMP-2, allow for the maximum possible digits.
- For COBOL DISPLAY fields with zoned decimal, server formats must be packed (P).
- For COMP-1 and COMP-2, PICTURE clauses are not permitted for internal floating-point formats (F and D).

GROUP Fields

The GROUP keyword identifies a set of fields following it with a single name. This GROUP name is any unique name up to 48 characters in length. Its usage is similar to that of a COBOL group name. Generally, this attribute is used for IDMS/DB indexed or CALC fields.

Note:

- USAGE and ACTUAL format types for a GROUP field are always alphanumeric (A).
- USAGE and ACTUAL format lengths for a GROUP field are the sums of the field lengths that form the GROUP field.

For example, a GROUP field called EMP_NAME is composed of two fields, FIRST_NAME and LAST_NAME. Notice the USAGE and ACTUAL formats.

```
GROUP=EMP_NAME    ,ALIAS=  ,USAGE=A25  ,ACTUAL=A25 ,$
  FIELD=FIRST_NAME,ALIAS=  ,USAGE=A10  ,ACTUAL=A10 ,$
  FIELD=LAST_NAME ,ALIAS=  ,USAGE=A15  ,ACTUAL=A15 ,$
```

The FIELDTYPE keyword identifies indexed fields, both SPF and Integrated, on network record types. Omit it for LRF records.

IDMS/DB Database Key

The database key of a network record type corresponding to a segment can optionally be described as the last field in the segment. To specify a database key, use the following values:

Keyword	Description
FIELDNAME	Is any unique key. It can be a maximum of 48 characters.
ALIAS	Value is DBKEY.
USAGE	Value is I10.
ACTUAL	Value is I4.

The database key is used to select records from entry segments when screening on particular IDMS/DB values in the user request.

Remote Descriptions

SEGTYPEs KLU and KL specify unique and non-unique segments for which the aggregate fields are remotely described in another Master File. In other words, a KLU or KL segment contains no field information; its fields are fully defined in another Master File. The CRFILE keyword specifies the source or remote Master File. At execution time, the remote field descriptions are, in effect, copied into the current Master File.

Generally, KLU and KL segment types are used when several Master Files are constructed for the same IDMS/DB subschema. They save typing and maintenance effort. Use KLU if the underlying segment is unique (U); KL if the underlying segment is non-unique (S). Remote descriptions have no logical implications regarding parent/descendant relationships, nor do they impact their implementation.

To specify a remote description, you must give the KLU or KL segment the same name as the source segment in the remote Master File to ensure the proper use of field descriptions. The source file is specified as the CRFILE value. For example:

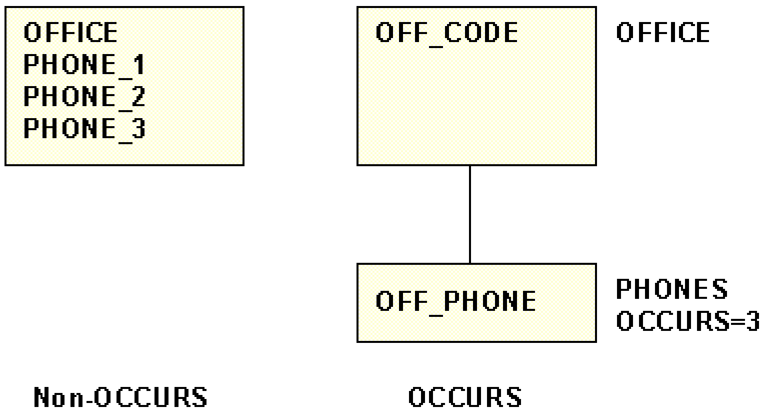
```
SEGMENT=SALES,PARENT=DEPT,SEGTYPE=KL,CRFILE=RECORDS,$
```

This segment declaration indicates that the field descriptions for the SALES segment are obtained from the SALES segment in the RECORDS Master File. The SEGTYPE for the SALES segment in the RECORDS Master File cannot be KL or KLU—only U or S. Only field names and their attributes from the source file are used; original segment attributes are not. The source file does not have to be a file description as long as it describes the named segment.

Intra-record Structures: OCCURS Segment

A common record structure is one where a field or group of adjacent fields is repeated in the same record type. In COBOL and PL/I syntax, repeating intra-record structures are defined through an OCCURS clause. The server equivalent of an OCCURS clause is an OCCURS segment.

For example, the OFFICE record type contains the field OFFICE-PHONE that occurs three times. The corresponding OFFICE segment can list three separate fields with different field names, or alternately OFFICE-PHONE can be described in a separate descendant OCCURS segment. Using the OCCURS method, each office phone is referenced by a single name (see the Master File for EMPSS01). The graphic below depicts the non-OCCURS method and the OCCURS method.



OCCURS segments have two attributes or keywords: ORDER and POSITION. The POSITION keyword directs the server to OCCURS segments when non-repeating fields exist between repeating fields. The ORDER keyword creates a fictitious count field that may be decoded.

Describing the Repeating Group to the Server

Any fixed or variable-length record type described in COBOL can be mapped into a server hierarchy using OCCURS segments. A simple OCCURS segment is a descendant of the parent segment which contains non-repeating fields found in the IDMS/DB record type. You must specify the OCCURS keyword on the descendant segment declaration that describes the repeating group. Like the COBOL OCCURS clause, the value of this keyword may be a numeric constant or a field name. The numeric constant indicates a fixed number of repetitions; a field name indicates a count field in the parent segment that maintains a count of the number of occurrences.

OCCURS segments also describe parallel and nested intra-record hierarchical structures. Parallel sets of repeating groups are described as multiple descendant segments of the same parent. In a nested structure, where a repeating group contains another repeating group, one OCCURS segment is the parent of another. Fixed and variable OCCURS segments can be intermixed in any order.

The restrictions for OCCURS segments are as follows:

- The count field for a variably-occurring repeating group must be located physically before the repeating group in the parent of the OCCURS segment.
- A record structure that has a variable number of occurrences but no count field is not within the scope of IDMS/DB.
- The SEGTYPE for an OCCURS segment is specified as S or KL, indicating that it is a non-unique segment.
- OCCURS segments must be defined in the Master File in the same order as they appear in the actual record type, unless the POSITION attribute is used.
- OCCURS segments do not have a corresponding segment declaration in the Access File, because the Adapter for IDMS/DB simulates them using the parent record. OCCURS segments do not generate any additional IDMS/DB calls.

From the server standpoint, OCCURS segments are indistinguishable from other segments. The server processes them, if referenced, in the usual top-to-bottom left-to-right retrieval order.

For example, consider this COBOL budget record type:

```
01  BUDGET-RCD.
02  ACCOUNT             PIC XXX.
02  ACTUAL-COUNT        PIC 99.
02  PLANNED-AMT         PIC 9(9) OCCURS 12 TIMES.
02  ACTUAL-AMT          PIC 9(9) OCCURS 12 TIMES
                        DEPENDING ON ACTUAL-COUNT.
```

The equivalent Master File is:

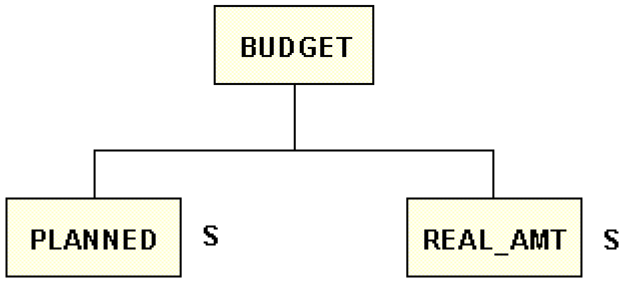
```
SEGMENT=BUDGET,SEGTYPE=S,$
   FIELD=ACCOUNT      ,ALIAS=   ,USAGE=A3  ,ACTUAL=A3       ,$
   FIELD=ACTUAL_COUNT ,ALIAS=   ,USAGE=P4  ,ACTUAL=Z2       ,$

SEGMENT=PLANNED ,PARENT=BUDGET,SEGTYPE=S,OCCURS=12          ,$
   FIELD=PLANNED_AMT  ,ALIAS=   ,USAGE=P12 ,ACTUAL=Z9       ,$

SEGMENT=REAL_AMT,PARENT=BUDGET,SEGTYPE=S,OCCURS=ACTUAL_COUNT,$
   FIELD=ACTUAL_AMT   ,ALIAS=   ,USAGE=P12 ,ACTUAL=Z9       ,$
```

The mapping principle is very simple: the non-repeating field in the record type is described in one parent segment, and the parallel COBOL OCCURS structures are made into descendant OCCURS segments. The OCCURS keyword on the descendant segments specifies either a number (fixed occurrences) or a count field (variable occurrences). In this case, the count field ACTUAL_COUNT is located in the immediate parent segment called BUDGET and is specified on the REAL_AMT descendant segment.

The following is a diagram for this Master File example:



If the PLANNED or REAL_AMT segments had repeating structures, they would in turn be parents of OCCURS segments defined by the same principles. The BUDGET segment could have other non-OCCURS descendants. The PLANNED and REAL_AMT segments might have CALC- or index-based descendants. However, set-based descendants of this record type would be tied to the BUDGET segment— not to the PLANNED or REAL_AMT segments.

POSITION

OCCURS segments are defined in the same order as they appear in the actual record type. In some cases, COBOL OCCURS clauses are separated by non-repeating fields. The POSITION keyword in a Master File indicates that the repeating fields are located in the middle of non-repeating fields.

The POSITION attribute can only be used for a repeating group with a fixed number of occurrences. This means that the OCCURS attribute of the descendant segment must equal a numeric constant and not a count field.

Suppose the previous COBOL record type looked like this:

```
01   BUDGET-RCD.
02   ACCOUNT           PIC XXX.
02   PLANNED-AMT       PIC 9(9) OCCURS 12 TIMES.
02   ACTUAL-COUNT      PIC 99.
02   ACTUAL-AMT        PIC 9(9) OCCURS 12 TIMES
                       DEPENDING ON ACTUAL-COUNT.
```

Here, the two repeating structures PLANNED-AMT and ACTUAL-AMT are separated by the non-repeating field ACTUAL-COUNT, which clearly belongs to the BUDGET segment. You must indicate in the Master File that the first occurrence of the PLANNED segment will not immediately follow the ACCOUNT field in the BUDGET segment (the PLANNED-AMT field is described in a separate descendant segment). The POSITION keyword accomplishes this task by directing the server to the descendant segment named PLANNED.

The corresponding Master File looks like this:

```
SEGMENT=BUDGET,SEGTYPE=S,$
  FIELD=ACCOUNT      ,ALIAS=    ,USAGE=A3   ,ACTUAL=A3        ,$
  FIELD=PLANNED_SEG1 ,ALIAS=    ,USAGE=A108 ,ACTUAL=A108      ,$
  FIELD=ACTUAL_COUNT ,ALIAS=    ,USAGE=P4   ,ACTUAL=Z2        ,$

SEGMENT=PLANNED ,PARENT=BUDGET,SEGTYPE=S,OCCURS=12,
        POSITION=PLANNED_SEG1                                 ,$
  FIELD=PLANNED_AMT  ,ALIAS=    ,USAGE=P12  ,ACTUAL=Z9        ,$

SEGMENT=REAL_AMT,PARENT=BUDGET,SEGTYPE=S,OCCURS=ACTUAL_COUNT ,$
  FIELD=ACTUAL_AMT   ,ALIAS=    ,USAGE=P12  ,ACTUAL=Z9        ,$
```

The POSITION keyword in the PLANNED segment names a field called PLANNED_SEG1 in BUDGET, the immediate parent segment. PLANNED_SEG1 in the parent coincides with the first field of the first occurrence in PLANNED, the OCCURS segment. The REAL_AMT segment does not require a POSITION keyword, because the position of its first occurrence is correctly inferred as following the last described field in the BUDGET segment.

In the previous example, the PLANNED_SEG1 spans all occurrences of the PLANNED segment. As an alternative, 12 individual fields named PLANNED_SEG1 through PLANNED_SEG12 could be described instead in the BUDGET segment. Each individual field would need the appropriate numeric format. Then you could refer to any one of the 12 amount fields by its specific name, or generically through the PLANNED_AMT field. The POSITION attribute can always be used for this purpose, even when it is not required for positioning the first position of an OCCURS segment located in the middle of the record type.

ORDER Field

Sometimes the sequence of fields within an OCCURS segment is significant. For example, each instance of the repeating field may represent one quarter of the year, but the segment may not have a field that specifies the quarter to which it applies.

ORDER is an optional counter used to identify the sequence number within a group of repeating fields. Specify it when the order of data is important. The ORDER field does not represent an existing field in the data source; it is used only for internal processing.

Syntax: How to Identify an ORDER Field

The ORDER field must be the last field described in the OCCURS segment.

```
FIELDNAME=name, ALIAS=ORDER, USAGE=In, ACTUAL=I4 ,$
```

where:

name

Is any valid field name.

In

Is an integer format.

Note:

- The ALIAS value must be ORDER.
- The ACTUAL format must be I4.
- The ORDER field must be the last field defined in the OCCURS segment.

Example: Using the ORDER Field in Requests

In requests, you can use the value of the ORDER field. You can also specify a DEFINE statement in the Master File to translate it to more meaningful values. For example:

```
DEFINE QTR/A3 = DECODE ORDER(1 '1ST' 2 '2ND' 3 '3RD' 4 '4TH');
```

A subsequent request could include

```
SELECT TOT.TAXES FROM JOBMAST
WHERE QTR = 1
```

or

```
SELECT QTR,BALANCE,INTEREST
```

Access Files for IDMS/DB

In this section:

An Access File describes the database access information, for example, database number, file number, and descriptor fields.

Note that before you can access a CA-IDMS/DB file, you must first describe it to the server in two files: a Master File, and an associated Access File. A Master File is required to describe the CA-IDMS/DB file in standard server terminology to the Adapter for IDMS/DB. For more information on the Master File, see *Managing IDMS/DB Metadata* on page 11-17.

An Access File is required to translate a request for network record types and LRF records into the appropriate CA-IDMS/DB Data Management Language (DML) retrieval commands. This file description consists of 80-character records called declarations in comma-delimited format (keyword=value).

An Access File contains three kinds of declarations:

- Subschema
- Segment
- Index

Subschema declarations identify the subschema used, the IDMS/DB release under which the subschema was compiled, the calling mode to be used to retrieve records, and whether a trace is to be produced. Several subschema declarations may be specified in a single Access File. Each subschema declaration is followed by its segment and index declarations. *Subschema Declaration Keywords* on page 11-33 describes keywords for subschema declarations.

Access File segment declarations indicate the IDMS/DB record information and the parent/descendant relationship for each network record type or LRF record described as a segment in a Master File (Access File segment declarations are not defined for OCCURS segments). Segment declarations can be specified in any order after their corresponding subschema declaration. *Segment Declaration Keywords for Network Record Types* on page 11-34 and *Segment Declaration Keywords for LRF Records* on page 11-37 describe segment declarations for network record types and LRF records, respectively.

Index declarations provide information about each IDMS/DB index. They may also be in any order—one for each indexed field described in the Master File—after their corresponding subschema declaration. *Index Declaration Keywords for Network Record Types* on page 11-39 describes index declarations for network record types.

Access File Syntax

Each declaration consists of a list of keyword and value pairs, separated by commas. The list is free-form and may span several lines; keywords may be specified in any order. Each declaration is completed with a comma followed by a dollar sign ($). For example, this Access File contains three declarations:

```
SSCHEMA=PAYROLL,RELEASE=15,INDEXAREA=PRIMARY-IX-AREA,
    DBNAME=EMPDEMO,DICTNAME=APPLDICT,$

SEGNAME=ACCOUNT,RECORD=PAYREC,AREA=PAY-REGION,
    CLCFLD=EMPLOYEE_ID,CLCDUP=N,$

IXSET=IXREC-SSN,IXAREA=IX-AREA1,IXFLD=PERS_SSN,
    IXDUP=N,IXORD=A,$
```

Blank lines and lines starting with an asterisk (*) in column 1 are treated as comments. Leading and trailing blanks around keywords and values are ignored. Values that contain commas, equal signs, dollar signs, or spaces must be enclosed in single quotation marks.

Subschema Declaration Keywords

If your Master File defines record types and LRF records from multiple IDMS/DB subschemas, the Access File should contain multiple subschema declarations. After each subschema declaration, list its segment and index declarations.

Subschema declarations for DML and LRF subschemas contain the following keywords; certain keywords are optional as explained in the following summary chart.

Keyword	Description
`SSCHEMA`	IDMS/DB subschema name.
`RELEASE`	Release of the IDMS/DB software that was used in the last compilation of the subschema. The value can be 10.2, 12, or 12.*x* (where *x* is your release version), 14, or 15.
`MODE`	Type of IDMS/DB access the adapter is to perform. Specify LR for LR and ASF records; DML is the default value.
`TRACE`	Optional keyword used for debugging purposes. It specifies whether a basic trace of all IDMS/DB calls or a detailed trace of all the parameters passed to IDMS/DB will be displayed. Values can be YES, PARMS, or NO (NO is the default value).
`READY`	Optional keyword that specifies when an LRF record is built from two or more physical record types located in several database areas. The adapter prepares or readies all the areas of the subschema. Values can be ALL or null or omitted entirely.
`DBNAME`	IDMS/DB database name from the DBNAME table corresponding to its subschema. It can be used in local or Central Version mode to translate the subschema name into the proper load module(s) for data access.
`INDEXAREA`	Name of the primary SPF index area. Required for any subschema with SPF indices.

CV Mode Only

Keyword	Description
`DICTNAME`	Secondary dictionary load area, if the subschema is not located in the primary dictionary or in a load PDS. Remember that ASF subschemas are located in secondary dictionary load areas by default; so if your Access File describes an ASF record, you must specify this keyword.
`NODE`	Distributed Database Systems (DDS) node location of an IDMS/DB distributed database. The value is the IDMS/DB data dictionary table entry that identifies the DDS node location of an IDMS/DB distributed database. This keyword is required only if DDS is installed at a user site and if the subschema is located in a remote site location.
`DICTNODE`	DDS node location of an IDMS/DB distributed database subschema in a secondary dictionary load area. The value is the IDMS/DB data dictionary table entry that identifies the DDS node location of an IDMS/DB distributed database subschema. This keyword is required only if DDS is installed at the user site and if the subschema is located in a remote site location.

Note: When running using DDS, Central Version must be used. DDS access is not supported in local mode.

Segment Declaration Keywords for Network Record Types

Example:

Describing One Subschema With Two Segments

Describing Two Subschemas

Your use of keywords in segment declarations for DML record types depends on whether the record type contains a CALC key, acts as a descendant segment, or contains an index. The following keywords are common to all segment declarations.

Keyword	Description
`SEGNAM`	Corresponding Master File segment name of the DML record type.
`RECORD`	IDMS/DB record type name.
`AREA`	IDMS/DB area name that contains the record type.

If your record type is CALC (that is, if the record contains a CALC key) include the two keywords below. These keywords are required for all CALC record types, regardless of how their parent/descendant relationships are implemented.

Keyword	Description
`CLCFLD`	Master File field name of the CALC field.
`CLCDUP`	Indicates if the CALC field allows duplicates. The value can be Y or N.

Record types are assigned parent/descendant relationships in the server. These relationships are based on sets and CALC or index fields. Keywords for descendant segments follow. Consult your Master File to determine whether a segment is a descendant or parent.

Field	Keyword	Description
	`ACCESS`	Indicates the relationship that exists between record types. The value CLC or IX specifies an embedded cross-reference based on a CALC or indexed field. The value SET indicates a physical relationship based on a set of pointers.
`Set-Based (ACCESS=SET)`	`SETNAME`	Name of the set relating a descendant to its parent.
	`SETMBR`	Specifies whether the set membership is mandatory/automatic, mandatory/manual, optional/automatic, or optional/manual. This information is used to verify set membership at execution time. To determine the appropriate membership value, check the set label on your Bachman diagram. Values can be MA, MM, OA, or OM.
	`GETOWN`	Allows or inhibits GET OWNER calls which obtain the owner records from a member record type. If the value is Y, the adapter issues GET OWNER calls to retrieve the owner record in the set when SEGTYPE is U or KLU. If the value is N, GET OWNER calls are inhibited. Specify N only when the set has no owner pointers and long member record chains are apt to occur. When GET OWNER calls are inhibited, the owner record type cannot be a descendant of its member. In other words, if GET OWNER calls are inhibited, SEGTYPE cannot be U or KLU.

Field	Keyword	Description
	MULTMBR	Indicates whether the set, in which this record type participates, contains more than one member record type. Values can be Y or N.
	KEYFLD	Server field that sequences the set. Select a field from a parent or descendant segment as the value of KEYFLD in the descendant segment declaration.
	SETORD	A (ascending) or D (descending) for sorted set sequence. This keyword is required.
	SETDUP	Y or N if duplicates are allowed. Required for sorted sets.
CALC-Based (ACCESS=CLC)	KEYFLD	Provides the search value to read CALC and index fields in descendant segments. These search values are located in parent segment fields. Specify the parent field name for the value of KEYFLD in the descendant segment declaration. The KEYFLD keyword is especially important when the two record types in a parent/descendant relationship are from different subschemas. The record type that acts as the descendant segment is the entry point into the second subschema. It must have a CALC key (CLCFLD) or index set (SETNAME) with ACCESS=CLC or ACCESS=IX, respectively. The descendant segment declaration must also list the KEYFLD value from the parent segment in the first subschema.
Index-Based (ACCESS=IX)	KEYFLD	See above.
	SETNAME	IDMS/DB name of the index set. A corresponding index declaration is required.

Example: Describing One Subschema With Two Segments

This example shows one subschema with two segments that are CALC record types. The INVOICE record type has a set-based relationship with the CUSTOMER record type.

```
SSCHEMA=SAMPSSCH,RELEASE=15,
SEGNAM=CUSTOMER,RECORD=CUSTOMER,AREA=CUSTOMER-REGION,
 CLCFLD=CUST_NUMBER,CLCDUP=N,$
SEGNAM=INVOICE,RECORD=INVOICE,AREA=INVO-REGION,
 CLCFLD=INV_NUMBER,CLCDUP=N,ACCESS=SET,
 SETNAME=CUSTOMER-INVO,SETMBR=MA,GETOWN=Y,MULTMBR=N,$
```

Example: Describing Two Subschemas

The following example shows two subschemas. The INSURANCE-PLAN record type has a CALC-based relationship with the COVERAGE record type.

```
SSCHEMA=EMPSS01,RELEASE=15,
SEGNAM=COVERAGE,RECORD=COVERAGE,AREA=INS-DEMO-REGION,$

SSCHEMA=EMPSS03,RELEASE=15,$
SEGNAM=INSURNCE,RECORD=INSURANCE-PLAN,
 AREA=INS-DEMO-REGION,CLCFLD=INS_PLAN_CODE,CLCDUP=N,
 ACCESS=CLC,KEYFLD=COV_CODE,$
```

If your record type contains an indexed field, you may suppress area sweeps when the segment is used as a point of entry into the database. To prevent area sweeps, specify the optional keyword below on the segment declaration. Only those record instances connected to the specified index field are accessed.

Keyword	Description
SEQFIELD	Master File field name (FIELDTYPE=I) of the index.

Note: This optional keyword requires an index declaration (see *Index Declaration Keywords for Network Record Types* on page 11-39).

Segment Declaration Keywords for LRF Records

Segment declaration keywords for LR and ASF records are basically the same as those for network (DML) record types. Specified values, of course, differ.

Keyword	Description
SEGNAM	Corresponding Master File segment name of the LRF record.
RECORD	IDMS/DB name of the LRF record.
LR	Value is Y.
AREA	IDMS/DB area name that contains physical record types. Specify READY=ALL on the subschema declaration for more than one area (if fields originate from many areas).

LRF records use embedded cross-references to create parent/descendant relationships. Specify the following keywords for LRF records that act as descendant segments:

Keyword	**Description**
`ACCESS`	Value is LR.
`KEYFLD`	Master File field name found in the parent.
`IXFLD`	Master File field name found in the descendant.

The KEYFLD and IXFLD keywords are required to implement parent/descendant relationships. The KEYFLD keyword specifies a field from the parent segment that provides search values. The value of IXFLD, in turn, is a field in the descendant segment that contains equivalent values for KEYFLD. Any field may be selected for IXFLD provided that the record possesses a null SELECT clause.

Suppose your Master File for one subschema contained two LRF records that act as parent and descendant segments.

```
FILE=DEPTEMP,SUFFIX=IDMSR,$
SEGNAME=DEPTEMP,$
  .
  .
  .
 FIELD=EMP_ID       ,ALIAS=   ,USAGE=A4 ,ACTUAL=A4,$
SEGNAME=EMPJOB,  PARENT=DEPTEMP,  SEGTYPE=S,$
 FIELD=EMPLOYEE_ID ,ALIAS=   ,USAGE=A4 ,ACTUAL=A4,$
  .
  .
  .
```

The common field is EMPLOYEE_ID; its field format is the same in both segments. The EMPJOB segment is the descendant of DEPTEMP. The descendant segment declaration in the Access File below contains KEYFLD and IXFLD values.

```
SSCHEMA=EMPSS06,RELEASE=15,MODE=LR,READY=ALL
  DBNAME=EMPDEMO,DICTNAME=APPLDICT,$

SEGNAM=DEPTEMP,RECORD=DEPT-EMPLOYEE,
  AREA=EMP-DEMO-REGION,LR=Y,$

SEGNAM=EMPJOB,RECORD=EMPLOYEE-JOB,
  AREA=ORG-DEMO-REGION,LR=Y,
  ACCESS=LR,KEYFLD=EMP_ID,IXFLD=EMPLOYEE_ID,$
```

If the EMPJOB segment (EMPLOYEE-JOB record) belonged to a different subschema, the Access File example above would include another subschema declaration positioned between the two segment declarations.

Index Declaration Keywords for Network Record Types

The Adapter for IDMS/DB supports two indexing schemes: the traditional method of indexing, using the Sequential Processing Facility (SPF), and IDMS/DB Integrated Indexes. Under SPF, index entries are stored in separate index areas. As of IDMS/DB Release 10, indexes may be integrated with the Database Management System (DBMS).

The following keywords apply to declarations for both SPF and Integrated Indexes. For an example of an index declaration, see *Access File Syntax* on page 11-32.

Keyword	Description
`IXSET`	IDMS/DB setname of the index set.
`IXFLD`	Corresponding Master File field name with FIELDTYPE=I.
`IXDUP`	Indicates if duplicate index values are allowed. Value can be Y or N.
`IXORD`	Indicates the sort order of the index. Value can be A (ascending) or D (descending).
`IXAREA`	IDMS/DB area name of the index; usage varies, see below.

Note:

- This declaration is not used in LRF Access Files.
- For subschemas with SPF indexing, the IXAREA keyword is required.
- For subschemas with Integrated Indexes, the IXAREA keyword is omitted unless the index entries reside in a different area from the record type being indexed.
- A non-unique descendant with an SPF index may not have descendants (immediate or several levels removed) that are related to the same index. This is due to the lack of an IDMS/DB command, which would restore currency after it has been disturbed by the record retrieval of a descendant segment. This restriction does not apply to SPF unique descendants or Integrated Index descendants.

IDMS/DB Sample File Descriptions

In this section:

The following sections contain:

- The schema EMPSCHM.
- A network subschema for the schema EMPSCHM, plus corresponding Master and Access Files.
- An LRF subschema for the schema EMPSCHM, plus corresponding Master and Access Files.
- A sample of a NULL SELECT clause that creates an LRF partial record occurrence.
- A sample from a subschema that contains SPF indexes.

Note: Some samples are annotated to illustrate specific clauses.

Schema: EMPSCHM

This schema is the physical description of the IDMS/DB EMPSCHM database. It contains the following items:

- Area-to-file and area-to-DDNAME mapping.
- A CALC-based record.
- A VIA-based record.
- A sorted set ordered by the SKILL-LEVEL field.
- A sorted set ordered by Integrated Indexes using the SKILL-LEVEL field.
- An Integrated Index set ordered by the SKILL-NAME field.

```
ADD SCHEMA NAME IS EMPSCHM VERSION IS 1

SCHEMA DESCRIPTION IS 'EMPLOYEE DEMO DATABASE'

COMMENTS 'INSTALLATION: COMMONWEATHER CORPORATION'
    .

ADD FILE NAME IS EMPDEMO ASSIGN TO EMPDEMO
           DEVICE TYPE IS 3380
    .

       ADD FILE NAME IS INSDEMO ASSIGN TO INSDEMO
                  DEVICE TYPE IS 3380
    .

       ADD FILE NAME IS ORGDEMO ASSIGN TO ORGDEMO
                  DEVICE TYPE IS 3380
    .

       ADD AREA NAME IS EMP-DEMO-REGION
                  RANGE IS 75001 THRU 75100
                  WITHIN FILE EMPDEMO FROM 1 THRU 100
    .

       ADD AREA NAME IS ORG-DEMO-REGION
                  RANGE IS 75151 THRU 75200
                  WITHIN FILE ORGDEMO FROM 1 THRU 50
    .

       ADD AREA NAME IS INS-DEMO-REGION
                  RANGE IS 75101 THRU 75150
                  WITHIN FILE INSDEMO FROM 1 THRU 50
    .
```

```
        ADD RECORD NAME IS COVERAGE
            SHARE STRUCTURE OF RECORD COVERAGE VERSION IS 1
            RECORD ID IS 0400
            LOCATION MODE IS VIA              EMP-COVERAGE SET
            WITHIN AREA INS-DEMO-REGION
            OFFSET  2 PAGES FOR 48 PAGES
   .

        ADD RECORD NAME IS DENTAL-CLAIM
            SHARE STRUCTURE OF RECORD DENTAL-CLAIM VERSION IS 1
            RECORD ID IS 0405
            LOCATION MODE IS VIA           COVERAGE-CLAIMS SET
            WITHIN AREA INS-DEMO-REGION
            OFFSET  2 PAGES FOR 48 PAGES
            MINIMUM ROOT LENGTH IS              130 CHARACTERS
            MINIMUM FRAGMENT LENGTH IS           RECORD LENGTH
   .

        ADD RECORD NAME IS DEPARTMENT
            SHARE STRUCTURE OF RECORD DEPARTMENT VERSION IS 1
            RECORD ID IS 0410
            LOCATION MODE IS CALC           USING DEPT-ID-0410
                                        DUPLICATES NOT ALLOWED
            WITHIN AREA ORG-DEMO-REGION
            OFFSET  2 PAGES FOR 48 PAGES
   .

        ADD RECORD NAME IS EMPLOYEE
            SHARE STRUCTURE OF RECORD EMPLOYEE VERSION IS 1
            RECORD ID IS 0415
            LOCATION MODE IS CALC            USING EMP-ID-0415
                                        DUPLICATES NOT ALLOWED
            WITHIN AREA EMP-DEMO-REGION
            OFFSET  2 PAGES FOR 98 PAGES
   .

        ADD RECORD NAME IS EMPOSITION
            SHARE STRUCTURE OF RECORD EMPOSITION VERSION IS 1
            RECORD ID IS 0420
            LOCATION MODE IS VIA            EMP-EMPOSITION SET
            WITHIN AREA EMP-DEMO-REGION
            OFFSET  2 PAGES FOR 98 PAGES
   .

        ADD RECORD NAME IS EXPERTISE
            SHARE STRUCTURE OF RECORD EXPERTISE VERSION IS 1
            RECORD ID IS 0425
            LOCATION MODE IS VIA             EMP-EXPERTISE SET
```

```
          WITHIN AREA EMP-DEMO-REGION
          OFFSET  2 PAGES FOR 98 PAGES
 .

     ADD RECORD NAME IS HOSPITAL-CLAIM
          SHARE STRUCTURE OF RECORD HOSPITAL-CLAIM VERSION IS 1
          RECORD ID IS 0430
          LOCATION MODE IS VIA          COVERAGE-CLAIMS SET
          WITHIN AREA INS-DEMO-REGION
          OFFSET  2 PAGES FOR 48 PAGES
 .

     ADD RECORD NAME IS INSURANCE-PLAN
          SHARE STRUCTURE OF RECORD INSURANCE-PLAN VERSION IS 1|
          RECORD ID IS 0435
          LOCATION MODE IS CALC    USING INS-PLAN-CODE-0435
                                     DUPLICATES NOT ALLOWED
          WITHIN AREA INS-DEMO-REGION
          OFFSET  1 PAGE  FOR  1 PAGE
 .

     ADD RECORD NAME IS JOB
          SHARE STRUCTURE OF RECORD JOB VERSION IS 1
          RECORD ID IS 0440
          LOCATION MODE IS CALC           USING JOB-ID-0440
                                     DUPLICATES NOT ALLOWED
          WITHIN AREA ORG-DEMO-REGION
          OFFSET  2 PAGES FOR 48 PAGES
          MINIMUM ROOT LENGTH IS CONTROL LENGTH
          MINIMUM FRAGMENT LENGTH IS RECORD LENGTH
          CALL IDMSCOMP BEFORE STORE
          CALL IDMSCOMP BEFORE MODIFY
          CALL IDMSDCOM AFTER GET
 .

     ADD RECORD NAME IS NON-HOSP-CLAIM
          SHARE STRUCTURE OF RECORD NON-HOSP-CLAIM VERSION IS 1
          RECORD ID IS 0445
          LOCATION MODE IS VIA          COVERAGE-CLAIMS SET
          WITHIN AREA INS-DEMO-REGION
          OFFSET 2 PAGES FOR 48 PAGES
          MINIMUM ROOT LENGTH IS             248 CHARACTERS
          MINIMUM FRAGMENT LENGTH IS          RECORD LENGTH
 .

     ADD RECORD NAME IS OFFICE
          SHARE STRUCTURE OF RECORD OFFICE VERSION IS 1
          RECORD ID IS 0450
```

```
            LOCATION MODE IS CALC      USING OFFICE-CODE-0450
                                       DUPLICATES NOT ALLOWED
            WITHIN AREA ORG-DEMO-REGION
            OFFSET 2 PAGES FOR 48 PAGES
    .

        ADD RECORD NAME IS SKILL
            SHARE STRUCTURE OF RECORD SKILL VERSION IS 1
            RECORD ID IS 0455
            LOCATION MODE IS CALC         USING SKILL-ID-0455
                                       DUPLICATES NOT ALLOWED
            WITHIN AREA ORG-DEMO-REGION
            OFFSET 2 PAGES FOR 48 PAGES
    .

        ADD RECORD NAME IS STRUCTURE
            SHARE STRUCTURE OF RECORD STRUCTURE VERSION IS 1
            RECORD ID IS 0460
            LOCATION MODE IS VIA                   MANAGES SET
            WITHIN AREA EMP-DEMO-REGION
            OFFSET 2 PAGES FOR 98 PAGES
    .

        ADD SET NAME IS COVERAGE-CLAIMS
            ORDER IS LAST
            MODE IS CHAIN LINKED TO PRIOR
            OWNER IS COVERAGE
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
            MEMBER IS HOSPITAL-CLAIM
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
                MANDATORY AUTOMATIC
            MEMBER IS NON-HOSP-CLAIM
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
                MANDATORY AUTOMATIC
            MEMBER IS DENTAL-CLAIM
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
                MANDATORY AUTOMATIC
    .

        ADD SET NAME IS DEPT-EMPLOYEE
            ORDER IS SORTED
            MODE IS CHAIN LINKED TO PRIOR
            OWNER IS DEPARTMENT
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
            MEMBER IS EMPLOYEE
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
                LINKED TO OWNER
                    OWNER DBKEY POSITION IS AUTO
                OPTIONAL AUTOMATIC
                ASCENDING KEY IS ( EMP-LAST-NAME-0415
                                   EMP-FIRST-NAME-0415 )
                    DUPLICATES LAST
     .

        ADD SET NAME IS EMP-COVERAGE
            ORDER IS FIRST
            MODE IS CHAIN LINKED TO PRIOR
            OWNER IS EMPLOYEE
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
            MEMBER IS COVERAGE
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
                LINKED TO OWNER
                    OWNER DBKEY POSITION IS AUTO
                MANDATORY AUTOMATIC
     .

        ADD SET NAME IS EMP-EMPOSITION
            ORDER IS FIRST
            MODE IS CHAIN LINKED TO PRIOR
            OWNER IS EMPLOYEE
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
            MEMBER IS EMPOSITION
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
                LINKED TO OWNER
                    OWNER DBKEY POSITION IS AUTO
                MANDATORY AUTOMATIC
     .

        ADD SET NAME IS EMP-EXPERTISE
            ORDER IS SORTED
            MODE IS CHAIN LINKED TO PRIOR
            OWNER IS EMPLOYEE
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
            MEMBER IS EXPERTISE
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
                LINKED TO OWNER
                    OWNER DBKEY POSITION IS AUTO
                MANDATORY AUTOMATIC
                DESCENDING KEY IS (SKILL-LEVEL-0425 )
                    DUPLICATES FIRST
     .

        ADD SET NAME IS EMP-NAME-NDX
            ORDER IS SORTED
            MODE IS INDEX BLOCK CONTAINS 40 KEYS
            OWNER IS SYSTEM
                WITHIN AREA EMP-DEMO-REGION
                OFFSET  1 PAGE  FOR  1 PAGE
            MEMBER IS EMPLOYEE
                INDEX DBKEY POSITION IS AUTO
                OPTIONAL AUTOMATIC
                ASCENDING KEY IS ( EMP-LAST-NAME-0415
                                   EMP-FIRST-NAME-0415 )
                    COMPRESSED
                    DUPLICATES LAST
     .

        ADD SET NAME IS JOB-EMPOSITION
            ORDER IS NEXT
            MODE IS CHAIN LINKED TO PRIOR
            OWNER IS JOB
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
            MEMBER IS EMPOSITION
                NEXT DBKEY POSITION IS AUTO
                PRIOR DBKEY POSITION IS AUTO
                LINKED TO OWNER
                    OWNER DBKEY POSITION IS AUTO
                OPTIONAL MANUAL
     .

        ADD SET NAME IS JOB-TITLE-NDX
            ORDER IS SORTED
            MODE IS INDEX BLOCK CONTAINS 30 KEYS
            OWNER IS SYSTEM
                WITHIN AREA ORG-DEMO-REGION
                OFFSET  1 PAGE  FOR  1 PAGE
            MEMBER IS JOB
                INDEX DBKEY POSITION IS AUTO
                OPTIONAL AUTOMATIC
                ASCENDING KEY IS ( TITLE-0440 )
                    DUPLICATES NOT ALLOWED
```

```
.

   ADD SET NAME IS MANAGES
       ORDER IS NEXT
       MODE IS CHAIN LINKED TO PRIOR
       OWNER IS EMPLOYEE
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
       MEMBER IS STRUCTURE
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
           LINKED TO OWNER
               OWNER DBKEY POSITION IS AUTO
           MANDATORY AUTOMATIC
.

   ADD SET NAME IS OFFICE-EMPLOYEE
       ORDER IS SORTED
       MODE IS INDEX BLOCK CONTAINS 30 KEYS
       OWNER IS OFFICE
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
       MEMBER IS EMPLOYEE
           INDEX DBKEY POSITION IS AUTO
           LINKED TO OWNER
               OWNER DBKEY POSITION IS AUTO
           OPTIONAL AUTOMATIC
           ASCENDING KEY IS ( EMP-LAST-NAME-0415
                              EMP-FIRST-NAME-0415 )
               COMPRESSED
               DUPLICATES LAST
.

   ADD SET NAME IS REPORTS-TO
       ORDER IS NEXT
       MODE IS CHAIN LINKED TO PRIOR
       OWNER IS EMPLOYE
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
       MEMBER IS STRUCTURE
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
           LINKED TO OWNER
               OWNER DBKEY POSITION IS AUTO
           OPTIONAL MANUAL
.

   ADD SET NAME IS SKILL-EXPERTISE
```

```
        ORDER IS SORTED
        MODE IS INDEX BLOCK CONTAINS 30 KEYS
        OWNER IS SKILL
            NEXT DBKEY POSITION IS AUTO
            PRIOR DBKEY POSITION IS AUTO
        MEMBER IS EXPERTISE
            INDEX DBKEY POSITION IS AUTO
            LINKED TO OWNER
                OWNER DBKEY POSITION IS AUTO
            MANDATORY AUTOMATIC
            DESCENDING KEY IS ( SKILL-LEVEL-0425 )
                DUPLICATES FIRST
.

    ADD SET NAME IS SKILL-NAME-NDX
        ORDER IS SORTED
        MODE IS INDEX BLOCK CONTAINS 30 KEYS
        OWNER IS SYSTEM
            WITHIN AREA ORG-DEMO-REGION
            OFFSET  1 PAGE  FOR  1 PAGE
        MEMBER IS SKILL
            INDEX DBKEY POSITION IS AUTO
            OPTIONAL AUTOMATIC
            ASCENDING KEY IS ( SKILL-NAME-0455 )
                DUPLICATES NOT ALLOWED
.

    VALIDATE
 .
```

Network Subschema: EMPSS01

This subschema shows the network view of the schema EMPSCHM.

```
ADD SUBSCHEMA NAME IS EMPSS01
  OF SCHEMA NAME IS EMPSCHM VERSION  1
DMCL NAME IS EMPDMCL
  OF SCHEMA NAME IS EMPSCHM VERSION  1
COMMENTS 'THIS IS THE COMPLETE VIEW OF EMPSCHM'.
ADD AREA NAME IS EMP-DEMO-REGION.
ADD AREA NAME IS INS-DEMO-REGION.
ADD AREA NAME IS ORG-DEMO-REGION.
ADD RECORD NAME IS COVERAGE.
ADD RECORD NAME IS DENTAL-CLAIM.
ADD RECORD NAME IS DEPARTMENT.
ADD RECORD NAME IS EMPLOYEE.
ADD RECORD NAME IS EMPOSITION.
ADD RECORD NAME IS EXPERTISE.
ADD RECORD NAME IS HOSPITAL-CLAIM.
ADD RECORD NAME IS INSURANCE-PLAN.
ADD RECORD NAME IS JOB.
ADD RECORD NAME IS NON-HOSP-CLAIM.
ADD RECORD NAME IS OFFICE.
ADD RECORD NAME IS SKILL.
ADD RECORD NAME IS STRUCTURE.
ADD SET COVERAGE-CLAIMS.
ADD SET DEPT-EMPLOYEE.
ADD SET EMP-COVERAGE.
ADD SET EMP-EXPERTISE.
ADD SET EMP-NAME-NDX.
ADD SET EMP-EMPOSITION.
ADD SET JOB-EMPOSITION.
ADD SET JOB-TITLE-NDX.
ADD SET MANAGES.
ADD SET OFFICE-EMPLOYEE.
ADD SET REPORTS-TO.
ADD SET SKILL-EXPERTISE.
ADD SET SKILL-NAME-NDX.
GENERATE.
```

Master File for Network

This Master File corresponds to the network subschema EMPSS01.

```
FILE=EMPFULL,SUFFIX=IDMSR ,$

SEGNAME=DEPT,$
  FIELDNAME=DEPT_ID      ,ALIAS=      ,USAGE=A4    ,ACTUAL=A4       ,$
  FIELDNAME=DEPT_NAME    ,ALIAS=      ,USAGE=A45   ,ACTUAL=A45      ,$
  FIELDNAME=DEPT_HEAD    ,ALIAS=      ,USAGE=A4    ,ACTUAL=A4       ,$
  FIELDNAME=DEPT_DBKEY   ,ALIAS=DBKEY ,USAGE=I10   ,ACTUAL=I4       ,$

SEGNAME=EMPLOYE,PARENT=DEPT,SEGTYPE=S,$
  FIELDNAME=EMP_ID       ,ALIAS=      ,USAGE=A4    ,ACTUAL=A4       ,$
  GROUP=EMP_NAME         ,ALIAS=      ,USAGE=A25   ,ACTUAL=A25      ,$
    FIELDNAME=FIRST_NAME,ALIAS=       ,USAGE=A10   ,ACTUAL=A10      ,$
    FIELDNAME=LAST_NAME ,ALIAS=       ,USAGE=A15   ,ACTUAL=A15      ,$
  FIELDNAME=EMP_STREET   ,ALIAS=      ,USAGE=A20   ,ACTUAL=A20      ,$
  FIELDNAME=EMP_CITY     ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15      ,$
  FIELDNAME=EMP_STATE    ,ALIAS=      ,USAGE=A2    ,ACTUAL=A2       ,$
  GROUP=EMP_FULL_ZIP     ,ALIAS=      ,USAGE=A9    ,ACTUAL=A9       ,$
    FIELDNAME=EMP_ZIP   ,ALIAS=       ,USAGE=A5    ,ACTUAL=A5       ,$
    FIELDNAME=EMP_ZIP_L ,ALIAS=       ,USAGE=A4    ,ACTUAL=A4       ,$
  FIELDNAME=EMP_PHONE    ,ALIAS=      ,USAGE=A10   ,ACTUAL=A10      ,$
  FIELDNAME=STATUS       ,ALIAS=      ,USAGE=A2    ,ACTUAL=A2       ,$
  FIELDNAME=SOC_SEC_NUM  ,ALIAS=      ,USAGE=A9    ,ACTUAL=A9       ,$
  FIELDNAME=EMP_STRT_DTE,ALIAS=       ,USAGE=A6YMD,ACTUAL=A6        ,$
  FIELDNAME=EMP_TERM_DTE,ALIAS=       ,USAGE=A6YMD,ACTUAL=A6        ,$
  FIELDNAME=EMP_BRTH_DTE,ALIAS=       ,USAGE=A6YMD,ACTUAL=A6        ,$
  FIELDNAME=EMP_DBKEY    ,ALIAS=DBKEY ,USAGE=I10   ,ACTUAL=I4       ,$

SEGNAME=OFFICE,PARENT=EMPLOYE,SEGTYPE=U,$
  FIELDNAME=OFF_CODE     ,ALIAS=      ,USAGE=A3    ,ACTUAL=A3       ,$
  FIELDNAME=OFF_STREET   ,ALIAS=      ,USAGE=A20   ,ACTUAL=A20      ,$
  FIELDNAME=OFF_CITY     ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15      ,$
  FIELDNAME=OFF_STATE    ,ALIAS=      ,USAGE=A2    ,ACTUAL=A2       ,$
  GROUP=OFF_FULL_ZIP     ,ALIAS=      ,USAGE=A9    ,ACTUAL=A9       ,$
    FIELDNAME=OFF_ZIP   ,ALIAS=       ,USAGE=A5    ,ACTUAL=A5       ,$
    FIELDNAME=OFF_ZIP_L ,ALIAS=       ,USAGE=A4    ,ACTUAL=A4       ,$
  FIELDNAME=O_PHONES     ,ALIAS=      ,USAGE=A21   ,ACTUAL=A21      ,$
  FIELDNAME=OFF_AREA_CDE,ALIAS=       ,USAGE=A3    ,ACTUAL=A3       ,$
  FIELDNAME=SPEED_DIAL   ,ALIAS=      ,USAGE=A3    ,ACTUAL=A3       ,$

SEGNAME=PHONES,PARENT=OFFICE,SEGTYPE=S,OCCURS=3,POSITION=O_PHONES ,$
  FIELDNAME=OFF_PHONE    ,ALIAS=      ,USAGE=A7    ,ACTUAL=A7       ,$
  FIELDNAME=LINE_NO      ,ALIAS=ORDER ,USAGE=I4    ,ACTUAL=I4       ,$

SEGNAME=STRUCTUR,PARENT=EMPLOYE,SEGTYPE=S,$
  FIELDNAME=STRUCTURE_CD,ALIAS=       ,USAGE=A2    ,ACTUAL=A2       ,$
  FIELDNAME=STRUCTURE_DT,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$

SEGNAME=SUBORDS,PARENT=STRUCTUR,SEGTYPE=U,$
  FIELDNAME=SUB_ID      ,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  GROUP=SUB_NAME        ,ALIAS=      ,USAGE=A25  ,ACTUAL=A25      ,$
    FIELDNAME=SUB_F_NAME,ALIAS=      ,USAGE=A10  ,ACTUAL=A10      ,$
    FIELDNAME=SUB_L_NAME,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=SUB_STREET  ,ALIAS=      ,USAGE=A20  ,ACTUAL=A20      ,$
  FIELDNAME=SUB_CITY    ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=SUB_STATE   ,ALIAS=      ,USAGE=A2   ,ACTUAL=A2       ,$
  GROUP=SUB_FULL_ZIP    ,ALIAS=      ,USAGE=A9   ,ACTUAL=A9       ,$
    FIELDNAME=SUB_ZIP   ,ALIAS=      ,USAGE=A5   ,ACTUAL=A5       ,$
    FIELDNAME=SUB_ZIP_L ,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  FIELDNAME=SUB_PHONE   ,ALIAS=      ,USAGE=A10  ,ACTUAL=A10      ,$
  FIELDNAME=SUB_STATUS  ,ALIAS=      ,USAGE=A2   ,ACTUAL=A2       ,$
  FIELDNAME=SUB_SSN     ,ALIAS=      ,USAGE=A9   ,ACTUAL=A9       ,$
  FIELDNAME=SUB_STRT_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=SUB_TERM_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=SUB_BRTH_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$

SEGNAME=EMPOSIT,PARENT=EMPLOYE,SEGTYPE=S,$
  FIELDNAME=POS_STRT_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=POS_FIN_DTE ,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=SALARY_GRADE,ALIAS=      ,USAGE=P4   ,ACTUAL=Z2       ,$
  FIELDNAME=SALARY_AMT  ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5       ,$
  FIELDNAME=BONUS_PCT   ,ALIAS=      ,USAGE=P4   ,ACTUAL=P2       ,$
  FIELDNAME=COMMIS_PCT  ,ALIAS=      ,USAGE=P4   ,ACTUAL=P2       ,$
  FIELDNAME=OVERTIME_PCT,ALIAS=      ,USAGE=P5.2 ,ACTUAL=P2       ,$

SEGNAME=JOB,PARENT=EMPOSIT,SEGTYPE=U,$
  FIELDNAME=JOB_ID      ,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  FIELDNAME=TITLE       ,ALIAS=      ,USAGE=A20  ,ACTUAL=A20,
   FIELDTYPE=I,$
    DEFINE SHORTTITLE/A10 = EDIT(JTIT,'9999999999$');             ,$
  FIELDNAME=JOB_DESC    ,ALIAS=      ,USAGE=A120 ,ACTUAL=A120     ,$
  FIELDNAME=REQUIREMENTS,ALIAS=      ,USAGE=A120 ,ACTUAL=A120     ,$
  FIELDNAME=MIN_SALARY  ,ALIAS=      ,USAGE=P12.2,ACTUAL=Z8       ,$
  FIELDNAME=MAX_SALARY  ,ALIAS=      ,USAGE=P12.2,ACTUAL=Z8       ,$
  FIELDNAME=SAL_GRADE_1 ,ALIAS=      ,USAGE=P4   ,ACTUAL=Z2       ,$
  FIELDNAME=SAL_GRADE_2 ,ALIAS=      ,USAGE=P4   ,ACTUAL=Z2       ,$
  FIELDNAME=SAL_GRADE_3 ,ALIAS=      ,USAGE=P4   ,ACTUAL=Z2       ,$
  FIELDNAME=SAL_GRADE_4 ,ALIAS=      ,USAGE=P4   ,ACTUAL=Z2       ,$
  FIELDNAME=POSITION_NUM,ALIAS=      ,USAGE=P4   ,ACTUAL=Z3       ,$
  FIELDNAME=NUM_OPEN    ,ALIAS=      ,USAGE=P4   ,ACTUAL=Z3       ,$

SEGNAME=EXPERTSE,PARENT=EMPLOYE,SEGTYPE=S,$
  FIELDNAME=SKILL_LEVEL ,ALIAS=      ,USAGE=A2   ,ACTUAL=A2       ,$
  FIELDNAME=EXPERT_DTE  ,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
```

```
SEGNAME=SKILL,PARENT=EXPERTSE,SEGTYPE=U,$
  FIELDNAME=SKILL_ID    ,ALIAS=      ,USAGE=A4    ,ACTUAL=A4        ,$
  FIELDNAME=SKILL_NAME  ,ALIAS=      ,USAGE=A12   ,ACTUAL=A12,
   FIELDTYPE=I,$
  FIELDNAME=SKILL_DESC  ,ALIAS=      ,USAGE=A60   ,ACTUAL=A60       ,$

SEGNAME=COVERAGE,PARENT=EMPLOYE,SEGTYPE=S,$
  FIELDNAME=COV_SEL_DT  ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6         ,$
  FIELDNAME=COV_TERM_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6         ,$
  FIELDNAME=COVER_TYPE  ,ALIAS=      ,USAGE=A1    ,ACTUAL=A1        ,$
  FIELDNAME=COV_CODE    ,ALIAS=      ,USAGE=A3    ,ACTUAL=A3        ,$

SEGNAME=HOSPITAL,PARENT=COVERAGE,SEGTYPE=S,$
  FIELDNAME=H_CLAIM_DTE ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6         ,$
  FIELDNAME=H_FIRST_NAME,ALIAS=      ,USAGE=A10   ,ACTUAL=A10       ,$
  FIELDNAME=H_LAST_NAME ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15       ,$
  FIELDNAME=H_BIRTH_DTE ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6         ,$
  FIELDNAME=H_SEX       ,ALIAS=      ,USAGE=A1    ,ACTUAL=A1        ,$
  FIELDNAME=H_RELATED_BY,ALIAS=      ,USAGE=A10   ,ACTUAL=A10       ,$
  FIELDNAME=HOSP_NAME   ,ALIAS=      ,USAGE=A25   ,ACTUAL=A25       ,$
  FIELDNAME=HOSP_STREET ,ALIAS=      ,USAGE=A20   ,ACTUAL=A20       ,$
  FIELDNAME=HOSP_CITY   ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15       ,$
  FIELDNAME=HOSP_STATE  ,ALIAS=      ,USAGE=A2    ,ACTUAL=A2        ,$
  GROUP=HOSP_FUL_ZIP    ,ALIAS=      ,USAGE=A9    ,ACTUAL=A9        ,$
    FIELDNAME=HOSP_ZIP  ,ALIAS=      ,USAGE=A5    ,ACTUAL=A5        ,$
    FIELDNAME=HOSP_ZIP_L,ALIAS=      ,USAGE=A4    ,ACTUAL=A4        ,$
  FIELDNAME=ADMITTED    ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6         ,$
  FIELDNAME=DISCHARGED  ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6         ,$
  FIELDNAME=H_DIAGNOSIS1,ALIAS=      ,USAGE=A60   ,ACTUAL=A60       ,$
  FIELDNAME=H_DIAGNOSIS2,ALIAS=      ,USAGE=A60   ,ACTUAL=A60       ,$
  FIELDNAME=WARD_DAYS   ,ALIAS=      ,USAGE=P5    ,ACTUAL=P3        ,$
  FIELDNAME=WARD_RATE   ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5         ,$
  FIELDNAME=WARD_TOTAL  ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5         ,$
  FIELDNAME=SEMI_DAYS   ,ALIAS=      ,USAGE=P5    ,ACTUAL=P3        ,$
  FIELDNAME=SEMI_RATE   ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5         ,$
  FIELDNAME=SEMI_TOTAL  ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5         ,$
  FIELDNAME=DELIVERY_TOT,ALIAS=      ,USAGE=P10.2,ACTUAL=P5         ,$
  FIELDNAME=ANESTHES_TOT,ALIAS=      ,USAGE=P10.2,ACTUAL=P5         ,$
  FIELDNAME=LAB_TOT     ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5         ,$
  FIELDNAME=            ,ALIAS=      ,USAGE=A4    ,ACTUAL=A4        ,$
  FIELDNAME=CLAIM_MONTH ,ALIAS=CMO   ,USAGE=I2    ,ACTUAL=Z2        ,$

SEGNAME=NON_HOSP,SEGTYPE=S,PARENT=COVERAGE,$
  FIELDNAME=N_CLAIM_DTE ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6         ,$
  FIELDNAME=N_FIRST_NAME,ALIAS=      ,USAGE=A10   ,ACTUAL=A10       ,$
  FIELDNAME=N_LAST_NAME ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15       ,$
  FIELDNAME=N_BIRTH_DTE ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6         ,$
  FIELDNAME=N_SEX       ,ALIAS=      ,USAGE=A1    ,ACTUAL=A1        ,$
  FIELDNAME=N_RELATED_BY,ALIAS=      ,USAGE=A10   ,ACTUAL=A10       ,$
  FIELDNAME=PHYS_FNAME  ,ALIAS=      ,USAGE=A10   ,ACTUAL=A10       ,$
  FIELDNAME=PHYS_LNAME  ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15       ,$
  FIELDNAME=PHYS_STREET ,ALIAS=      ,USAGE=A20   ,ACTUAL=A20       ,$
  FIELDNAME=PHYS_CITY   ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15       ,$
  FIELDNAME=PHYS_STATE  ,ALIAS=      ,USAGE=A2    ,ACTUAL=A2        ,$
  GROUP=PHYS_FUL_ZIP    ,ALIAS=      ,USAGE=A9    ,ACTUAL=A9        ,$
    FIELDNAME=PHYS_ZIP  ,ALIAS=      ,USAGE=A5    ,ACTUAL=A5        ,$
    FIELDNAME=PHYS_ZIP_L,ALIAS=      ,USAGE=A4    ,ACTUAL=A4        ,$
  FIELDNAME=PHYS_ID     ,ALIAS=      ,USAGE=P6    ,ACTUAL=Z6        ,$
  FIELDNAME=P_DIAGNOSIS1,ALIAS=      ,USAGE=A60   ,ACTUAL=A60       ,$
  FIELDNAME=P_DIAGNOSIS2,ALIAS=      ,USAGE=A60   ,ACTUAL=A60       ,$
  FIELDNAME=P_NO_OF_PROC,ALIAS=      ,USAGE=I2    ,ACTUAL=I2        ,$
  FIELDNAME=            ,ALIAS=      ,USAGE=A1    ,ACTUAL=A1        ,$

SEGNAME=PHYSCHRG,SEGTYPE=S,PARENT=NON_HOSP,OCCURS=P_NO_OF_PROC    ,$
  FIELDNAME=P_SERVICE_DT,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6         ,$
  FIELDNAME=PHYS_PROC_CD,ALIAS=      ,USAGE=P4    ,ACTUAL=Z4        ,$
  FIELDNAME=P_SERV_DESC ,ALIAS=      ,USAGE=A60   ,ACTUAL=A60       ,$
  FIELDNAME=PHYS_FEE    ,ALIAS=      ,USAGE=P11.2,ACTUAL=P5         ,$
  FIELDNAME=            ,ALIAS=      ,USAGE=A1    ,ACTUAL=A1        ,$
  FIELDNAME=PHYS_CHRG_NO,ALIAS=ORDER ,USAGE=I4    ,ACTUAL=I4        ,$

SEGNAME=DENTAL,SEGTYPE=S,PARENT=COVERAGE,$
  FIELDNAME=D_CLAIM_DTE ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6         ,$
  FIELDNAME=D_FIRST_NAME,ALIAS=      ,USAGE=A10   ,ACTUAL=A10       ,$
  FIELDNAME=D_LAST_NAME ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15       ,$
  FIELDNAME=D_BIRTH_DTE ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6         ,$
  FIELDNAME=D_SEX       ,ALIAS=      ,USAGE=A1    ,ACTUAL=A1        ,$
  FIELDNAME=D_RELATED_BY,ALIAS=      ,USAGE=A10   ,ACTUAL=A10       ,$
  FIELDNAME=DENT_FNAME  ,ALIAS=      ,USAGE=A10   ,ACTUAL=A10       ,$
  FIELDNAME=DENT_LNAME  ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15       ,$
  FIELDNAME=DENT_STREET ,ALIAS=      ,USAGE=A20   ,ACTUAL=A20       ,$
  FIELDNAME=DENT_CITY   ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15       ,$
  FIELDNAME=DENT_STATE  ,ALIAS=      ,USAGE=A2    ,ACTUAL=A2        ,$
  GROUP=DENT_FUL_ZIP    ,ALIAS=      ,USAGE=A9    ,ACTUAL=A9        ,$
    FIELDNAME=DENT_ZIP  ,ALIAS=      ,USAGE=A5    ,ACTUAL=A5        ,$
    FIELDNAME=DENT_ZIP_L,ALIAS=      ,USAGE=A4    ,ACTUAL=A4        ,$
  FIELDNAME=DENT_LICENSE,ALIAS=      ,USAGE=P6    ,ACTUAL=Z6        ,$
  FIELDNAME=D_NO_OF_PROC,ALIAS=      ,USAGE=I2    ,ACTUAL=I2        ,$
  FIELDNAME=            ,ALIAS=      ,USAGE=A3    ,ACTUAL=A3        ,$

SEGNAME=DENTCHRG,SEGTYPE=S,PARENT=DENTAL,OCCURS=D_NO_OF_PROC,$
  FIELDNAME=TOOTH_NUM   ,ALIAS=      ,USAGE=P2    ,ACTUAL=Z2        ,$
  FIELDNAME=D_SERVICE_DT,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6         ,$
  FIELDNAME=DENT_PROC_CD,ALIAS=      ,USAGE=P4    ,ACTUAL=Z4        ,$
  FIELDNAME=D_SERV_DESC ,ALIAS=      ,USAGE=A60   ,ACTUAL=A60       ,$
  FIELDNAME=DENT_FEE    ,ALIAS=      ,USAGE=P11.2,ACTUAL=P5        ,$
  FIELDNAME=            ,ALIAS=      ,USAGE=A3   ,ACTUAL=A3        ,$
  FIELDNAME=DENT_CHRG_NO,ALIAS=ORDER ,USAGE=I9   ,ACTUAL=I4        ,$

SEGNAME=INSURNCE,PARENT=COVERAGE,SEGTYPE=U,$
  FIELDNAME=INS_PLAN_CDE,ALIAS=      ,USAGE=A3   ,ACTUAL=A3        ,$
  FIELDNAME=INS_CO_NAME ,ALIAS=      ,USAGE=A45  ,ACTUAL=A45       ,$
  FIELDNAME=INS_STREET  ,ALIAS=      ,USAGE=A20  ,ACTUAL=A20       ,$
  FIELDNAME=INS_CITY    ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15       ,$
  FIELDNAME=INS_STATE   ,ALIAS=      ,USAGE=A2   ,ACTUAL=A2        ,$
  GROUP=INS_FULL_ZIP    ,ALIAS=      ,USAGE=A9   ,ACTUAL=A9        ,$
    FIELDNAME=INS_ZIP   ,ALIAS=      ,USAGE=A5   ,ACTUAL=A5        ,$
    FIELDNAME=INS_ZIP_L ,ALIAS=      ,USAGE=A4   ,ACTUAL=A4        ,$
  FIELDNAME=INS_PHONE   ,ALIAS=      ,USAGE=A10  ,ACTUAL=A10       ,$
  FIELDNAME=INS_GROUP_NO,ALIAS=      ,USAGE=A6   ,ACTUAL=A6        ,$
  FIELDNAME=DEDUCT      ,ALIAS=      ,USAGE=P12.2,ACTUAL=P5        ,$
  FIELDNAME=MAX_LIFE_CST,ALIAS=      ,USAGE=P12.2,ACTUAL=P5        ,$
  FIELDNAME=FAMILY_COST ,ALIAS=      ,USAGE=P12.2,ACTUAL=P5        ,$
  FIELDNAME=DEPENDNT_CST,ALIAS=      ,USAGE=P12.2,ACTUAL=P5        ,$
```

Access File for Network

This Access File is associated with the network subschema EMPSS01 and corresponds to its Master File.

```
SSCHEMA=EMPSS01,RELEASE=15,MODE=DML,TRACE=NO,READY=,$

SEGNAM=DEPT,RECORD=DEPARTMENT,AREA=ORG-DEMO-REGION,
  CLCFLD=DEPT_ID,CLCDUP=N,$

SEGNAM=EMPLOYE,RECORD=EMPLOYEE,AREA=EMP-DEMO-REGION,
  CLCFLD=EMP_ID,CLCDUP=N,ACCESS=SET,SETNAME=DEPT-EMPLOYEE,
  SETMBR=OA,GETOWN=Y,MULTMBR=N,$

SEGNAM=OFFICE,RECORD=OFFICE,AREA=ORG-DEMO-REGION,
  CLCFLD=OFF_CODE,CLCDUP=N,ACCESS=SET,SETNAME=OFFICE-EMPLOYEE,
  SETMBR=OA,GETOWN=Y,MULTMBR=N,$

SEGNAM=STRUCTUR,RECORD=STRUCTURE,AREA=EMP-DEMO-REGION,
  ACCESS=SET,SETNAME=MANAGES,SETMBR=MA,GETOWN=Y,MULTMBR=N,$

SEGNAM=SUBORDS,RECORD=EMPLOYEE,AREA=EMP-DEMO-REGION,
  CLCFLD=SUB_ID,CLCDUP=N,ACCESS=SET,SETNAME=REPORTS-TO,
  SETMBR=OM,GETOWN=Y,MULTMBR=N,$

SEGNAM=EMPOSIT,RECORD=EMPOSITION,AREA=EMP-DEMO-REGION,
  ACCESS=SET,SETNAME=EMP-EMPOSITION,SETMBR=MA,GETOWN=Y,MULTMBR=N,$

SEGNAM=JOB,RECORD=JOB,AREA=ORG-DEMO-REGION,
  CLCFLD=JOB_ID,CLCDUP=N,ACCESS=SET,SETNAME=JOB-EMPOSITION,
  SETMBR=OM,GETOWN=Y,MULTMBR=N,SEQFIELD=TITLE,$

SEGNAM=EXPERTSE,RECORD=EXPERTISE,AREA=EMP-DEMO-REGION,
  ACCESS=SET,SETNAME=EMP-EXPERTISE,KEYFLD=SKILL_LEVEL,SETORD=D,
  SETDUP=Y,SETMBR=MA,GETOWN=Y,MULTMBR=N,$

SEGNAM=SKILL,RECORD=SKILL,AREA=ORG-DEMO-REGION,
  CLCFLD=SKILL_ID,CLCDUP=N,ACCESS=SET,SETNAME=SKILL-EXPERTISE,
  KEYFLD=SKILL_LEVEL,SETORD=D,SETDUP=Y,
  SETMBR=MA,GETOWN=Y,MULTMBR=N,SEQFIELD=SKILL_NAME,$

SEGNAM=COVERAGE,RECORD=COVERAGE,AREA=INS-DEMO-REGION,
  ACCESS=SET,SETNAME=EMP-COVERAGE,SETMBR=MA,GETOWN=Y,MULTMBR=N,$

SEGNAM=HOSPITAL,RECORD=HOSPITAL-CLAIM,AREA=INS-DEMO-REGION,
  ACCESS=SET,SETNAME=COVERAGE-CLAIMS,SETMBR=MA,GETOWN=Y,MULTMBR=Y,$

SEGNAM=NON_HOSP,RECORD=NON-HOSP-CLAIM,AREA=INS-DEMO-REGION,
  ACCESS=SET,SETNAME=COVERAGE-CLAIMS,SETMBR=MA,GETOWN=Y,MULTMBR=Y,$

SEGNAM=DENTAL,RECORD=DENTAL-CLAIM,AREA=INS-DEMO-REGION,
  ACCESS=SET,SETNAME=COVERAGE-CLAIMS,SETMBR=MA,GETOWN=Y,MULTMBR=Y,$

IXSET=JOB-TITLE-NDX,IXFLD=TITLE,IXDUP=N,IXORD=A,
  IXAREA=INS-DEMO-REGION,$

IXSET=SKILL-NAME-NDX,IXFLD=SKILL_NAME,IXDUP=N,IXORD=D,
  IXAREA=EMP-DEMO-REGION,$

SEGNAM=INSURNCE,RECORD=INSURANCE-PLAN,AREA=INS-DEMO-REGION,
  CLCFLD=INS_PLAN_CDE,CLCDUP=N,ACCESS=CLC,KEYFLD=COV_CODE,$
```

LRF Subschema: EMPSS02

This subschema shows the LRF view of the schema EMPSCHM. It contains the following items:

- Physical record types that are used to create logical records.
- A SELECT clause for CALC access.
- A SELECT ELEMENT clause.
- A SELECT NULL clause.
- A SELECT INDEX clause.

```
ADD SUBSCHEMA NAME IS EMPSS02
    OF SCHEMA NAME IS EMPSCHM VERSION  1
    USAGE IS LR
DMCL NAME IS EMPDMCL
    OF SCHEMA NAME IS EMPSCHM VERSION  1
COMMENTS 'THIS IS THE COMPLETE VIEW OF EMPSCHM'.
ADD AREA NAME IS EMP-DEMO-REGION.
ADD AREA NAME IS ORG-DEMO-REGION.
ADD RECORD NAME IS DEPARTMENT.
ADD RECORD NAME IS EMPLOYEE.
ADD RECORD NAME IS EMPOSITION.
ADD RECORD NAME IS JOB.
ADD SET DEPT-EMPLOYEE.
ADD SET EMP-NAME-NDX.
ADD SET EMP-EMPOSITION.
ADD SET JOB-EMPOSITION.
ADD SET JOB-TITLE-NDX.
ADD
   LOGICAL RECORD NAME IS DEPT-EMP-POS
   ELEMENTS ARE DEPARTMENT
                EMPLOYEE
                EMPOSITION.
ADD
PATH-GROUP NAME IS OBTAIN DEPT-EMP-POS
   SELECT FOR FIELDNAME-EQ DEPT-ID-0410
      OBTAIN DEPARTMENT
        WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
      IF DEPT-EMPLOYEE IS NOT EMPTY
      OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
      IF EMP-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION
   SELECT FOR FIELDNAME-EQ EMP-ID-0415
      OBTAIN EMPLOYEE
        WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
      IF DEPT-EMPLOYEE MEMBER
      OBTAIN OWNER WITHIN DEPT-EMPLOYEE
      IF EMP-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION
   SELECT FOR ELEMENT DEPARTMENT
      OBTAIN EACH DEPARTMENT WITHIN ORG-DEMO-REGION
      IF DEPT-EMPLOYEE IS NOT EMPTY
      OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
      IF EMP-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION
   SELECT FOR ELEMENT EMPLOYEE
      OBTAIN EACH EMPLOYEE WITHIN EMP-DEMO-REGION
      IF DEPT-EMPLOYEE MEMBER
      OBTAIN OWNER WITHIN DEPT-EMPLOYEE
      IF EMP-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION
   SELECT FOR ELEMENT EMPOSITION
```

```
      OBTAIN EACH EMPOSITION WITHIN EMP-DEMO-REGION
      OBTAIN OWNER WITHIN EMP-EMPOSITION
      IF DEPT-EMPLOYEE MEMBER
      OBTAIN OWNER WITHIN DEPT-EMPLOYEE
   SELECT
      OBTAIN EACH DEPARTMENT WITHIN ORG-DEMO-REGION
      IF DEPT-EMPLOYEE IS NOT EMPTY
      OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
      IF EMP-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION.
ADD
   LOGICAL RECORD NAME IS JOB-EMPOSITION
   ELEMENTS ARE JOB
                EMPOSITION.
ADD
PATH-GROUP NAME IS OBTAIN JOB-EMPOSITION
   SELECT FOR FIELDNAME-EQ JOB-ID-0440
      OBTAIN JOB
        WHERE CALCKEY EQ JOB-ID-0440 OF REQUEST
      IF JOB-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION
   SELECT USING INDEX JOB-TITLE-NDX
       FOR FIELDNAME TITLE-0440
      OBTAIN EACH JOB USING INDEX
      IF JOB-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION
   SELECT FOR FIELDNAME START-DATE-0420
      OBTAIN EACH EMPOSITION WITHIN EMP-DEMO-REGION
      IF JOB-EMPOSITION MEMBER
      OBTAIN OWNER WITHIN JOB-EMPOSITION
   SELECT FOR ELEMENT JOB
      OBTAIN EACH JOB WITHIN ORG-DEMO-REGION
      IF JOB-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION
   SELECT FOR ELEMENT EMPOSITION
      OBTAIN EACH EMPOSITION WITHIN EMP-DEMO-REGION
      IF JOB-EMPOSITION MEMBER
      OBTAIN OWNER WITHIN JOB-EMPOSITION
   SELECT
      OBTAIN EACH JOB WITHIN ORG-DEMO-REGION
        ON 0307 CLEAR RETURN LR-NOT-FOUND
        ON 0000 NEXT
      IF JOB-EMPOSITION IS NOT EMPTY
        ON 0000 ITERATE
        ON 1601 NEXT
      OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION
        ON 0000 NEXT
        ON 0307 ITERATE.
GENERATE.
```

Master File for LRF

```
FILE=EMPDATA,SUFFIX=IDMSR,$

SEGNAME=DEPTEMPO,$
  FIELD=DEPT_ID      ,ALIAS=        ,USAGE=A4    ,ACTUAL=A4    ,$
  FIELD=DEPT_NAME    ,ALIAS=        ,USAGE=A45   ,ACTUAL=A45   ,$
  FIELD=DEPT_HEAD    ,ALIAS=        ,USAGE=A4    ,ACTUAL=A4    ,$
  FIELD=             ,ALIAS=FILL.END,USAGE=A3    ,ACTUAL=A3    ,$
  FIELD=EMP_ID       ,ALIAS=        ,USAGE=A4    ,ACTUAL=A4    ,$
  GROUP=EMP_NAME     ,ALIAS=        ,USAGE=A25   ,ACTUAL=A25   ,$
    FIELD=FIRST_NAME,ALIAS=        ,USAGE=A10   ,ACTUAL=A10   ,$
    FIELD=LAST_NAME ,ALIAS=        ,USAGE=A15   ,ACTUAL=A15   ,$
  FIELD=EMP_STREET   ,ALIAS=        ,USAGE=A20   ,ACTUAL=A20   ,$
  FIELD=EMP_CITY     ,ALIAS=        ,USAGE=A15   ,ACTUAL=A15   ,$
  FIELD=EMP_STATE    ,ALIAS=        ,USAGE=A2    ,ACTUAL=A2    ,$
  GROUP=EMP_FULL_ZIP,ALIAS=        ,USAGE=A9    ,ACTUAL=A9    ,$
    FIELD=EMP_ZIP   ,ALIAS=        ,USAGE=A5    ,ACTUAL=A5    ,$
    FIELD=EMP_ZIP_L ,ALIAS=        ,USAGE=A4    ,ACTUAL=A4    ,$
  FIELD=EMP_PHONE    ,ALIAS=        ,USAGE=A10   ,ACTUAL=A10   ,$
  FIELD=STATUS       ,ALIAS=        ,USAGE=A2    ,ACTUAL=A2    ,$
  FIELD=SOC_SEC_NUM  ,ALIAS=        ,USAGE=A9    ,ACTUAL=A9    ,$
  FIELD=EMP_STRT_DTE,ALIAS=        ,USAGE=A6YMD,ACTUAL=A6    ,$
  FIELD=EMP_TERM_DTE,ALIAS=        ,USAGE=A6YMD,ACTUAL=A6    ,$
  FIELD=EMP_BRTH_DTE,ALIAS=        ,USAGE=A6YMD,ACTUAL=A6    ,$
  FIELD=             ,ALIAS=FILL.END,USAGE=A6    ,ACTUAL=A6    ,$
  FIELD=POS_STRT_DT1,ALIAS=        ,USAGE=A6YMD,ACTUAL=A6    ,$
  FIELD=POS_FIN_DT1  ,ALIAS=        ,USAGE=A6YMD,ACTUAL=A6    ,$
  FIELD=SALARY_GRAD1,ALIAS=        ,USAGE=P4    ,ACTUAL=Z2    ,$
  FIELD=SALARY_AMT1  ,ALIAS=        ,USAGE=P10.2,ACTUAL=P5    ,$
  FIELD=BONUS_PCT1   ,ALIAS=        ,USAGE=P4    ,ACTUAL=P2    ,$
  FIELD=COMMIS_PCT1  ,ALIAS=        ,USAGE=P4    ,ACTUAL=P2    ,$
  FIELD=OVERTIM_PCT1,ALIAS=        ,USAGE=P5.2  ,ACTUAL=P2    ,$

SEGNAME=JOBPOS,PARENT=DEPTEMPO,SEGTYPE=U,$
  FIELD=JOB_ID       ,ALIAS=        ,USAGE=A4    ,ACTUAL=A4    ,$
  FIELD=TITLE        ,ALIAS=        ,USAGE=A20   ,ACTUAL=A20   ,$
  FIELD=JOB_DESC     ,ALIAS=        ,USAGE=A120  ,ACTUAL=A120  ,$
  FIELD=REQUIREMENTS,ALIAS=        ,USAGE=A120  ,ACTUAL=A120  ,$
  FIELD=MIN_SALARY   ,ALIAS=        ,USAGE=P12.2,ACTUAL=Z8    ,$
  FIELD=MAX_SALARY   ,ALIAS=        ,USAGE=P12.2,ACTUAL=Z8    ,$
  FIELD=SAL_GRADE_1  ,ALIAS=        ,USAGE=P4    ,ACTUAL=Z2    ,$
  FIELD=SAL_GRADE_2  ,ALIAS=        ,USAGE=P4    ,ACTUAL=Z2    ,$
  FIELD=SAL_GRADE_3  ,ALIAS=        ,USAGE=P4    ,ACTUAL=Z2    ,$
  FIELD=SAL_GRADE_4  ,ALIAS=        ,USAGE=P4    ,ACTUAL=Z2    ,$
  FIELD=POSITION_NUM,ALIAS=        ,USAGE=P4    ,ACTUAL=Z3    ,$
  FIELD=NUM_OPEN     ,ALIAS=        ,USAGE=P4    ,ACTUAL=Z3    ,$
  FIELD=             ,ALIAS=FILL.END,USAGE=A2    ,ACTUAL=A2    ,$
  FIELD=POS_STRT_DT2,ALIAS=        ,USAGE=A6YMD,ACTUAL=A6    ,$
  FIELD=POS_FIN_DT2 ,ALIAS=        ,USAGE=A6YMD,ACTUAL=A6   ,$
  FIELD=SALARY_GRAD2,ALIAS=        ,USAGE=P4   ,ACTUAL=Z2   ,$
  FIELD=SALARY_AMT2 ,ALIAS=        ,USAGE=P10.2,ACTUAL=P5   ,$
  FIELD=BONUS_PCT2  ,ALIAS=        ,USAGE=P4   ,ACTUAL=P2   ,$
  FIELD=COMMIS_PCT2 ,ALIAS=        ,USAGE=P4   ,ACTUAL=P2   ,$
  FIELD=OVERTIM_PCT2,ALIAS=        ,USAGE=P5.2 ,ACTUAL=P2   ,$
```

Access File for LRF

This Access File is associated with LRF subschema EMPSS02, and corresponds to its Master File.

```
SSCHEMA=EMPSS02,RELEASE=15,MODE=LR,TRACE=PARMS,READY=ALL,$

SEGNAM=DEPTEMPO,RECORD=DEPT-EMP-POS,AREA=EMP-DEMO-REGION,LR=Y,$

SEGNAM=JOBPOS,RECORD=JOB-EMPOSITION,AREA=ORG-DEMO-REGION,LR=Y,
 ACCESS=LR,KEYFLD=POS_STRT_DT1,IXFLD=POS_STRT_DT2,$
```

Sample of a Partial LRF Record

The following is an example of a NULL SELECT clause that creates a partial record by returning a user-defined record code. The adapter does not support this user-defined code or any status code other than LR-FOUND or LR-NOT-FOUND.

```
SELECT
   OBTAIN EACH JOB WITHIN ORG-DEMO-REGION
   IF JOB-EMPOSITION IS NOT EMPTY
     ON 0000 RETURN NO-POS-FOR-JOB
   OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION.
```

SPF Indexes

The following is a section of a subschema that contains SPF indexes. Comparable Integrated Indexes are found in the LRF subschema EMPSS02 listed as EMP-NAME-NDX, JOB-TITLE-NDX, and SKILL-NAME-NDX.

```
ADD SET NAME IS IX-EMP-LNAME
  ORDER IS SORTED
  MODE IS CHAIN
  OWNER IS IXOWNER
     NEXT DBKEY POSITION IS AUTO
  MEMBER IS EMPLOYEE
     NEXT DBKEY POSITION IS AUTO
     OPTIONAL MANUAL
     ASCENDING KEY IS ( EMP-LAST-NAME-0415)
          DUPLICATES LAST
.

ADD SET NAME IS IX-TITLE
  ORDER IS SORTED
  MODE IS CHAIN
  OWNER IS IXOWNER
     NEXT DBKEY POSITION IS AUTO
  MEMBER IS JOB
     NEXT DBKEY POSITION IS AUTO
     OPTIONAL MANUAL
     DESCENDING KEY IS ( TITLE-0440 )
          DUPLICATES NOT ALLOWED
.

ADD SET NAME IS IX-SKILL-NAME
  ORDER IS SORTED
  MODE IS CHAIN
  OWNER IS IXOWNER
     NEXT DBKEY POSITION IS AUTO
  MEMBER IS SKILL
     NEXT DBKEY POSITION IS AUTO
     OPTIONAL MANUAL
     ASCENDING KEY IS ( SKILL-NAME-0455 )
          DUPLICATES NOT ALLOWED
    .
```

File Retrieval

In this section:

The Adapter for IDMS/DB retrieves the necessary records that fulfill a request. It uses information from the following sources to choose the most appropriate and efficient retrieval method:

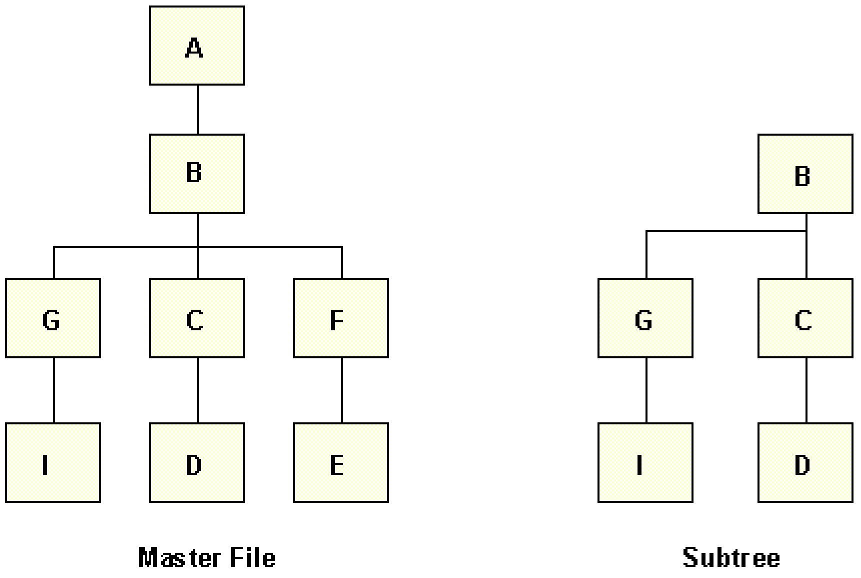
The server uses the Master File to select segments and retrieve data from them.

Note: Sections pertaining to unique segments discuss rules that apply regardless of whether the unique segment is the owner of its parent or is related to its parent through a CALC field, index, or LRF field.

Retrieval Subtree

To retrieve records for a request, the server first constructs a smaller subtree structure from the structure defined by the Master File. The subtree consists of segments that contain fields named explicitly in the request and those named implicitly by DBA DEFINE or COMPUTE statements. The subtree also includes any segments needed to connect these segments.

For example, if an SQL request needs fields from segments D, G, and I, the server constructs the subtree shown below. Segments C and B do not contain fields needed for the request, but they are included in the subtree to connect segment I with segments D and G. Since the segments are descendants of segment B, the entry or root segment of the subtree is segment B. The Master File root segment A is not included, because its fields are not referenced; the records corresponding to segment B can be obtained independently of A. However, if segment B were an OCCURS segment, the IDMS/DB calls would be issued for segment A to obtain B's information.

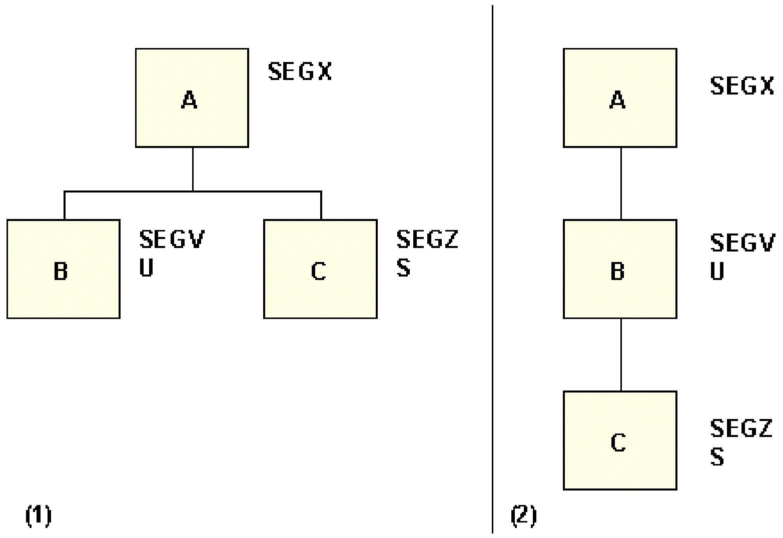


Retrieval Sequence With Unique Segments

The retrieval sequence for subtrees containing unique segments is still top-to-bottom, left-to-right, but the unique segments are treated as extensions of their parents. Records in a unique segment correspond one-to-one with the records in a parent; records in a non-unique segment have a one-to-many correspondence. In cases where the parent segment has unique and non-unique descendants, the unique descendants are always retrieved first regardless of the left-to-right order.

A retrieval view shows which sort and WHERE clauses are valid. For sort statements (BY or ACROSS), the segment containing the sortfield must lie on the same path as the segments with all requested fields. That is, the segment with the BY or ACROSS field must be an ancestor or descendant of the segments containing the required fields.

For instance, panel 1 of the following graphic shows two descendant segments. The statement SELECT B ORDER BY C is invalid, because the segments containing fields B and C do not lie on the same path. However, if the segment containing B is a unique segment, the two segments do lie on the same path. Panel 2 shows the retrieval view; the statement SELECT B ORDER BY C is valid.



The retrieval sequence for unique segments may also affect the results of an SQL statement that contains COUNT or SUM. If a segment is the parent of a unique descendant, there is a one-to-one relationship. A COUNT statement, such as COUNT A AND B, returns identical results for each field, because the same record A is counted several times for each record B. If the parent/descendant relationship is reversed with a non-unique parent, the result for field A is a greater number than the result for field B.

Screening Conditions

If a record in a segment fails a WHERE condition, the server does not retrieve the corresponding records in descendant segments. Suppose this request is entered against the structure EMPSS01 (corresponding Master File is EMPFULL).

```
SELECT EMP_ID,OFF_CODE
FROM EMPFULL
WHERE DEPT_NAME = 'PERSONNEL'
ORDER BY OFF_CODE
```

Every time a record in segment DEPT has a value in the DEPT_NAME field not equal to PERSONNEL, the server ignores the corresponding records in descendant segments EMPLOYEE and OFFICE, and retrieves the next record in segment DEPT. In addition, when a WHERE clause on a lower segment fails, the row is removed from the server answer set.

To increase I/O efficiency, place the WHERE clauses at a higher level in the file structure. This restricts the number of records the server has to test. The example below shows the benefits of two WHERE clauses versus one.

Assume a subtree has four segments:

- DEPT contains department IDs and information.
- EMPLOYEE contains employee names and IDs.
- EMPOSIT contains the positions that the employee has held.
- JOB contains a list of jobs offered by the company.

To list all employees who are programmer/analysts:

```
SELECT TITLE,DEPT,LAST_NAME,FIRST_NAME
FROM EMPFULL
WHERE TITLE = 'PROGRAMMER/ANALYST'
ORDER BY DEPT,LAST_NAME,FIRST_NAME
```

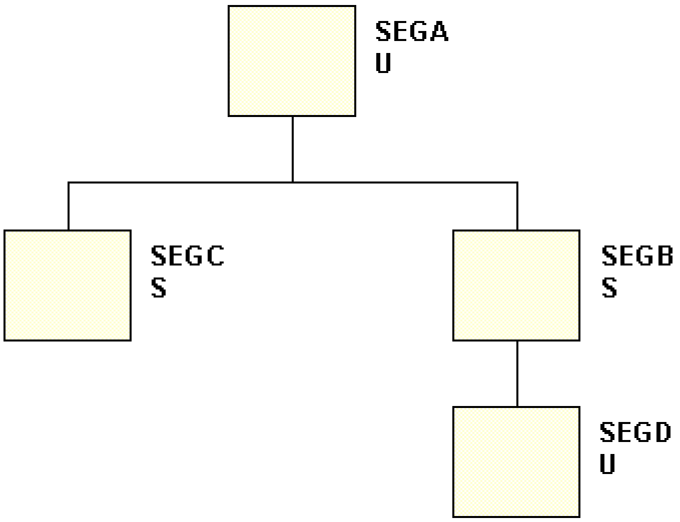
For this request with only one WHERE clause, the server retrieves each DEPT record, each EMPLOYEE record for a given DEPT, each EMPOSIT record for a given EMPLOYEE record, and the JOB record connected to each EMPOSIT. After retrieval, the server determines whether to include in the answer set the record from the value of the TITLE field. To make retrieval more efficient, add another WHERE clause on a segment higher in the structure. In this company, only the Internal Software department has programmer/analysts working for it:

```
SELECT TITLE,DEPT,LAST_NAME,FIRST_NAME
FROM EMPFULL
WHERE DEPT_NAME = 'INTERNAL SOFTWARE'
WHERE TITLE = 'PROGRAMMER/ANALYST'
```

Now the server retrieves and tests records only when the DEPT_NAME field equals the value INTERNAL SOFTWARE.

Screening Conditions With Unique Segments

If a record in a unique segment fails a WHERE test, the server rejects its parent and retrieves the next record of the parent segment. For example, in the following graphic, if a record in non-unique segment C fails a WHERE clause, the server retrieves the next record in segment C. Only if all C records for a given A fail the test is the A record rejected. When a record in unique segment D fails a test, the server rejects the parent B record and retrieves the next record in segment B. When a record in the entry A segment fails a test, the server retrieves the next A record, even if the entry segment is defined as unique.



Short Paths

When the server retrieves a record in a parent segment, it retrieves the corresponding records in the descendant segment. If descendant records do not exist, the processing of the parent record and whether it is included in an answer set depends on whether the descendant segment is unique or non-unique.

Short Paths in Unique Descendants

For a unique descendant with a missing record, the server creates a temporary record to replace the missing record. The temporary record contains fields with default values: blanks for alphanumeric fields and zeroes for numeric fields.

For example, an EMPLOYEE segment with the field EMP_NAME has a unique descendant OFFICE segment with the field OFF_CITY. The field OFF_CITY indicates the location of an employee's office. Gary Smith does not work out of an office location; so, he has no OFFICE record. In this situation, all requests that refer to OFF_CITY display blank spaces for the entry GARY SMITH.

Short Paths in Non-Unique Descendants

For a non-unique descendant segment with a missing record, the server rejects the parent instance and retrieves the next parent instance.

Syntax: How to Specify Short Paths in Non-Unique Descendants

For a non-unique descendant segment with a missing record, the results depend on how the ALL parameter is set:

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

where:

`ON`

The parent record is processed provided that there are no screening conditions on fields in the descendant segment. Missing data is usually indicated on the report by the default NODATA character (.).

`OFF`

The parent instance is rejected and the next parent instance is retrieve. OFF is the default value.

Note: SET ALL = PASS is *not* supported by the adapter.

Record Retrieval

In this section:

To obtain all of the necessary records to fulfill a request, the Adapter for IDMS/DB navigates the IDMS/DB database using DML or LRF commands. The adapter automatically generates DML or LRF commands based on information from the Master File, Access File, and your request for the most appropriate and efficient IDMS/DB retrieval method.

There are three kinds of IDMS/DB access:

- Entry Segment Retrieval of Network Records.
- Descendant Segment Retrieval of Network Records.
- LRF Record Retrieval (LR and ASF).

Subsequent sections discuss the navigational strategies used for each kind of access.

Entry Segment Retrieval of Network Records

The server constructs a retrieval subtree based on the Master and Access Files and your request. The root of this subtree is called the entry segment, because the server begins its retrieval search of the database at that point. The actual IDMS/DB retrieval calls used on the entry segment depend on the entry segment's Access File information and any WHERE clauses. To perform the most efficient record retrieval on the entry segment, the adapter chooses one of the following techniques:

- Retrieval by Database Key
- Retrieval by CALC Field
- Retrieval by Index
- SEQFIELD Parameter
- Retrieval by Area Sweep

These techniques are listed in descending order of efficiency. The idea behind selection logic is to perform as many WHERE clauses as possible at the IDMS/DB level. This minimizes the actual I/O operations required to access the necessary data. Area sweeps are the least desirable retrieval technique, because they read through every record type in the named area, including record types that correspond to other segments, and return every entry segment record to the server. At this point, the server selects those records that satisfy the request's test criteria and discards the rest.

Retrieval by Database Key

The IDMS/DB database key method of retrieval takes precedence over the other methods because it is the most efficient. This method depends on the existence of two conditions:

- A field that corresponds to the IDMS/DB database key (ALIAS=DBKEY) for the entry segment.
- An equality test in the request on the DBKEY field.

The equality test sets the field name of the DBKEY from the entry segment in the Master File equal to a specified numeric value(s):

```
WHERE field = 'value1'
```

Retrieval by CALC Field

If there is no WHERE clause on the DBKEY for the entry segment, the second choice is CALC access. Retrieval through the CALC key, while not as efficient as DBKEY access, takes precedence over an index or area sweep retrieval.

The CALC retrieval method depends on the existence of two conditions:

- The entry segment must contain a CALC keyfield as specified in the Access File.
- A fully qualified equality test in the request on the CALC field.

The WHERE clause sets the field name of the CALC key for the segment equal to fully qualified values:

```
WHERE field = 'value1'
```

For each value specified in the WHERE clause, the adapter calls IDMS/DB with the following DML command:

```
OBTAIN CALC record
```

If the Access File indicates duplicate records (CLCDUP=Y), each DML call is followed by subsequent calls:

```
OBTAIN DUPLICATE record
```

This ensures that all appropriate records are obtained to satisfy the request.

Retrieval by Index

If there is no DBKEY or CALC key test criteria in the request, the adapter selects the index retrieval method.

Note: An index field must be defined with the FIELDTYPE=I attribute in the Master File and a corresponding index declaration must exist in the Access File.

Index retrieval is performed using:

- WHERE clauses for an indexed field or GROUP in the entry segment.
- The optional parameter SEQFIELD in the Access File segment declaration.

The first method requires at least one WHERE clause that specifies the field name of the index field. The following WHERE clause invokes index-based retrieval:

- Fully qualified.

  ```
  WHERE = 'PROGRAMMER/ANALYST'
  ```

- Partially qualified (generic), also called masking. This applies to alphanumeric fields only.

  ```
  WHERE TITLE LIKE 'PROGRAMMER%'
  ```

- Specified as a range of values.

  ```
  WHERE TITLE BETWEEN 'PROGRAMMER' AND 'WORD PROCESSOR'
  ```

When two or more WHERE clauses in a request qualify an index on the entry-level segment, fully qualified retrieval takes precedence over generic; generic retrieval takes precedence over range. If two WHERE clauses are the same type, the index in the first WHERE clause is used. All types of WHERE clauses are also supported for indices that allow duplicate values.

If the test criteria indicates index retrieval, the adapter issues this DML command to IDMS/DB:

```
OBTAIN FIRST record WITHIN setname USING value
```

In this command, *setname* is the name of the index set specified in the Access File.

Then, if the Access File indicates duplicate records, the adapter issues this DML command for index sets:

```
OBTAIN NEXT record WITHIN setname
```

The above OBTAIN NEXT call is issued until all the duplicate records are retrieved. This same NEXT call is also issued if your request contains generic or range WHERE clauses. It is performed once for every value or range specified in the WHERE clause.

Note: If the IXORD parameter is improperly specified in the Access File, a range WHERE clause may erroneously produce an answer set with one or no records.

SEQFIELD Parameter

The second method of index retrieval does not require WHERE clauses, and yet prohibits area sweeps on entry segments. To use this method, add the optional SEQFIELD parameter to the Access File and specify the name of the indexed field as the value of SEQFIELD.

When your request does not contain a WHERE (for DBKEY, CALC key, or another index) and a SEQFIELD is specified for the entry segment, the adapter issues this DML command to IDMS/DB:

```
OBTAIN FIRST record WITHIN setname
```

In this command, *setname* is the IDMS/DB set name of the indexed field (IXSET parameter) from the Access File.

The adapter then issues the next command until all records connected to the index set are retrieved:

```
OBTAIN NEXT record WITHIN setname
```

If no sort criteria (ORDER BY) is specified in the request, the answer set is produced in ascending or descending index set order.

The SEQFIELD method is recommended for indexed segments in large IDMS/DB databases where only a small percentage of record occurrences in a given area are the record types defined by the segments. In such cases, IDMS/DB resource utilization can be greatly reduced through the use of this parameter.

Note: If the index set connection is not mandatory/automatic (MA), some of the records in this record type may not be accessed if it is the entry segment. In this situation, only records that are members of the index set are supplied to the server. If this retrieval result is undesirable, you should omit the SEQFIELD parameter.

Retrieval by Area Sweep

An area sweep is the least efficient method of entry-level retrieval, because it reads through every record in an IDMS/DB area to return records of a given record type. Despite its inefficiency, an area sweep is sometimes the only method available for retrieval.

The adapter performs an area sweep if one of the following occurs:

- No equality WHERE on the DBKEY for the root exists.
- No equality WHERE on the CALC key for the root exists.
- No equality or range WHERE on an indexed field exists.
- No SEQFIELD parameter is specified.

If one of the above situations occurs, the adapter issues this DML command to IDMS/DB:

```
OBTAIN FIRST record WITHIN areaname
```

In this command, *areaname* is the name of the IDMS/DB database area specified in the Access File. Next, the adapter continues to issue this command until all the records are obtained:

```
OBTAIN NEXT record WITHIN areaname
```

Descendant Segment Retrieval of Network Records

To retrieve records from a descendant segment, the adapter's navigational strategy depends on Access File parameters and the SEGTYPE parameter in the Master File. In a few cases, the WHERE clauses in a request also affect the strategy.

In general, the ACCESS parameter in the Access File determines retrieval strategy, because it indicates how parent/descendant relationships are implemented. There is a retrieval strategy for each kind of relationship:

- Set-Based Retrieval
- CALC-Based Retrieval
- Index-Based Retrieval

- LRF Record Retrieval
- Overriding DBNAME and DICTNAME

Set-Based Retrieval

For a set relationship (ACCESS=SET), the IDMS/DB set is searched starting with the owner to obtain related member record(s). Set relationships are physical ones, implemented by set pointer chains. The SEGTYPE parameter in the Master File indicates whether the descendant segment is unique or non-unique.

If the descendant segment is non-unique (SEGTYPE=S), it represents a member record type. For non-unique descendants, the adapter issues this command:

```
OBTAIN NEXT record WITHIN setname
```

This command is repeated until IDMS/DB indicates that the end of the set is reached. Then the adapter obtains records of other descendant segments for the same parent segment. If no other descendant segments exist, the next parent record is retrieved.

If the descendant segment is unique (SEGTYPE=U), it represents an owner record type. For unique descendant segments, the adapter issues this command:

```
OBTAIN OWNER WITHIN setname
```

The command is issued once, since there is one owner per set. The adapter continues to retrieve descendant records for the same parent or retrieves the next parent record.

The KEYFLD and MULTMBR parameters in the Access File also affect retrieval for certain requests. For a sorted set, the KEYFLD parameter specifies that the set is ordered by a specified field. When the request references a field from the parent segment and has a WHERE (=, BETWEEN, >, >=, <, <=) on the sortfield, the adapter sends this command to IDMS/DB:

```
OBTAIN record WITHIN setname USING value
```

If there are duplicate records (SETDUP=Y), the adapter issues this command:

```
OBTAIN NEXT record WITHIN setname
```

When the value of the sortfield changes beyond the specified range, retrieval for that segment stops. I/O operations are minimized when the sortfield value is supplied in the OBTAIN command.

Note: The means of implementing the sorted set—the traditional method or using IDMS/DB Integrated Indices—is transparent to the adapter.

The MULTMBR parameter indicates an IDMS/DB multi-member set. When it is specified, the adapter searches for other member record types (segments) in the Access File with the same setname. As a result, all necessary IDMS/DB areas are activated.

CALC-Based Retrieval

CALC-based relationships (ACCESS=CLC) are performed with embedded cross-references. A field in the parent segment corresponds to the CALC field in its descendant segment. The adapter uses the value from the parent's field and performs entry-level IDMS/DB retrieval. The process of retrieving records from a descendant segment is similar to that of an entry segment. The difference is that the value supplied by the parent segment acts as the WHERE clause as if it were an explicit WHERE.

After the adapter retrieves the host field value (KEYFLD=value) from the parent segment, it calls IDMS/DB:

```
OBTAIN CALC record
```

Then, if the descendant segment is non-unique (SEGTYPE=S), the adapter issues this command until all of the appropriate records are obtained to satisfy the request:

```
OBTAIN DUPLICATE record
```

Note: If the CLCDUP parameter does not correspond to the SEGTYPE parameter, message EDA919 displays.

A descendant segment with a CALC-based relationship (ACCESS=CLC) may act as a parent and be related to its descendants using set-, CALC-, or index-based relationships.

Index-Based Retrieval

Like CALC-based relationships, index-based relationships (ACCESS=IX) also use embedded cross-references. In index-based relationships, the field in a descendant segment represents an index on the IDMS/DB record type. The index can be either a Sequential Processing Facility (SPF) index or an Integrated Index. The adapter uses the value from the parent segment and performs entry-level IDMS/DB retrieval by searching the index set. The process of retrieving records from a descendant segment is similar to that of an entry segment. The difference is that the value supplied by the parent segment acts as the WHERE clause, as if it were an explicit WHERE.

After the adapter retrieves the host field value (KEYFLD=value) from the parent segment, it calls IDMS/DB:

```
OBTAIN FIRST record WITHIN setname USING value
```

Then, if the descendant segment is non-unique (SEGTYPE=S), the adapter issues the following command until all indexed records with the host value are retrieved:

```
OBTAIN NEXT record WITHIN setname
```

Note: If the IXDUP parameter does not correspond to the SEGTYPE parameter, message EDA919 displays.

Only a descendant segment with an Integrated Index may act as a parent and be related to its descendants using set-, CALC-, or index-based relationships.

LRF Record Retrieval

To retrieve LR and ASF records, the adapter sends LRF calls to an access module IDMS/DB which invokes the Logical Record Facility program.

LRF-based records are retrieved when the Access File specifies MODE=LR in the subschema declaration and the adapter constructs an LR call to IDMS/DB with explicit or implicit WHERE clauses from the request. The LRF processes the request as generated by the adapter, selects the appropriate LR path, and constructs each flat view using the full set of WHERE clauses. The process is highly efficient in terms of I/O; only those records which pass the WHERE clause are passed back to the adapter from IDMS/DB.

The retrieval process for LRF records is identical to that of network record types, but the Logical Record Facility maintains its own navigational information for the database, selects the retrieval strategy most appropriate for a given request, and maintains its own set of occurrences.

When the subschema mode is Logical Record (MODE=LR), the adapter analyzes the request and creates an LRF command to be sent to the Logical Record Facility. The LRF command has two formats. The first format is for an entry segment without WHERE clauses:

```
OBTAIN NEXT record
```

The second format is for an entry with WHERE clauses or for a descendant segment:

```
OBTAIN NEXT record WHERE expression1 [AND expressionN]
```

As indicated by the brackets, this command is also used to pass compound WHERE clauses to IDMS/DB.

IDMS/DB processes the adapter's call and returns a record if two conditions are met:

- There is a SELECT path in the LR path-group that can process the particular request. (**Note:** For entry segments without WHERE clauses, there must be a null SELECT clause defined for the logical record.) If there is no available SELECT path, IDMS/DB returns message 2041, and the adapter terminates its processing with an EDA949 message.
- There is a complete LRF record created by the path-group logic. If the selected path-group can return partial records, the adapter processes returned records until the first partial record is returned and the LR status field is not LR-FOUND or LR-NOT-FOUND. When the adapter encounters any other status in the LR status field, it aborts the record retrieval process and returns messages EDA967 and EDA949. Logical records that allow for retrieval of partial records should not be used with the adapter.

The adapter continues to call IDMS/DB for LRF records that correspond to a segment until IDMS/DB returns LR-NOT-FOUND in the LR status field. Then the adapter retrieves records for other segments, or it terminates retrieval and the answer set is produced.

Overriding DBNAME and DICTNAME

How to:

Set DBNAME and DICTNAME

Display the Current Settings

Revert to Original Settings

Override the DBNAME and DICTNAME in All IDMS/DB Access Files With SYSDIRL

You can dynamically override the DBNAME and DICTNAME parameters within the Access File. This allows you to specify the DBNAME and DICTNAME for all IDMS/DB Master File and Access File pairs during a session by using a SET command, eliminating the need to modify each Access File manually.

To override the DBNAME and DICTNAME, issue the SET commands outlined below. Once the SET commands are issued, the DBNAME and DICTNAME specified will override the same keywords in all Access Files during your session until you end your session or set the DBNAME and DICTNAME to default.

If no SET command is issued, the default behavior is followed.

Syntax: How to Set DBNAME and DICTNAME

```
ENGINE IDMSR SET DBNAME dbname
```

```
ENGINE IDMSR SET DICTNAME dictname
```

where:

dbname

Is the IDMS/DB database name that you want to access.

dictname

Is the IDMS/DB dictionary name that you want to access.

Note: These set commands can be included in any supported server profile.

Syntax: How to Display the Current Settings

To display the settings that are currently in effect, issue the following command from a client application or from a remote procedure:

```
EX EDAEXEC 'ENGINE IDMSR SET ?'
```

Syntax: How to Revert to Original Settings

To revert to original settings in the Access File, issue the following commands in a remote procedure:

```
ENGINE IDMSR SET DBNAME DEFAULT
ENGINE IDMSR SET DICTNAME DEFAULT
```

Syntax: How to Override the DBNAME and DICTNAME in All IDMS/DB Access Files With SYSDIRL

```
ENGINE IDMSR SET DBNAME SYSDIRL
ENGINE IDMSR SET DICTNAME SYSDIRL
```

Customizing the IDMS/DB Environment

In this section:

File Inversion

Joining Master Files

You can customize the IDMS/DB environment by inverting files and joining Master Files.

File Inversion

How to:

Invert a File

Example:

Using File Inversion to Solve Sort Path Problems

When you create a Master File, you create a default representation of a hierarchy. Sometimes, however, you may not want to follow the default route to retrieve records. Two such instances might be when:

1. Your IF criteria screen a segment at the bottom of a subtree.
2. You are processing a multi-path report with IF criteria or sort phrases that are not on a common path.

When these situations occur, you can specify a new entry segment (root) at execution time for a specific request. This process is called file inversion, because the parent/descendant relationships along the path linking the original root and the new root are reversed; other parent/descendant relationships remain unchanged.

Note: File inversions only change the file views; they do not affect the data.

Syntax: How to Invert a File

```
TABLE FILE filename.field
```

where:

field

May be any field in the new root segment.

For example, to invert the EMPFULL file so that the office segment is the new root, specify the field OFFICE_CODE:

```
TABLE FILE EMPFULL.OFFICE_CODE
```

You can also display a diagram of the inverted file with the CHECK FILE command (include the RETRIEVE option for a subtree diagram):

```
CHECK FILE filename.fieldname PICTURE [RETRIEVE]
```

You cannot invert a Master File if:

- The path linking the old and new roots passes through segments that have a CALC- or index-based relationship.
- The GETOWN parameter in the ACCESS File for a set-based relationship is set to N.
- It is an LRF Master File.

File inversion is a simple solution to two common problems:

- Denied access because the segment is on the wrong sort path.
- Denied access because the field named in an IF test is not on the root path.

Example: Using File Inversion to Solve Sort Path Problems

In addition to solving the sort path problem, file inversion can improve I/O efficiency which, in turn, minimizes production costs.

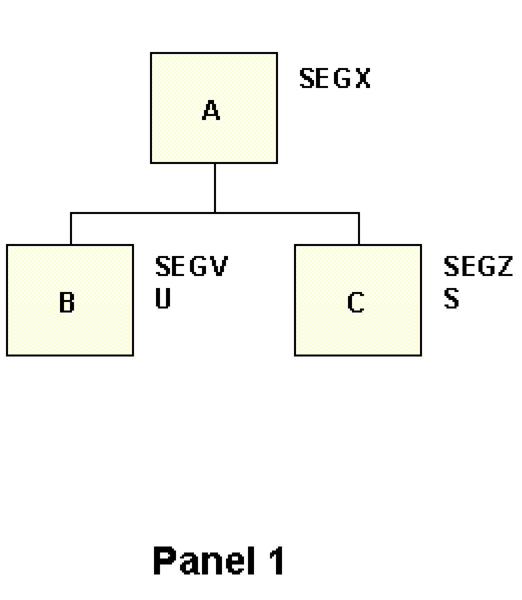
Consider this request:

```
TABLE FILE EMPFULL
LIST SKILL_LEVEL BY SALARY_GRADE
END
```

In panel 1 of the following figure, an error occurs because segments C and B are not on the same path. Therefore, you must use an inverted view:

```
TABLE FILE EMPFULL.SALARY_GRADE
LIST SKILL_LEVEL BY SALARY_GRADE
END
```

In the inverted view (panel 2), segment C is a descendant of segment B. Using this inverted view, the request can be executed.



As this request is executed, record occurrences multiply. Every record of segment C is paired with every record in segment B. If, for example, A had two B descendants and four C descendants, the report would contain eight lines of output. This effect is advantageous when it is necessary to pair every record associated with one linkpath to a record associated with another linkpath. Record pairing may produce undesirable results when the inverted segments are not directly related to each other.

If you use file inversion in conjunction with MISSING=ON, you may access orphan record occurrences that could not be accessed with the default Master File. An orphan record occurrence is one that has no parent record connection. Due to the network structure of IDMS/DB, any hierarchical view may contain orphans. IDMS/DB set connection options OA, OM, or MM indicate the possibility of orphans. Inversion enables the adapter to reconstruct the IDMS/DB relationships so that these orphans can be retrieved.

Joining Master Files

You can join the Master Files describing any of these data sources to that of your IDMS/DB data source:

- Other IDMS/DB data sources (SUFFIX=IDMSR)
- VSAM or ISAM or QSAM
- SQL or UDB (DB2)
- DOS/DL1 or IMS
- CA-Datacom/DB
- MODEL 204
- Fixed-format sequential

A JOIN structure is implemented by matching one field that is common to both data sources. The fields on the IDMS/DB target file can be:

- An IDMS/DB CALC field on a network record-type.
- An indexed field (FIELDTYPE=I) on a network record-type.
- A field on an LRF record.

The fields on the host file can be:

- A virtual field located in a host Master File or created as a separate command.
- Any field.

In the Master File, the names of common fields can differ, but their field formats (ACTUAL and USAGE) must be the same.

Syntax: How to Join Two Data Sources

```
JOIN field1 [WITH rfield] IN hostfile [TAG tag1]
TO [ALL] field2 IN crfile [TAG tag2] [AS name]
[END]
```

where:

`field1, field2`
: Are the fields common to both Master Files.

`WITH rfield`
: Use only if *field1* is a virtual field; assigns a logical home with a real field in the host file.

`hostfile`
: Is the host Master File.

`TAG tag1`
: Is a tag name of up to eight characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in the host file.

  The tag name for the host file must be the same in all the JOIN commands of a joined structure.

`ALL`
: Use if non-unique relationships exist in the target file.

`crfile`
: Is the target or cross-referenced Master File.

`TAG tag2`
: Is a tag name of up to eight characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in the cross-referenced file. In a recursive joined structure, if no tag name is provided, all field names and aliases are prefixed with the first four characters of the join name.

`AS name`
: Assigns a name to the JOIN structure. You must assign a unique name to a join structure if:

  - You want to ensure that a subsequent JOIN command will not overwrite it.
  - You want to clear it selectively later.
  - The structure is recursive, and you do not specify tag names.

`END`
: Required when the JOIN command is longer than one line; terminates the command.

To join more than two files as a single structure, indicate the common fields as follows:

```
JOIN field1 IN file1 TO field2 IN file2 AS name1
JOIN field3 IN file1 TO field4 IN file3 AS name2
```

Reference: Usage Notes for the JOIN Command

- Up to 16 joins may be active in one session.
- For a DML target file, field2 must be indexed (FIELDTYPE=I).
- If you intend to use a virtual field as *field1*, specify its field name in the JOIN command and then issue its DEFINE command. Any DEFINE commands issued prior to the JOIN are cleared.
- If you know that the target file is unique, omit the ALL in the JOIN command; omitting ALL reduces I/O overhead.
- To display the JOIN structure, use the CHECK FILE command and specify the name of the host file.

Syntax: How to List JOIN Structures

To list your JOIN structures, enter:

```
? JOIN
```

Syntax: How to Clear JOIN Structures

To clear a specific JOIN structure, specify the name that you assigned to the join:

```
JOIN CLEAR name
```

Syntax: How to Clear All JOIN Structures

To clear all structures, use an asterisk (*) instead of a join name:

```
JOIN CLEAR *
```

Example: Reporting From a Joined Structure

This example joins the DML data source JOBFILE to the IDMS/DB EMPFULL data source based on job codes. First the JOBCODE field in JOBFILE is edited to make it compatible with the JOB_ID field in EMPFULL. The JOIN command is issued prior to the DEFINE. If the DEFINE were issued first, it would be cleared by the JOIN command:

```
JOIN JOBID WITH JOBCODE IN JOBFILE TO
ALL JOB_ID IN EMPFULL AS J1
END
DEFINE FILE JOBFILE
JCODE/A2 = IF JOBCODE LIKE 'A__' THEN '10' ELSE '20';
JOBID/A4 = JCODE|EDIT(JOBCODE,'$99');
END
TABLE FILE JOBFILE
SUM EMP_NAME IN 25 TITLE
BY DEPT_NAME
END
```

The output is:

```
DEPT_NAME                EMP_NAME                   TITLE
---------                ---------                  -----
ACCOUNTING AND PAYROLL   RUPERT     JENSON          MGR ACCTNG/PAYROLL
PERSONNEL                ELEANOR    PEOPLES         MGR PERSONNEL
```

Tracing the Adapter for IDMS/DB

From the Web Console main screen, select *Diagnostics* and click *Enable Traces*. The default traces include the Adapter for IDMS/DB trace information.

CHAPTER 12

Using the Adapter for CA-IDMS/SQL

Topics:

The Adapter for CA- IDMS/SQL allows applications to access IDMS/SQL data sources. The adapter converts application requests into native IDMS/SQL statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

In order to use the Adapter for CA-IDMS/SQL, you must configure and map data to its corresponding server counterparts. Check with your Server Administrator for further information.

Note: For the remainder of this manual, the name IDMS/SQL refers to CA-IDMS/SQL.

Preparing the IDMS/SQL Environment

Prior to configuring the Adapter for IDMS/SQL using the Web Console, it is necessary to edit the ISTART JCL used to initiate the server. The changes required are documented as follows:

- Allocate the IDMS.LOADLIB and IDMS.DBA.LOADLIB libraries to the server's STEPLIB ddname allocation.
- Allocate ddname SYSCTL to the IDMS.SYSCTL data set.
- Allocate ddname SYSIDMS to the IDMS.SYSIDMS data set.

Configuring the Adapter for IDMS/SQL

In this section:

Configure the adapter using the Web Console Adapter Add screen and click *Configure*.

Overriding the Default Connection

How to:

An SQL session is a connection between the application and the IDMS/SQL data source. It begins when the application connects to a dictionary. You use the CONNECT command to override the IDMS/SQL default (automatic) connection. The length of time an SQL session stays in effect depends on whether the connection began automatically or a CONNECT command was issued. If the CONNECT command was issued, the SQL session is in effect until a COMMIT RELEASE, ROLLBACK RELEASE, or RELEASE command is executed. All of these commands may be executed within the IDMS/SQL session using SQL Passthru. Refer to the appropriate IDMS documentation for more information regarding SQL sessions.

Syntax: How to Override Default Connections With the CONNECT Command

Issue the following syntax within a stored procedure

```
ENGINE [SQLIDMS] CONNECT TO dictionary
```

where:

`SQLIDMS`

Indicates the Adapter for IDMS/SQL. You can omit this value if you previously issued the SET SQLENGINE command.

`dictionary`

Is the data source (dictionary) to start the IDMS/SQL session. The default is the dictionary in effect for the user session. This default is set outside of the server session, for example, with a SYSIDMS DICTNAME parameter. Refer to the appropriate IDMS documentation for a complete description.

Other Session Commands

Other IDMS/SQL commands that affect the IDMS/SQL session can be executed explicitly.

To issue IDMS/SQL session commands such as COMMIT, COMMIT RELEASE, ROLLBACK, ROLLBACK RELEASE, and COMMIT CONTINUE, the syntax is:

```
ENGINE [SQLIDMS] COMMIT RELEASE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLIDMS] SET AUTODISCONNECT ON {COMMIT|FIN}
```

where:

`SQLIDMS`

Indicates the Adapter for IDMS/SQL. You can omit this parameter value if you previously issued the SET SQLENGINE command.

`COMMIT`

Disconnects when a COMMIT or ROLLBACK is issued as a native SQL command. This setting frees the thread of execution for use by other users. The disadvantage is the cost of repeatedly connecting and acquiring a thread. Threads, once released, may not be available when needed, so you may experience delays while your request waits for a thread.

`FIN`

Disconnects automatically only after the server session has been terminated. FIN is the default value.

Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing IDMS/SQL Metadata

In this section:

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the IDMS/SQL data types.

Creating Synonyms

How to:

Create a Synonym Using the Web Console

Create a Synonym Manually

Example:

Using CREATE SYNONYM

Reference:

Managing Synonyms

Access File Keywords

Synonyms define unique names (or aliases) for each IDMS/SQL table or view that is accessible from the server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value.
11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.
12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```
14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.

Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLIDM [NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

`SQLIDM`

Indicates the Data Adapter for IDMS/SQL.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLIDMS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLIDMS ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4,     I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=IDMS901,KEYS=1, WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Name of the table or view. This value can include a location or owner name as follows: `TABLENAME=[`*`location.`*`][`*`owner.`*`]`*`tablename`*
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.

`DBSPACE`	Optional keyword that indicates the storage area for the table. For example:
	`datasource.tablespace`
	`DATABASE` *`datasource`*

Keyword	Description
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Master Files

In this section:

MISSING Attribute

How to:

Declare File Attributes

Declare Segment Attributes

Declare Field Attributes

The following describes the three types of Master File declarations:

Declaration Type	Description
`File`	Names the file and describes the type of data source.
`Segment`	Identifies a table, file, view, or segment.
`Field`	Describes the columns of the table or view.

Each declaration must begin on a separate line. A declaration consists of attribute-value pairs separated by commas. A declaration can span as many lines as necessary, as long as no single keyword-value pair spans two lines.

Do not use system or reserved words as names for files, segments, fields, or aliases. Specifying a reserved word generates syntax errors.

Syntax: How to Declare File Attributes

Each Master File begins with a file declaration. The file declaration has two attributes, FILENAME and SUFFIX:

```
FILE[NAME]=file, SUFFIX=SQLIDMS [,$]
```

where:

file

Identifies the Master File. The file name can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents.

SQLIDMS

Is the value for the Adapter for IDMS/SQL.

Syntax: How to Declare Segment Attributes

Each table described in a Master File requires a segment declaration. The segment declaration consists of at least two attributes, SEGNAME and SEGTYPE:

```
SEGNAME=segname, SEGTYPE=S0 [,$]
```

where:

segname

Is the segment name that serves as a link to the actual IDMS/SQL table name. It can consist of a maximum of 8 alphanumeric characters. It may be the same as the name chosen for FILENAME, the actual table name, or an arbitrary name.

The SEGNAME value in the Master File must be identical to the SEGNAME value specified in the Access File.

S0

Indicates that IDMS/SQL assumes responsibility for both physical storage of rows and the uniqueness of column values (if a unique index or calc key exists). It always has a value of S0 (S zero).

Syntax: How to Declare Field Attributes

Each row in a table may consist of one or more columns. These columns are described in the Master File as fields with the primary field attributes, FIELDNAME, ALIAS, USAGE, ACTUAL, MISSING.

Note: You can get values for these attributes from the IDMS/SQL schema definition or standard IDMS/SQL dictionary reports.

```
FIELD[NAME]=fieldname, [ALIAS=]sqlcolumn, [USAGE=]display_format,
[ACTUAL=]storage_format [,MISSING={ON|OFF}], $
```

where:

`fieldname`

Is the unqualified name of the field. This value must be unique within the Master File. The name can consist of a maximum of 48 alphanumeric characters (including any file name and segment name qualifiers and qualification characters you may later prefix to them in your requests). The name must begin with a letter. Special characters and embedded blanks are not recommended. The order of field declarations in the Master File is significant with regard to the specification of key columns. For more information, see *Primary Key* on page 12-15.

It is not necessary to describe all the columns of the IDMS/SQL table in your Master File.

`sqlcolumn`

Is the full IDMS/SQL column name (the adapter uses it to generate SQL statements). This value must comply with the same naming conventions described for field names.

`display_format`

Is the display format. The value must include the field type and length and may contain edit options.

The data type of the display format must be identical to that of the ACTUAL format. For example, a field with an alphanumeric USAGE data type must have an alphanumeric ACTUAL data type.

Fields or columns with decimal or floating-point data types must be described with the correct scale (s) and precision (p). Scale is the number of positions to the right of the decimal point. Precision is the total length of the field.

For the server, the total display length of the field or column includes the decimal point and negative sign. In SQL, the total length of the field or column excludes positions for the decimal point and negative sign.

For example, a column defined as DECIMAL(5,2) would have a USAGE attribute of P7.2 to allow for the decimal point and a possible negative sign.

`storage_format`

Is the storage format of the IDMS/SQL data type and length, in bytes, for the field. For more information on data type support, see the *iWay SQL Reference* manual.

`ON`

Displays the character specified by the NODATA parameter for missing data. For more information, see *MISSING Attribute* on page 12-13.

`OFF`

Displays blanks or zeroes for fields having no value. OFF is the default value. For more information, see *MISSING Attribute* on page 12-13.

MISSING Attribute

In a table, a null value represents a missing or unknown value; it is not the same as a blank or a zero. For example, a column specification that allows null values is used where a column need not have a value in every row (such as a raise amount in a table containing payroll data).

Note:

- The default NODATA character is a period (.).
- A column in an IDMS/SQL table that allows null data does not need to include the NULL clause in its table definition, since that is the default. In the Master File for that table, the column that allows null data must be described with the MISSING attribute value ON. The default for this attribute is OFF, which corresponds to the NOT NULL attribute in the IDMS/SQL table definition.
- Null data values appear as zeroes or blanks, if the column allows null data but the corresponding field in the Master File is described with the MISSING attribute value OFF.

Access Files

Each Master File must have a corresponding Access File. The file name of the Access File must be the same as that used for the Master File.

The Access File serves as a link between the server and the data source by providing the means to associate a segment in the Master File with the table it describes. The Access File minimally identifies the table and primary key. It may also indicate the logical sort order of data.

Syntax: How to Establish Segment Declarations

The segment declaration in the Access File establishes the link between one segment of the Master File and the actual IDMS/SQL table or view. Attributes that constitute the segment declaration are:

```
SEGNAME=segname, TABLENAME=[schema.]tablename, [,DBSPACE=storage,]
  [,KEYS={0|n}] [,KEYORDER={ASC|LOW|HIGH|DESC}]  ,$
```

where:

segname

Is the 1- to 8-character SEGNAME value from the Master File.

`schema`

Is the IDMS/SQL schema name for the table or view. It can consist of a maximum of 8 characters. If it is not specified, IDMS/SQL searches for a temporary table definition for the named table. If the named table does not exist, IDMS/SQL uses the current schema in effect for the current user session.

`tablename`

Is the name of the table or view. It can consist of a maximum of 18 characters.

Note: If any part of the TABLENAME includes a dollar sign ($), enclose that part in double quotation marks, and enclose the entire TABLENAME value in single quotation marks.

The maximum IDMS/SQL length for a fully qualified tablename is 36. All names must conform to the rules for identifiers stated in the *CA-IDMS/DB Release SQL Reference* manual.

`storage`

Is the IDMS/SQL segment and area name (in the form segment.area). If not specified, IDMS/SQL uses the default area associated with the schema. Enclose it in double quotation marks if it begins with a number or special character or contains special characters.

The Access File DBSPACE attribute overrides both the SET command and the installation default.

`n`

Is the number of columns that constitute the primary key. It can be a value between 0 and 64. 0 is the default value. For more information, see *Primary Key* on page 12-15.

`LOW (ASC)`

Indicates an ascending primary key sort sequence. LOW is the default value. ASC is a synonym for LOW.

`HIGH (DESC)`

Indicates a descending primary key sort sequence. DESC is a synonym for HIGH.

Primary Key

A table's primary key consists of the column or combination of columns whose values uniquely identify each row of the table. In the employee table, for example, every employee is assigned a unique employee identification number. Each employee is represented by one and only one row of the table, and is uniquely identified by that identification number.

The order of field declarations in the Master File is significant to the specification of key columns. To define the primary key in a Master File, describe its component fields immediately after the segment declaration. You can specify the remaining fields in any order. In the Access File, the KEYS attribute completes the process of defining the primary key.

To identify the primary key, the adapter uses the number of columns (n) indicated by the KEYS attribute in the Access File and the first n fields described in the Master File.

Typically, the primary key is supported by the creation of a unique index in the SQL language to prevent the insertion of duplicate key values. The adapter itself does not require any index in the column(s) comprising the primary key (although a unique index is certainly desirable for both data integrity and performance reasons).

Data Type Support

The following chart provides information about the default mapping of IDMS/SQL data types to server data types:

SQLIDMS Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
Binary(1..32,760)	A1	A1	*n*<257
Character(1..32,760)	A*n*	A*n*	*n* up to 32000
Date	YYMD	DATE	
Decimal (1..31,0..31)(*n,m*)	P(*n*+2,*m*)	P8	*n*<=14
Decimal (1..31,0..31)(*n,m*)	P(*n*+2,*m*)	P16	*n*>14
Double Precision	D20.2	D8	
Float (1..56)(*n*)	F9.2	F4	*n*<=24
Float (1..56)(*n*)	D20.2	D8	*n*>24
Graphic (1..16380)(*n*)	A(2**n*)	K*n*	
Integer	I11	I4	

SQLIDMS Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
Longint/BIGINT	I11	I4	
Numeric (1..31,0..31)(*n*,*m*)			Same as decimal
Real	F9.2	F4	
Smallint	I6	I4	
Time	A8	A8	
Timestamp	A26	A26	
Varchar (1..32760)(*n*)	A*n*	A*n*	*n*<=32000
Vargraphic (1..16379)(*n*)	TX	TX	

Changing the Precision and Scale of Numeric Columns

How to:

Override Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLIDM] SET CONVERSION RESET
ENGINE [SQLIDM] SET CONVERSION format RESET
ENGINE [SQLIDM] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLIDM] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLIDM`

Indicates the Adapter for IDMS/SQL. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`REAL` which indicates that the command applies only to single precision floating point columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT and REAL data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	**MAX Precision**
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER fields to 7:

```
ENGINE SQLIDM SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLIDM SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER fields to the default:

```
ENGINE SQLIDM SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLIDM SET CONVERSION RESET
```

Customizing the IDMS/SQL Environment

In this section:

The Adapter for IDMS/SQL provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Setting the User-specific Schema Name

The Adapter for IDMS/SQL uses the user-specified schema name as the first qualifier for all SQL requests involving SQL tables or views. This command overrides the IDMS/SQL current schema in effect and precludes the specification of unqualified table names. This prevents passing of unqualified table names to IDMS/SQL.

Syntax: How to Set SESSION

```
ENGINE [SQLIDMS] SET SESSION CURRENT SCHEMA schema
```

where:

`SQLIDMS`

Indicates the Adapter for IDMS/SQL. You can omit this value if you previously issued the SET SQLENGINE command.

`schema`

Is the name of the schema in SQL requests.

Controlling Transactions

IDMS/SQL protects data being read by one user from changes (INSERT, UPDATE, or DELETE) made by others; the Isolation Level setting governs the duration of the protection. That is, the Isolation Level determines when shared locks on rows are released, so that those rows or pages become available for updates by other users. IDMS/SQL allows you to dynamically set the Isolation Level within the server session using the IDMS/SQL SET TRANSACTION command.

The SET TRANSACTION CURSOR STABILITY or TRANSIENT READ command affects the duration of row or page shared locks on IDMS/SQL tables for the duration of the IDMS/SQL transaction. You can specify the command within a stored procedure. The setting remains in effect for the server session or until you reset it.

Syntax: How to Control Transactions

```
ENGINE [SQLIDMS] SET TRANSACTION
 {CURSOR STABILITY|TRANSIENT READ|READ ONLY|READ WRITE}
```

where:

`SQLIDMS`

Indicates the Adapter for IDMS/SQL. You can omit this value if you previously issued the SET SQLENGINE command.

`CURSOR STABILITY`

Provides the maximum amount of concurrency while guaranteeing the integrity of the data selected. CURSOR STABILITY is the default value.

`TRANSIENT READ`

Allows the reading of records locked by other users. This is recommended for SQL request only. Transient read prevents the SQL transaction from performing updates. Use this only when you do not need the data retrieved to be absolutely consistent and accurate. If you specify Transient Read, IDMS/SQL assumes it is read-only.

`READ ONLY`

Allows data to be retrieved, but does not allow the data source to be updated.

`READ WRITE`

Allows data to be retrieved, and allows the data source to be updated.

Designating a Default Tablespace

You can use the SET DBSPACE command to designate a default tablespace for tables you create.

For the duration of the session, the adapter places these tables in the IDMS/SQL tablespace that you identify with the SET DBSPACE command. If the SET DBSPACE command is not used, IDMS/SQL uses the default tablespace for the connected user.

Syntax: How to Set DBSPACE

```
ENGINE [SQLIDMS] SET DBSPACE storage
```

where:

`SQLIDMS`

Indicates the Adapter for IDMS/SQL. You can omit this value if you previously issued the SET SQLENGINE command.

`storage`

Is the segment.area name that will contain the IDMS/SQL tables created by the CREATE FILE command. If a name is not specified, the table will be placed in the IDMS/SQL default area for the current schema in effect for the user's SQL session.

Note: This command will only affect CREATE TABLE requests made by Table Services. It does not affect Passthru CREATE TABLE commands.

Overriding Default Parameters for Index Space

How to:

Set IXSPACE

Example:

Specifying Segment.Area Name and Index Block

You can use the SET IXSPACE command to override the default parameters for the IDMS/SQL index space implicitly created by the CREATE FILE and HOLD FORMAT SQLIDMS commands.

Syntax: How to Set IXSPACE

```
ENGINE [SQLIDMS] SET IXSPACE [index-spec]
```

where:

`SQLIDMS`

Indicates the Adapter for IDMS/SQL. You can omit this value if you previously issued the SET SQLENGINE command.

`index-spec`

Is the portion (up to 94 bytes) of the SQL CREATE INDEX statement beginning with IN segment.area (as specified in the *CA-IDMS/DB Reference* CREATE INDEX syntax diagram).

Note: To reset to the IDMS/SQL default index space parameters, issue the SET IXSPACE command with no operands.

Example: Specifying Segment.Area Name and Index Block

The following example shows how to set the segment.area name and INDEX BLOCK of the CREATE INDEX statement with IXSPACE:

```
ENGINE SQLIDMS
SET IXSPACE IN EMPSEG.EMPAREA INDEX BLOCK CONTAINS 5 KEYS
END
```

You can use the SQL ? query command to determine the current IXSPACE setting. If the current setting is the default, IXSPACE does not display in the SQL SQLIDMS ? output.

Note: This command will only affect CREATE INDEX requests made by Table Services. It does not affect Passthru CREATE INDEX commands.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLIDMS] SET PASSRECS {ON|OFF}
```

where:

`SQLIDMS`

Indicates the Adapter for IDMS/SQL. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides an informational message after the successful execution of an SQL Passthru UPDATE or DELETE command. ON is the default value.

`OFF`

Does not provide a message after the successful execution of an SQL Passthru UPDATE or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Note that since, by definition, the successful execution of an INSERT command always affects one record, INSERT does not generate this information.

Optimization Settings

In this section:

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

How to:

Example:

Reference:

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLIDM] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLIDM`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

`setting`

Is the optimization setting. Valid values are as follows:

`OFF` instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

<u>`ON`</u> instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

`FOCUS` passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

`SQL` passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLIDM set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLIDM set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLIDM] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLIDM`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLIDM SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLIDM SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLIDM SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

  Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Specifying Block Size for Retrieval Processing

How to:

Specify the Block Size for Array Retrieval

The Adapter for IDMS/SQL supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Specify the Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE [SQLIDM] SET FETCHSIZE n
```

where:

`SQLIDM`

Indicates the Adapter for IDMS/SQL. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

CHAPTER 13

Using the Adapter for IMS

Topics:

The Adapter for IMS allows applications to access IMS data sources. The adapter converts application requests into native IMS statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

The Adapter for IMS is supported on the OS/390 and z/OS platform and uses the IMS DBCTL environment to connect to IMS databases.

Preparing the IMS Environment

Before configuring the Adapter for IMS, verify that your site is running IMS Version 3.1 or higher and that all APPLCTN macros describing PSBs that will be accessed by multiple users specify SCHDTYP=Parallel. If necessary, modify the APPLCTN macros for such PSBs to include the attribute SCHDTYP=Parallel, and rerun the IMS SYSGEN to make the changes effective.

The Adapter for IMS connects using the DBCTL environment. For the connection to be made, the following libraries must be identified and allocated to the STEPLIB in the ISTART JCL member of *qualif.release.servertype*.DATA:

- The DFHPZP library.
- The SDFSRESL library.

Note: *qualif* is provided by the user. The *release* and *servertype* vary with the release and license key being used. Refer to the Server Installation for OS/390 and z/OS in the *iWay Server Installation* manual for further information on the configuration dataset naming convention.

The PZP library must have a member that has been generated to identify the correct DBCTL environment for the connection. The following steps must be completed before the server can be configured, using the Web Console, for the Adapter for IMS:

1. Create the DRA startup table.
2. Assemble and link the DRA startup table.
3. Verify that the DBCTL is operational.
4. Establish security.

Syntax: How to Create the DRA Startup Table: DFSPZPxx

The Database Resource Adapter (DRA) is the interface between a user task and DBCTL. The DRA Startup Table contains values that define the characteristics of the DRA. The name of the DRA Startup Table is

```
DFSPZPxx
```

where:

`xx`

Is a two-character suffix that should be chosen to comply with your site's standards.

Once this suffix has been chosen, its value is needed during the configuration phase using the Web Console.

Example: Assembling and Linking the DRA Startup Table

This sample JCL illustrates how to assemble and link the DFHPZPxx member. You can normally find sample JCL in your IMS installation library.

```
//job card goes here
//ASSEMBLE EXEC  PGM=IEV90,REGION=2M,PARM='OBJECT,NODECK'
//SYSLIB     DD DSN=IMS.MACLIB,DISP=SHR
//SYSLIN     DD UNIT=SYSDA,DISP=(,PASS),
//                SPACE=(80,(100,100),RLSE),
//                DCB=(BLKSIZE=80,RECFM=F,LRECL=80)
//SYSPRINT   DD SYSOUT=*,DCB=BLKSIZE=1089
//SYSUT1     DD UNIT=SYSDA,DISP=(,DELETE),
//                SPACE=(CYL,(10,5))
//SYSIN      DD    *
PRP      TITLE 'DATABASE RESOURCE ADAPTER STARTUP PARAMETER TABLE'
DFSPZPxx CSECT
         EJECT
         DFSPRP DSECT=NO,                                             X
                DBCTLID=IMS3,                                         X
                DDNAME=DFSRESLB,                                      X
                DSNAME=IMS.RESLIB,                                    X
                CNBA=150,                                             X
                MAXTHRD=150,                                          X
                MINTHRD=5
         END
/*
//*
//LINK      EXEC  PGM=IEWL,PARM='XREF,LIST',COND=(0,LT,ASSEMBLE),
//                  REGION=4M
//SYSLIN     DD DSN=*.ASSEMBLE.SYSLIN,DISP=(OLD,DELETE)
//SYSPRINT   DD SYSOUT=*,DCB=BLKSIZE=1089
//SYSUT1     DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),
//                SPACE=(1024,(100,10),RLSE),DISP=(,DELETE)
//SYSLMOD    DD DISP=SHR,DSN=qualif.release.HOME.LOAD(DFSPZPxx)
```

where:

qualif

Is the high level qualifier for the data sets.

xx

Is the two character suffix chosen for the DRA Startup Table.

Note: The *qualif.release*.HOME.LOAD library must be concatenated ahead of IMS.RESLIB in the allocation for DDNAME STEPLIB in *qualif.release.servertype*.DATA(ISTART).

Reference: Keywords

The following chart describes the keywords required to generate the DFHPZP library. Other keywords that affect your IMS environment, and may be required at your site, are described in the *IBM IMS System Definition Reference*.

Keyword	**Description**	**Default**
`AGN`	Is a 1 to 8 character application group name used as part of the DBCTL security function. For more information on DBCTL security, see the *IMS System Administration Guide*.	N/A
`CNBA`	Is the total number of Fast Path buffers for the server's use. For a description of Fast Path DEDB buffer usage, see the *IMS System Administration Guide*. You can omit this parameter if your site does not utilize Fast Path.	N/A
`DBCTLID`	Is the 4–character name of the DBCTL region. This is the same as the IMSID parameter in the DBC procedure. For more information on the DBC procedure, see the *IBM IMS System Definition Reference*.	`SYS1`
`DDNAME`	Must be DFSRESLB. It is the DDNAME that will be allocated to the DBCTL RESLIB library (see the DSNAME keyword).	N/A
`DSNAME`	Is the 1– to 44–character data set name of the DBCTL RESLIB library. It must contain the DRA modules and be MVS authorized.	`IMS.RESLIB`
`FPBOF`	Is the number of Fast Path DEDB overflow buffers to be allocated per thread. For a description of Fast Path DEDB buffer usage, see the *IMS System Administration Guide*. You can omit this keyword if your site does not use Fast Path.	00
`FPBUF`	Is the number of Fast Path DEDB buffers to be allocated and fixed per thread. For a description of Fast Path DEDB buffer usage, see the *IMS System Administration Guide*. You can omit this parameter if your site does not utilize Fast Path.	00
`FUNCLV`	Is the level of the DRA that the CCTL supports. FUNCLV=1 means the CCTL uses the DRA at the IMS 3.1 level.	1
`MAXTHRD`	Is the maximum number of concurrent DRA threads. The value cannot exceed 255.	1

Keyword	Description	Default
MINTHRD	Is the minimum number of concurrent DRA threads available. Since the number of threads specified by this keyword will be allocated at all times, regardless of whether they are used, be careful when selecting this value. The value cannot exceed 255.	1
SOD	Is the output class to use for a SNAP DUMP of abnormal thread termination.	A
TIMEOUT	Is the number of seconds a CCTL should wait for the successful completion of a DRA TERM request. Specify this value only if the CCTL is coded to use it. This value is returned to the CCTL upon completion of an INIT request.	60
TIMER	Is the number of seconds between attempts of the DRA to identify itself to DBCTL during an INIT request.	60
USERID	Is the 8–character name of the CCTL region. No two CCTLs (servers accessing the same IMS region through DBCTL) can have the same USERID (address space ID).	N/A

Syntax: How to Verify the DBCTL Is Operational

The following JCL can be used to verify that the DBCTL is properly installed. Run this job after making any changes necessary to the JCL to suit your site's standards and naming conventions.

```
//*********************************************************************
//* Substitutions:
//*
//*       qualif   Change to the high level qualifier
//*                for your data sets.
//*
//*       edapsb1  Change to any PSB name.
//*
//*       pcicspsb RACF class to be used in the test. If this
//*                parameter is omitted, security checking will
//*                not be performed.
//*
//*       xx       Is the 2 character suffix that identifies the DRA
//*                startup module.
//*
//*********************************************************************
//STEP1     EXEC  PGM=IMDTEST,PARM='emppsb1,pcicspsb,xx',
//                  REGION=4096K,TIME=(1,5)
//STEPLIB   DD    DSN=qualif.release.HOME.LOAD,DISP=SHR
//          DD    DSN=IMS.RESLIB,DISP=SHR
//SYSIN     DD    DUMMY
//SYSOUT    DD    SYSOUT=*
//SYSPRINT  DD    SYSOUT=*
```

where:

emppsb1

Is a PSB name.

pcicspsb

Is a security class.

xx

Is a 2-character suffix that identifies the DRA Startup Module to be loaded as described in *Create the DRA Startup Table: DFSPZPxx* on page 13-3.

Note: If you omit this parameter, its value defaults to 00, which may cause IMS to load the incorrect module.

qualif

Is the high level qualifier for the data sets.

After running the JCL, examine the JES output. The return codes displayed on the following lines indicate that DBCTL is properly installed:

```
12.45.47 JOB05736  +CCTL: JOBNAME(USERID1) PSB(EMPPSB1) CLASS(PCICSPSB)
12.45.47 JOB05736  +CCTL: INIT CALLED
12.45.47 JOB05736  +CCTL: INIT RETURNED,RETC(00000000)
12.45.47 JOB05736  +CCTL: WAITING FOR CONTROL EXIT
12.45.47 JOB05736  +CTLX: CONTROL EXIT CALLED
12.45.47 JOB05736  +CTLX: CONTROL EXIT RETURNED
12.45.47 JOB05736  +CCTL: INIT COMPLETED, FUNC(02) SFNC(00) RCOD(00)
RETC(00000000)
12.45.47 JOB05736  +CCTL: SECURITY CHECK COMPLETED, RC(00000004)
12.45.47 JOB05736  +CCTL: TERMINATE DRA CALLED
12.45.47 JOB05736  +CTLS: SUSPEND EXIT CALLED
12.45.47 JOB05736  +CTLR: RESUME EXIT CALLED
12.45.47 JOB05736  +CTLS: SUSPEND EXIT RETURNED
12.45.47 JOB05736  +CTLR: RESUME EXIT RETURNED
12.45.47 JOB05736  +CCTL: TERMINATE DRA COMPLETED, RETC(00000000)
```

Note that the installation is successful even when the security check completes with a return code of 4.

If the JES output indicates an error, make sure that a DRA Startup Table with a suffix of 00 exists. See *Create the DRA Startup Table: DFSPZPxx* on page 13-3. If it exists, verify that:

- The parameters specified in it (especially DSNAME and DDNAME) are correct.
- The DBCTLID keyword points to the correct IMS system.
- DBCTL is installed at your site, and that it is functioning correctly.

Establishing Security

How to:

Define a PSB Resource

The DBCTL environment, when accessed through the server, enables the use of security systems through the standard SAF interface. With the SAF interface, your site can use security products such as RACF, CA TOP SECRET, and CA ACF2 to restrict access to PSBs. Before allowing access to a particular PSB, the security system verifies that the user is authorized to read the PSB.

Note: The DBCTL function is tested and verified with the RACF product. Other SAF products using identical calls should perform properly when installed and verified by your site's security administrator.

RACF comes with several predefined security classes. Customer sites can use an existing class (such as PCICSPSB) or define a resource class specifically for DBCTL use.

Syntax: How to Define a PSB Resource

The following syntax illustrates how to define a PSB resource through a PCICSPSB profile to RACF, and how to grant users permission to access the resource.

```
RDEFINE PCICSPSB (psbname) UACC(NONE) NOTIFY(sys_admin_userid)
PERMIT psbname CLASS(PCICSPSB) ID(user_or_group [user_or_group ...])
ACCESS(READ)
```

where:

psbname

Is a PSB name to be protected by RACF.

sys_admin_userid

Is the user ID of the system administrator.

user_or_group

Authorizes the user IDs and/or user groups listed in the PERMIT command to read the specified PSB. Separate items in the list with blanks.

At run time, after the PSB is selected, but prior to scheduling it, the server issues a call to the security system and verifies that the user is authorized to read the PSB.

Configuring the Adapter for IMS

In this section:

Dynamic Setting of PSB

How to:

Declare Connection Attributes From the Web Console

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. You can declare connection attributes from the Web Console.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, select *Data Adapters*.
2. Expand the *Add* folder, expand the *Other DBMS* group folder, then expand the *IMS* folder and click a connection. The Add IMS to Configuration pane opens.
3. Supply values for the following parameters:

Field	Description
IMS Sec	ON activates DBCTL security. This setting requires that the server be in authorized mode. OFF does not implement DBCTL security.
IMS DBCTL	START.
IMS Class	Points to a valid class that contains the security rule. The Systems Administrator can define any class name required by the security implementation of the site. The default value is PCICSPSB.
IMS PZP	The suffix of the DFSPZP module created during the generation of the Adapter for IMS/DBCTL.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

   Tip: Once the adapter is configured, the connection name appears in the navigation pane under *Configured*. If you left-click either the adapter name or the connection name, an appropriate set of options becomes available.

5. Restart the server so the connection to DBCTL can be initialized.

If you do not have existing synonyms (Master Files), proceed to *Managing IMS Metadata* on page 13-11. If existing synonyms are available through JCL allocations, the Adapter for IMS is ready for use.

Dynamic Setting of PSB

The PSB name to use for access can be set dynamically with the following command

```
ENGINE IMS SET PSB psbname
```

where:

psbname

Is the PSB name you wish to dynamically set.

Managing IMS Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the IMS data types.

For the Adapter for IMS to access IMS files, you must describe each IMS file you use in a Master File and Access File. The logical description of an IMS file is stored in a Master File, which describes the field layout.

Creating Synonyms

How to:

Create a Synonym From the Web Console

Create a Synonym Manually

Example:

Using CREATE SYNONYM

Reference:

Managing Synonyms

Customization Options for COBOL File Descriptions

CHECKNAMES for Special Characters and Reserved Words

Defining Subsets of DBD Segments and IMS Fields in Segments

Synonyms define unique names (or aliases) for each IMS file that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. For IMS the connection is *local*.

   The right pane displays selection options for the adapter.

4. Choose your selection options:

   - **Select All Members.** Select this option button to display a drop-down list containing the names of all the PSBs in the designated PSBLIB.

   - **Filter by Member Name.** Select this option button to filter the members for which you wish to create synonyms. Selecting this option adds a Member Name input field.

     Enter a string for filtering the Members, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter ABC% to select Members which begin with the letters ABC; %ABC to select Members which end with the letters ABC; %ABC% to select Members which contain the letters ABC at the beginning, middle, or end.

5. Continuing in the Step 1 of 4 pane, enter the following additional parameters to specify the fully qualified library names:

Parameter	**Definition**
DBD library name	Is the MVS library name that corresponds to the IMS DBDLIB. This must be a fully qualified MVS dataset name.
PSB library name	Is the MVS library name that corresponds to the IMS PSBLIB. This must be a fully qualified MVS dataset name.

6. Click *Next*. The PSB selection pane (Step 2 of 4) opens.

7. Select a PSB member name from the drop-down list.

8. Click *Next*. The PCB selection pane (Step 3 of 4) opens.

9. Select the *Map using COBOL FDs* option if you have a Cobol FD library available that describes the PCB view, or select the *Create Skeleton* option button.

   Note: If you select *Create skeleton*, the final screen opens without the option to select a Cobol FD entry. Create Synonym then creates a skeleton synonym using the Data Base Definition defined fields. In this case, click *Create Synonym* to complete the process.

10. Click *Fully qualified PDS name* and enter a fully qualified MVS Library in the entry box.

    or

    Click the *Full HFS path* and enter the HFS location that contains the COBOL FD.

11. Select the PCB you want to use by clicking the option button to the left of the synonym name.

12. Click *Next*. The final pane opens.

13. Enter the following parameters:

Parameter	**Description**
Select Application	Select an application directory in which to create the synonym. The default value is baseapp.
Prefix/Suffix	If you have identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all operations have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

14. From the drop-down list under the COBOL FD column, select the COBOL FD that will be mapped to each IMS segment name.

15. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

    When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

16. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

17. Optionally, select *Customize options* to customize how the COBOL FD is translated. For details about these options, see *Customization Options for COBOL File Descriptions* on page 13-16. If you do not select the check box, default translation settings are applied.

18. Click *Create Synonym*. The result of the Create Synonym process will be a Master File and an Access File. A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

19. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.

Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: Customization Options for COBOL File Descriptions

You can customize how a COBOL FD is translated by selecting *Customize options* in the Synonym creation pane. The following options are added to the right pane:

Parameter	**Definition**
On Error	Choose *Continue* to continue generating the Master File when an error occurs. Choose *Abort* to stop generating the Master File when an error occurs. Choose *Comment* to produce a commented Master File when an error occurs. Continue is the default value.
Hyphens as	Choose *No* to remove all hyphens in the COBOL name from the Master File field names. Choose *Yes* to replace all hyphens in the COBOL name with the underscore character. *Yes* is the default value.
Redefines	You may treat COBOL REDEFINE fields in one of three ways: Choose *Segments* to describe REDEFINE fields as segments in the Master File. Segments is the default value. Choose *Comments* to describe REDEFINE fields as comments in the Master File. Choose *None* to exclude REDEFINE fields altogether.
Occurs as	Choose *Segments* to describe OCCURS structures as segments, or *Field* otherwise. Segments is the default value.
Alignment	Choose *Yes* to insert slack bytes into a record to ensure alignment of numeric fields. Choose *No* to generate Master Files without alignment of slack bytes. No is the default value.

Parameter	**Definition**
Number of Hyphens to skip	FD Translator removes characters from the left, up to and including the Nth hyphen (depending on the value of N chosen in the menu). A value of *0* will retain the entire COBOL name. *All* means all prefixes will be removed. *0* is the default value.
Order fields	Choose *Yes* to generate Order fields in a Master File. Choose *No* to generate a Master File without Order fields. *No* is the default value.
Level 88 as	Choose *Comment* to include COBOL Level 88 fields as comments in the Master Files. Choose *Skip* to exclude level 88 fields. *Skip* is the default value.
Zoned Numeric Fields	Sets how zoned numeric values will be stored.
Numeric Fields Edit Options	
Zeroes	Choose *Suppress* to suppress printing of the digit zero for a field whose value is zero. Choose *Display* to display leading zeroes, for example, 00124. Choose *None* for no formatting.
Negative value	Choose *Bracket* to bracket negative values, for example, (1234). Choose *Credit* to credit negative values, for example, 1234 CR. Choose *None* for no formatting.
Dollar Sign	Choose *Floating* to display a floating dollar sign and commas, for example, $1,1234. Choose *Fixed* to display a fixed dollar sign and commas, for example, $ 1,1234. Choose *None* for no formatting.

Parameter	**Definition**
Separate Thousands	Choose *Comma* to include commas where appropriate. Choose *None* for no formatting.

For additional information about customization options, see Appendix D, *Translating COBOL File Descriptions*.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '| ', '&', ' (', ') ', ' <', '> ', '' ', ' =', '''

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR psbname DATABASE 1,2,3,etc. DBMS IMS
AT psblib/dbdlib PARMS [" < string 'fdparms' > "]
[CHECKNAMES][UNIQUENAMES]
END
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used.

If your server is not APP enabled, this application name must not be used.

`FOR`

Indicates the *psbname* that contains a PCB reference to the database being processed.

`DATABASE`

Refers to the numeric position of the database PCB within the PSB.

`AT`

Indicates the corresponding MVS dataset names for the PSBLIB and the DBDLIB. Both must be specified. The dataset names must be fully qualified MVS loadlib names and are delimited with a slash.

`PARMS`

Contains the location of the individual corresponding COBOL FD (within double quotes); the associated database segment; and parameters specific to COBOL FDs (within single quotes).

The location may be an HFS file—/u/pgmjsk/fds/patinfo/patinfo.cbl, or an MVS PDS (partitioned dataset)—'PMSERH.TSO.SRCE(PATINFO)'. The syntax is

```
<fd name><comma><segment name><space><fd name><comma><segment name>
etc.
```

fdparms

Indicates COBOL FD specific parameters, such as:

```
[SKIPHYPHEN  = {<number> | all}]
                [UNDERBARS]
                [ORDER        = {Y | N}]
                [REDEFINE     = {comment | segm}]
                [LEVEL88      = {comment | skip}]
                [OCCURS       = {segm | field}]
                [EDITFIELDS   = {S,C,B,R,M,N,L}]
                [ZONEFIELDS   = {packed | alpha}]
```

If no FDPARMS are entered, the underlined defaults will be in effect.

`CHECKNAMES`

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

`UNIQUENAMES`

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

CREATE SYNONYM creates a Master File and an Access File, which represent the server metadata.

Example: Using CREATE SYNONYM

Use the following syntax to create a synonym for the variable "AIHDAM".

```
CREATE SYNONYM AIHDAM FOR AIHDPSB DATABASE 4 DBMS IMS AT
IMS.V7R1M0.B.PSBLIB/IMS.V7R1M0.B.DBDLIB
      PARMS "//'PMSERH.TSO.SRCE(AIHDAM)',AIHDAM
      FDPARMS = 'SKIPHYPHEN 0 UNDERBARS ONERROR C ORDER N REDEFINE S
LEVEL88 N OCCURS Y ZONEFIELDS A'"
END
 CREATE SYNONYM AIHDAM FOR AIHDPSB DATABASE 4 DBMS IMS AT
IMS.V7R1M0.B.PSBLIB/IMS.V7R1M0.B.DBDLIB
      PARMS "/u/pgmjsk/fds/aihdam,AIHDAM
      FDPARMS = 'SKIPHYPHEN 0 UNDERBARS ONERROR C ORDER N REDEFINE S
LEVEL88 N OCCURS Y ZONEFIELDS A'"
END
```

Generated Master File aihdam.mas

```
FILENAME=AIHDAM, SUFFIX=IMS     , $
SEGMENT=LANGUAGE, SEGTYPE=S0, $
$  GROUP=AIHDAM_IO, USAGE=A19, ACTUAL=A19, $
FIELDNAME=EMPL6, ALIAS=EMPL6.HKY, USAGE=A4, ACTUAL=A4, $
FIELDNAME=LANG6, ALIAS=LANG6.IMS, USAGE=A15, ACTUAL=A15, $
GROUP=IXEMP6.SKY, ALIAS=IXEMP6.SKY, USAGE=A4, ACTUAL=A4, $
FIELDNAME=IX_EMPL6   , ALIAS=EMPL6, USAGE=A4, ACTUAL=A4, $
```

Generated Access File aihdam.acx

```
PSB=AIHDPSB,PCBNUMBER= 4,PL1=NO,WRITE=NO,$
SEGNAME=LANGUAGE, KEYTYPE=S1   ,$
ALTPCBNAME=IXEMP6   , ALTPCBNUMBER=5,$
```

Reference: Defining Subsets of DBD Segments and IMS Fields in Segments

The SENSEG parameter is used in a PCB description to define a subset of accessible DBD segments. CREATE SYNONYM produces Master File segments based on the SENSEG information obtained from the PSB, and not from the database definition.

The SENSFLD parameter is used in a PCB description to define a subset of accessible IMS fields in a segment. IMS fields are defined in the DBD. The list of fields in a Master File SEGMENT is generated based on the SENSFLD information, and not on the DBD field definitions.

SENSFLD macros can redefine the layout of an IMS database segment record by using the "START = xx" parameter. Different layouts can be defined in the DBD and SENSFLD. The Master File segment layout (field information) is derived from the SENSFLD.

Master File Attributes

Each Master File that provides access to an IMS data source describes the segments and fields that are available through one IMS PCB.

Note: You do not have to describe every segment from the PCB in the Master File. However, the portion of the hierarchy you describe must be a subtree starting from the root. Any segment or field that you do not describe in the Master File remains invisible to the server.

Reporting costs are largely a function of the volume of data transferred. Therefore, requests issued through the adapter are efficient and cost effective, because only segments referenced in your request are retrieved.

In a Master File, you can describe up to 64 segments across 15 levels. The cumulative length of all fields across all segments cannot exceed 12,000 bytes (an IMS restriction). See *How to Specify IMS Field Attributes* on page 13-25. for additional information about this limit.

Each Master File is stored as a member of a Master File PDS. The member name for a Master File must be the name assigned to that Master File in the Access File record for the corresponding PCB. At run time, the Master File data set is allocated to ddname MASTER.

A Master File consists of file, segment, and field declarations. Rules for declarations are:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,$). Text appearing after the comma and dollar sign is treated as a comment.
- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas separate attribute=value pairs.

Syntax: How to Identify Master Files for IMS

Each Master File begins with a file declaration that names the file and describes the type of data source—an IMS data source in this case. The file declaration has two attributes, FILENAME and SUFFIX.

```
FILE[NAME]=name, SUFFIX=IMS [,$]
```

where:

`name`

Is any 1- to 8-character name.

`IMS`

Indicates that the Adapter for IMS is required for data retrieval.

Syntax: How to Specify IMS Segment Attributes

Each IMS segment described in a Master File requires a segment declaration that consists of at least two attributes, SEGNAME and SEGTYPE. The SEGNAME value is the name of the corresponding IMS segment. The SEGTYPE value identifies the segment's characteristics.

```
SEGNAME=segname, SEGTYPE=segtype [,PARENT=parent] [,$]
```

where:

`segname`

Is the 1- to 8-character IMS segment name from the NAME parameter of the SENSEG record in the IMS PCB.

`segtype`

Identifies the segment's characteristics. Possible values are:

`S0` indicates that the segment is data sensitive and has no key.

`S` or `S1` indicates that the segment is data sensitive and has a non-unique key.

`S2` indicates that the segment is data sensitive and has a unique key.

`SH` or `SH1` indicates that the segment is key sensitive and has a non-unique key.

`SH2` indicates that the segment is key sensitive and has a unique key.

`parent`

Is the name of the parent segment from the IMS data source. Its value comes from the PARENT parameter of the SENSEG record in the IMS PCB.

SEGNAME in IMS

The SEGNAME attribute identifies the IMS segments you can access. The SEGNAME value is the name of the IMS segment from the SENSEG record in the PCB.

The server retrieves segments in top-to-bottom left-to-right sequence as described by the Master File. Therefore, the order of segments in the Master File should be the same as their order in the PCB to maintain the correct hierarchical sequence.

SEGTYPE in IMS

The SEGTYPE attribute:

- Identifies the characteristics of the segment's key field. A segment can have a unique key, a non-unique key, or no key.

  If a segment has a unique key, at least one FIELD record in the DBD must specify NAME=(fieldname,SEQ,U) or NAME=(fieldname,SEQ). Similarly, if the segment has a non-unique key, at least one FIELD record in the DBD must specify NAME=(fieldname,SEQ,M). If no FIELD record in the DBD specifies a SEQ attribute, the segment has no key. The key referred to by the SEGTYPE attribute may actually be a secondary index (see *Segment Redefinition in IMS: The RECTYPE Attribute* on page 13-31).

  The characteristics of the segment's key determine the types of SSAs and the number of DL/I calls the adapter must issue to satisfy a retrieval request. To handle keys made up of multiple fields, describe the multiple fields as a group in the Master File (for details, see *How to Issue a GROUP Field in IMS* on page 13-29).

- Identifies whether the data from the segment can be retrieved (data sensitive) or if only the segment's key is accessible (key sensitive). For a key sensitive segment, the SENSEG record in the PCB includes the parameter PROCOPT=K. Key sensitive segments are used for access to lower level segments.

The following chart lists the valid SEGTYPE values:

SEGTYPE	Definition	PROCOPT=K In PCB?	NAME= From DBD
`S0`	Data sensitive, no key	No	(name)
`S or S1`	Data sensitive, non-unique key	No	(name,SEQ,M)
`S2`	Data sensitive, unique key	No	(name,SEQ,U) or (name,SEQ)
`SH or SH1`	Key sensitive, non-unique key	Yes	(name,SEQ,M)
`SH2`	Key sensitive, unique key	Yes	(name,SEQ,U) or (name,SEQ)

PARENT in IMS

The PARENT attribute identifies the segment's parent in the hierarchy. It appears in the PARENT parameter of the SENSEG record in the PCB. The only exception is in the root segment, where the PCB either omits the PARENT parameter or specifies PARENT=0. In the Master File, you can specify the PARENT attribute of the root segment as PARENT= , or you can omit it.

Syntax: How to Specify IMS Field Attributes

Each segment consists of one or more fields. The IMS DBD contains FIELD declarations for all sequence (key) and search fields, but other fields are optional.

If the PCB you are describing contains SENFLD records for a segment, the Master File can view only fields explicitly specified in those SENFLD records.

However, if the PCB does not contain any SENFLD records for a segment, you can describe the entire segment in the Master File. You can get information about sequence and search fields from the DBD. To describe other fields, you may have to refer to an external description of the segment, for example, a COBOL FD.

The Master File need not describe all fields from a segment, but it must include an initial subset of the segment (that is, it must start from the beginning and not contain any gaps).

To describe a field in the Master File, you must specify the primary attributes FIELDNAME, ALIAS, USAGE, and ACTUAL. These attributes are discussed in this topic. Note that the adapter does not support the MISSING attribute.

```
FIELD[NAME]=field,[ALIAS=]alias,[USAGE=]display,[ACTUAL=]imsformat ,$
```

where:

field

Is a 1- to 66-character field name. In requests, the field name can be qualified with the Master File and/or segment name. Although the qualifiers and qualification characters do not appear in the Master File, they count toward the 66-character maximum.

alias

Is the alias for a type of field. Possible values are:

imsfield.KEY, the alias for a field that is an IMS key field. Form the alias by appending the suffix KEY to the name of the IMS field.

imsfield.IMS, the alias for a field that is an IMS search field. Form the alias by appending the suffix IMS to the name of the IMS field.

imsfield.HKY, the alias for a field that is the key of the root segment in an HDAM data source. Form the alias by appending the suffix HKY to the name of the IMS field.

`display`

Is the server display format for the field.

`imsformat`

Is the server definition of the IMS field format and length (n).

You can omit the ALIAS, USAGE, and ACTUAL keywords from the field declaration if the values are specified in the standard order (FIELD, ALIAS, USAGE, ACTUAL). For example, the following declarations are equivalent:

```
FIELD = YEAR, ALIAS=, USAGE=A2, ACTUAL=A2,$
FIELD = YEAR, ,A2, A2,$
```

FIELD NAME in IMS

Field names can consist of a maximum of 66 alphanumeric characters. IMS field names are acceptable values if they meet the following naming conventions:

- Names can consist of letters, digits, and underscore characters. Special characters and embedded blanks are not advised.
- The name must contain at least one letter.
- Duplicate field names (the same field names and aliases) within a segment are not permitted.

Since field names appear as default column titles for reports, select names that are representative of the data.

Note: You can only specify field names in an SQL request. You can not specify the ALIAS name.

ALIAS in IMS

The ALIAS value in the Master File distinguishes between fields that are defined in the IMS DBD and fields that are not defined to IMS. The adapter uses this information in constructing DL/I calls to IMS.

If a field name in a WHERE clause is a sequence or search field, the adapter may be able to create an SSA that instructs IMS to apply the screening test and return the appropriate records to the server. If the field is not defined in the DBD, the adapter must retrieve all records sequentially from IMS so that the server can screen them.

Note: In certain cases, the adapter can instruct IMS to screen values based on a secondary index. *Segment Redefinition in IMS: The RECTYPE Attribute* on page 13-31 describes the technique for taking advantage of a secondary index.

The ALIAS value for a field defined in the DBD is composed of:

- The name of the field as specified in the IMS DBD (1- to 8-characters in length).
- A separation character, the period (.).
- A suffix value that describes whether the field is a sequence field (suffix KEY), a search field (suffix IMS), or the key of the root segment of an HDAM database (suffix HKY).

Except for certain types of control field entries (for example, ORDER and RECTYPE), fields not defined in the DBD should not be assigned ALIAS names. For more information on ORDER and RECTYPE control fields, see *Segment Redefinition in IMS: The RECTYPE Attribute* on page 13-31.

USAGE in IMS

The USAGE attribute indicates the display format of the field. An acceptable value must include the field type and length and may contain edit options. The server uses the USAGE format for data display on reports. All standard USAGE formats (A, D, F, I, P) are available.

ACTUAL in IMS

How to:

Issue a GROUP Field in IMS

Example:

Defining a Group Key

The ACTUAL attribute indicates the server representation of IMS field formats.

For fields defined in the DBD (sequence and search fields), use the format specified in the DBD.

Use the following chart as a guide for describing ACTUAL formats of those fields not defined in the DBD:

COBOL Format	COBOL PICTURE	Bytes of Storage	ACTUAL Format	USAGE Format
DISPLAY	X(4)	4	A4	A4
DISPLAY	S99	2	Z2	P3
DISPLAY	9(5)V9	6	Z6.1	P8.1
DISPLAY	99	2	A2	A2
COMP	S9	4	I2	I1

COBOL Format	**COBOL PICTURE**	**Bytes of Storage**	**ACTUAL Format**	**USAGE Format**
COMP	S9(4)	4	I2	I4
COMP	S9(5)	4	I4	I5
COMP	S9(9)	4	I4	I9
COMP-1	-	4	F4	F6
COMP-2	-	8	D8	D15
COMP-3	9	1	P1	P1
COMP-3	S9V99	2	P2	P5.2
COMP-3	9(4)V9(3)	4	P4	P8.3
FIXED BINARY(7) (COMP-4)	B or XL1	4	I4	I7

Note: The USAGE lengths shown are minimum values. You can make them larger and add edit options. You must allow space for all possible digits, a minus sign for negative numbers, and a decimal point in numbers with decimal digits.

The cumulative length of all fields, across all segments, cannot exceed 12K bytes.

The following example illustrates the DI21PART Master File that provides access to the DI21PART data source.

```
FILE=DI21PART     ,SUFFIX=IMS,$
SEGNAME=PARTROOT ,PARENT=,SEGTYPE=S2,$
  FIELD=PARTKEY   ,ALIAS=PARTKEY.HKY   ,USAGE=A17   ,ACTUAL=A17   ,$
  FIELD=SKIP1     ,ALIAS=              ,USAGE=A33   ,ACTUAL=A33   ,$
SEGNAME=STANINFO ,PARENT=PARTROOT,SEGTYPE=S2,$
  FIELD=STANKEY   ,ALIAS=STANKEY.KEY   ,USAGE=A2    ,ACTUAL=A2    ,$
  FIELD=SKIP2     ,ALIAS=              ,USAGE=A83   ,ACTUAL=A83   ,$
SEGNAME=STOKSTAT ,PARENT=PARTROOT,SEGTYPE=S2,$
  FIELD=STOCKEY   ,ALIAS=STOCKEY.KEY   ,USAGE=A16   ,ACTUAL=A16   ,$
  FIELD=SKIP3     ,ALIAS=              ,USAGE=A124  ,ACTUAL=A124  ,$
SEGNAME=CYCCOUNT ,PARENT=STOKSTAT,SEGTYPE=S2,$
  FIELD=CYCCKEY   ,ALIAS=CYCCKEY.KEY   ,USAGE=A2    ,ACTUAL=A2    ,$
  FIELD=SKIP4     ,ALIAS=              ,USAGE=A23   ,ACTUAL=A23   ,$
SEGNAME=BACKORDR ,PARENT=STOKSTAT,SEGTYPE=S2,$
  FIELD=BACKEY    ,ALIAS=BACKEY.KEY    ,USAGE=A10   ,ACTUAL=A10   ,$
  FIELD=SKIP5     ,ALIAS=              ,USAGE=A65   ,ACTUAL=A65   ,$
```

Syntax: How to Issue a GROUP Field in IMS

IMS fields can consist of multiple elementary fields. In the Master File, you can break the IMS field into component parts using a GROUP field.

The GROUP record in the Master File describes the combined elementary fields. FIELD records immediately following the GROUP record describe the individual elementary fields.

Each element of the group can have a different format. However, retrieval is more efficient if each element's USAGE format is of the same type (for example, alphanumeric or packed) as its ACTUAL format; the lengths may differ.

```
GROUP=gname, ALIAS=alias, USAGE=An, ACTUAL=Am,$
 FIELD=fld1,,usage1,actual1,$
  .
  .
  .
 FIELD=fldn,,usagen,actualn,$
```

where:

gname

Is the group name. It can be any name that complies with field naming conventions.

alias

Is the IMS field name from the DBD if the group field is a sequence or search field. Possible values are:

imsfield.KEY is the alias for a field that is an IMS key field. Form the alias by appending the suffix KEY to the name of the IMS field.

imsfield.IMS is the alias for a field that is an IMS search field. Form the alias by appending the suffix IMS to the name of the IMS field.

imsfield.HKY is the alias for a field that is the key of the root segment in an HDAM data source. Form the alias by appending the suffix HKY to the name of the IMS field.

Note: The keyword ALIAS is required.

An

Is the USAGE format for the GROUP. It must be alphanumeric and its length must count all F fields as length 4, all D fields as length 8, and all integer fields as length 4, regardless of the length specified in the ACTUAL format. For alphanumeric fields, USAGE length must equal ACTUAL length. For P (packed) fields, the length depends on the USAGE of the packed field:

USAGE of Packed Field	Length for GROUP USAGE
Pn or Pn.d, n≤15	8

USAGE of Packed Field	Length for GROUP USAGE
P16	8
P16.d	16
Pn or Pn.d, 16<n≤31	16

Am

Is the ACTUAL format for the GROUP. Its length is the sum of the lengths of the IMS fields.

fld1,...,fldn

Are field names for the individual elements that compose the group.

usage1,...,usagen

Are USAGE formats for the individual elements. For efficient retrieval, each individual USAGE format must be of the same data type as its corresponding ACTUAL format, but their lengths can differ.

actual1,...,actualn

Are ACTUAL formats for the individual elements. For efficient retrieval, each individual USAGE format must be of the same data type as its corresponding ACTUAL format, but their lengths can differ.

Example: Defining a Group Key

The following is an example of a group key definition in the Master File:

```
GROUP=G1, ALIAS=FILEKEY.KEY, USAGE=A16, ACTUAL=A11 ,$
  FIELD=F1,,A4,A4,$
  FIELD=F2,,P6,P3,$
  FIELD=F3,,I9,I4,$
```

The GROUP in the example describes an IMS key named FILEKEY that is 11 bytes long and consists of a 4-byte alphanumeric field, a 3-byte packed number, and a 4-byte integer.

The ACTUAL length of the GROUP is 11 (4+3+4).

The USAGE length of the GROUP is 16 (4+8+4) because it counts the packed field as 8.

Since the group components are of mixed data types, you must use individual fields in the WHERE expression of the server query.

```
WHERE F1 = ABCD
WHERE F1 BETWEEN A AND B
WHERE F2 = 245
```

If you reference either the group or the first elementary field from the group in your request, the adapter generates qualified SSAs in its DL/I calls for retrieval. Thus, IMS does the screening and returns the segments that pass the screening test back to the server.

However, if you reference an elementary field that is not the first field in the group, the server constructs DL/I calls to retrieve the segments sequentially, and then the server applies the screening test to the returned data.

Note: The adapter does not support a group field within a group field. You may be able to use DEFINE fields in the Master File instead.

Segment Redefinition in IMS: The RECTYPE Attribute

How to:

Establish Multiple RECTYPE Values in IMS

Example:

Illustrating an IMS DBD With a Redefined Segment

An IMS segment can have multiple definitions. For instance, a segment may contain either shipment or order information, depending on the value of one of its fields. If the field that identifies the type of segment is at the same position and has the same format and length in each redefinition, you can use the RECTYPE attribute to define the different segment types in the Master File.

The record type (RECTYPE) field can be part of the key or of the body of the segment. When you issue a request, you do not have to know which segment definition is called for. The server retrieves the appropriate fields and values based on the value in the RECTYPE field.

In the Master File, you describe the one IMS segment with multiple server segments: a base segment describing the unchanging portion, and a child segment describing each redefinition:

- First define the constant portion of the segment as the base segment; give it the same name as the IMS segment. Include a filler field for the portion that will be redefined. The field that identifies the different types must be in the redefined portion.

- Describe each redefined portion as a child of the base segment. You can give these children any valid segment names, but do not assign them SEGTYPE values. Include a filler field in each child to occupy the positions of fields actually defined in the base segment.

  In each child segment, describe the field that identifies the segment type with FIELDNAME=RECTYPE. The value in the RECTYPE field is the value that identifies the type of segment. For example, the RECTYPE field could contain the value S for a shipment record, or O for an order record. The format of the RECTYPE field can be alphanumeric, integer, or packed.

  Assign the identifying value (for example, S or O) as the ALIAS for the RECTYPE field. If more than one value can identify the same record type, use the ACCEPT attribute instead.

  Describe the remaining fields of each child segment based on its contents and function.

Example: Illustrating an IMS DBD With a Redefined Segment

The following example illustrates an IMS DBD with a redefined segment. The CLIENT segment contains client ID, address, and other client information. The INFO segment contains either shipment information or order information, depending on the value in the INFOTYPE field. If INFOTYPE contains the value S, the segment is a shipment segment; if INFOTYPE contains the value O, the segment is an order segment.

The relevant portions of the DBD are:

```
SEGM NAME=CLIENT,BYTES=(200),PTR=(TWIN),PARENT=0
  FIELD=(CLID,SEQ,U),BYTES=8,START=1,TYPE=C
   .
   .
   .
SEGM NAME=INFO,BYTES=(200),PTR=(TWIN),PARENT=CLIENT
  FIELD=(IKEY,SEQ,U),BYTES=8,START=1,TYPE=C
  FIELD=(INFOTYPE),BYTES=1,START=09,TYPE=C
   .
   .
   .
```

The corresponding Master File represents the IMS INFO segment with three segments:

```
    FILE=IMS1,SUFFIX=IMS
     SEGNAME=CLIENT,SEGTYPE=S2
       FIELD=F1,CLID.KEY,A8,A8,$
1.   SEGNAME=INFO    ,SEGTYPE=S2,PARENT=CLIENT,$
       FIELD=F3,IKEY.KEY,A8,A8,$
       FIELD=,,          A20,A20,$
2.   SEGNAME=SHIP,SEGTYPE=,PARENT=INFO,$
       FIELD=,,          A8,A8,$
       FIELD=RECTYPE,S,  A1,A1,$
       FIELD=SHIPDATE,,  A6,A6,$
        .
        .
        .
       other shipment info
3.   SEGNAME=ORDER,SEGTYPE=,PARENT=INFO,$
       FIELD=,,         A8,A8,$
       FIELD=RECTYPE,O, A1,A1,$
       FIELD=ORDERDATE,,A6,A6,$
        .
        .
        .
       other order info.
```

Note:

1. The base segment is named INFO, like the IMS segment. It contains the key field; the redefined portion is described as a filler field.

2. The SHIP segment describes the shipment record type; the field that corresponds to the IMS INFOTYPE field has FIELDNAME=RECTYPE and ALIAS=S.

3. The ORDER segment describes the order record type; the field that corresponds to the IMS INFOTYPE field has FIELDNAME=RECTYPE and ALIAS=O.

Notice that each child segment has a filler for the key field defined in the base segment.

Syntax: How to Establish Multiple RECTYPE Values in IMS

If multiple values identify the same record type (for example, S or T for a shipment record), use the ACCEPT attribute to enumerate the list or range of acceptable values. In this case, define the ALIAS value as blank.

```
FIELDNAME=RECTYPE, ALIAS=, USAGE=usage, ACTUAL=actual,
   ACCEPT=val1 [OR] val2 [[OR] ...valn],$
```

or

```
FIELDNAME=RECTYPE, ALIAS=, USAGE=usage, ACTUAL=actual,
   ACCEPT=val1 to val2
```

where:

`val1,val2,valn`

Defines a list or range of values that identifies the record type. A list can be continued on more than one line. Enclose values that contain embedded blanks or special characters within single quotation marks.

The following examples illustrate the ACCEPT attribute:

```
ACCEPT=AAA TO RRR
ACCEPT=136 TO 1029
ACCEPT=RED OR WHITE OR BLUE
ACCEPT=RED WHITE BLUE
ACCEPT=6 OR 11 OR 922 OR 1000
ACCEPT=RED WHITE 'GREEN GREY'
```

Repeating Data in IMS Fields: The OCCURS Segment

In an IMS data source, segments can have repeating fields or repeating groups of fields. The number of repetitions:

- Can be a fixed number.
- Can depend on the value of a field from the parent segment or from the non-repeating portion of the variable segment.
- May have to be calculated from the segment length.

In the Master File, you define multiple segments to describe one IMS variable length segment:

- Define the fixed (single-occurrence) portion of the IMS segment as the base segment; give it the same name as the IMS segment.
- Define each repeating field or group of fields from the IMS segment as a child segment whose parent is the base segment. The child segment definition has no SEGTYPE, but it includes the OCCURS attribute to specify how many times the field repeats.
- Define the ORDER field in the OCCURS segment if you need to associate a sequence number with each occurrence.

The OCCURS segment is a virtual segment (it does not physically exist) that describes the repetitions to the server. Permissible values for the OCCURS attribute are as follows:

OCCURS=	**Description**
n	The number of times the field repeats in the segment.
fieldname	The name of a field that contains a value indicating the number of times the field repeats in the segment. The repeating field must be at the end of the segment.
VARIABLE	Indicates that the number of repetitions must be computed from the length of the segment. In this case, the segment must contain a counter field as its first field; the counter field alias in the Master File must be IMS*name*.CNT. The repeating field must be at the end of the segment.

With a fixed number of occurrences, it is possible for the repeating field to be located between other fields in the segment rather than at the end of the segment. In this case, you must define a place-holder field at its position in the base segment. Then, in the OCCURS segment, identify the location of the repeating field by specifying the name of the place-holder field as the POSITION attribute.

Syntax: How to Describe Repetitions to the Server with an OCCURS Segment

```
SEGNAME=occseg, PARENT=imsseg, OCCURS=nfield                     ,$
SEGNAME=occseg, PARENT=imsseg, OCCURS=n [,POSITION=posfield]     ,$
SEGNAME=occseg, PARENT=imsseg, OCCURS=VARIABLE                   ,$
```

where:

occseg

Is the name of the OCCURS segment. It can be any valid segment name.

imsseg

Is the name of the base segment. It must be the IMS segment name.

nfield

Is the name of a field in the parent or non-repeating portion of the segment whose value is the number of times that the group repeats. You must define this field in the Master File whether or not it is a search field defined in the DBD.

n

Is the fixed number of times that the group repeats in the segment. It is an integer value from 1 to 4095.

posfield

Signals that the repeating field is embedded within the base segment rather than occurring at the end, and names a field in the base segment that marks the starting position of the repeating field.

VARIABLE

Indicates that the length of the repeating segment varies and that the number of occurrences can be computed from each segment. In this case, the (base) segment must contain a counter field as its first field; the counter field's alias value must be

`IMSname.CNT`

where:

IMSname

Is the name of the field in the IMS DBD.

In the IMS DBD, a variable length segment differs from a fixed length segment only in the BYTES parameter. For variable length segments, the BYTES parameter consists of two values: the maximum and minimum number of bytes. IMS cannot search for values among the repetitions within a segment. Therefore, in any request that references a field in a repeating group, the server searches and screens the OCCURS segments, not IMS.

Syntax: How to Describe the ORDER Field in IMS

Sometimes the sequence of fields within an OCCURS segment is significant. For example, each instance of the repeating field may represent one quarter of the year, but the segment may not have a field that specifies the quarter to which it applies.

ORDER is an optional counter used to identify the sequence number within a group of repeating fields. Specify it when the order of data is important. The ORDER field does not represent an existing field in the data source; it is used only for internal processing.

The ORDER field must be the last field described in the OCCURS segment.

```
FIELDNAME=name, ALIAS=ORDER, USAGE=In, ACTUAL=I4 ,$
```

where:

name

Is any valid field name.

I*n*

Is an integer format.

Note:

- The ALIAS value must be ORDER.
- The ACTUAL format must be I4.

The ORDER field must be the last field defined in the OCCURS segment.

In requests, you can use the value of the ORDER field. You can also specify a DEFINE statement in the Master File to translate it to more meaningful values. For example:

```
DEFINE QTR/A3 = DECODE ORDER(1 '1ST' 2 '2ND' 3 '3RD' 4 '4TH');
```

A subsequent request could include

```
SELECT TOT.TAXES WHERE QTR=1
```

or:

```
SELECT QTR,BALANCE,INTEREST
```

Example: Issuing OCCURS=n

In the following example, the IMS segment, IMS1, includes a group (consisting of the two fields MONTH and AMOUNT) that repeats 12 times. The COBOL FD for the segment is:

```
01  IMS1
    05  ACCOUNT   PIC X(9)
    05  TYPE      PIC XXX
    05  PAYMENT OCCURS 12 TIMES
        10 MONTH PIC 99
        10 AMT    PIC S9(3)V(99)COMP-3
```

The Master File uses two segments to describe this IMS variable length segment:

```
1. SEGNAME=IMS1,PARENT=,SEGTYPE=S2
     FIELD=ACT_NUM,ALIAS=ACCOUNT.KEY,A9,A9,$
     FIELD=TYPE,ALIAS=,A3,A3,$
2. SEGNAME=OCC1,PARENT=IMS1,OCCURS=12
     FIELD=MM, ALIAS=,A2,A2,$
     FIELD=AMT,ALIAS=,P6.3,P3,$
```

1. Segment IMS1 is the base segment. It has the same name as the IMS segment and describes the two non-repeating fields: ACT_NUM and TYPE.
2. The OCCURS segment, OCC1, identifies IMS1 as its parent. It has no SEGTYPE, and it includes the OCCURS attribute. The two repeating fields are described in this segment.

In the following example, the repeating group is not at the end of the segment; it is embedded in the segment before the LNAME field. The COBOL FD for this situation is:

```
01   IMS1
     05   ACCOUNT PIC X(9)
     05   TYPE    PIC XXX
     05   PAYMENT OCCURS 12 TIMES
          10 MONTH PIC 99
          10 AMT   PIC S9(3)V(99)COMP-3
     05   LNAME    PIC X(20)
```

The Master File must include LNAME in the base segment. It must also describe where the repeating fields fit into the base segment by defining a place-holder field before LNAME, equal to the length of the 12 occurrences, and by pointing to the place-holder field with the POSITION attribute:

```
     SEGNAME=IMS1,PARENT=,SEGTYPE=S2
      FIELD=ACT_NUM,ALIAS=ACCOUNT.KEY,A9,A9,$
      FIELD=TYPE,ALIAS=,A3,A3,$
1.    FIELD=HOLDIT,ALIAS=,A60,A60,$
      FIELD=LNAME,ALIAS=,A20,A20,$
2.   SEGNAME=OCC1,PARENT=IMS1,OCCURS=12,POSITION=HOLDIT
      FIELD=MM,ALIAS=,A2,A2,$
      FIELD=AMT,ALIAS=,P6.3,P3,$
```

1. The HOLDIT field is the place holder for the repeating group in the base segment (IMS1). Since the repeating group consists of 12 occurrences, each of which is 5 bytes long (A2 and P3, described in segment OCC1), the HOLDIT field is defined as A60.
2. The attribute POSITION=HOLDIT in the OCCURS segment declaration describes where the repeating group is located in the actual (base) segment.

Example: Issuing OCCURS=fieldname

In the next example, the number of occurrences is specified by the value in the TIMES field. The following COBOL FD describes this situation:

```
01   IMS1
     05   ACCOUNT PIC X(9)
     05   TYPE    PIC XXX
     05   TIMES   PIC S999   COMP-3
     05   PAYMENT OCCURS DEPENDING ON TIMES
          10 MONTH PIC 99
          10 AMT   PIC S9(3)V(99)COMP-3
```

The Master File attribute, OCCURS=TIMES, identifies the TIMES field as containing the number of repetitions. The Master File also defines the optional ORDER field as the last field in the OCCURS segment:

```
    SEGNAME=IMS1,SEGTYPE=S2
     FIELD=ACT_NUM,ALIAS=ACCOUNT.KEY,A9,A9,$
     FIELD=TYPE,ALIAS=,A3,A3,$
     FIELD=TIMES,ALIAS=,P4,P2,$
1.  SEGNAME=OCC1,PARENT=IMS1,OCCURS=TIMES
     FIELD=MM,ALIAS=,A2,A2,$
     FIELD=AMT,ALIAS=,P6.3,P3,$
2.   FIELD=WHICH,ALIAS=ORDER,I4,I4,$
```

1. The attribute OCCURS=TIMES identifies the value in the TIMES field as the number of instances of the repeating group.
2. The field named WHICH is the optional ORDER field (ALIAS=ORDER). It is an internal counter defined as the last field in the OCCURS segment. It associates a sequence number with each occurrence of the repeating group. With the ORDER field defined, a request can include a test that selects a specific occurrence. For example:

   ```
   WHERE WHICH = 3
   ```

Example: Issuing OCCURS=VARIABLE

This example describes a segment in which the number of occurrences must be calculated from the length of the segment. The first field in the segment must be a 2-byte counter field that contains the true length of the segment and is defined to IMS in the DBD. The COBOL FD for this variable length segment is:

```
01  IMS1
    05  COUNTER PIC 99 COMP
    05  ACCOUNT PIC X(9)
    05  TYPE    PIC XXX
    05  PAYMSCHED OCCURS 1 TO 12 TIMES
        10 MONTH PIC 99
        10 AMT   PIC 59(3)V(99)COMP-3
```

The Master File must specify OCCURS=VARIABLE and must describe the counter field with the attribute

```
ALIAS=IMSname.CNT
```

```
    SEGNAME=IMS1,SEGTYPE=S2
1.   FIELD=COUNTFLD,ALIAS=COUNTER.CNT,I2,I2,$
     FIELD=ACT_NUM,ALIAS=ACCOUNT.KEY,A9,A9,$
     FIELD=TYPE,ALIAS=,A3,A3,$
2.  SEGNAME=OCC1,PARENT=IMS1,OCCURS=VARIABLE
     FIELD=MM,ALIAS=,A2,A2,$
     FIELD=AMT,ALIAS=,P6.3,P3,$
```

1. The 2-byte counter field must be the first field in the base segment. Its alias is formed by appending the suffix CNT to the field name defined in the IMS DBD:

   ```
   ALIAS=COUNTER.CNT
   ```

2. In the OCCURS segment definition, the attribute OCCURS=VARIABLE indicates that the first field defined in the Master File contains the segment length and that the number of repetitions must be calculated from this length.

Syntax: How to Redefine Fields in IMS Databases

Support is provided for redefining record fields in IMS databases. This allows a field to be described with an alternate layout.

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

```
SEGNAME=segname,SEGTYPE=U,PARENT=parent,
OCCURS=1,POSITION=position ,$
```

where:

segname

Is the segment name.

parent

Is the name of the parent segment.

position

Is the field name of the field being redefined.

A one-to-one relationship is established between the parent record and the redefined segment. The following example illustrates redefinition of the IMS structure described in the COBOL file description:

```
01 ALLFIELDS.
  02 FLD1    PIC X(4).
  02 FLD2    PIC X(4).
  02 RFLD2   PIC 9(5)V99 COMP-3 REDEFINES FLD2.
  02 FLD3    PIC X(8).
```

```
FILENAME=REDEF, SUFFIX=IMS,$
  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=RFLD2,     ,USAGE=P8.2 ,ACTUAL=Z4     ,$
```

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

Use of the unique segment with redefined fields helps avoid problems with multipath reporting.

Note:

- 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 integer and packed fields, you must know your data. Attempts to print numeric fields that contain alpha data will produce data exceptions or errors converting values.
- More than one field can be redefined in a segment.

Syntax: How to Select a PSB

You have two options for selecting a PSB within the session:

- You can issue the IMS SET PSB command in your server profile, or in a stored procedure.
- You can use an Access File to identify the PSB. Each Access File corresponds to one Master File; it identifies the PSB associated with its corresponding Master File.

If you use an Access File, the adapter identifies the appropriate PSB when you reference a Master File in a request. The selection is automatic and transparent to you. Each Access File will contain six or more attributes.

```
PSB=psbname,PCBNUMBER= pcb_number,PL1=(NO | YES),WRITE=(YES | NO),$

SEGNAME=segment_name, KEYTYPE=segtype (S0|(S or S1)|S2|(SH or SH1)|SH2),$

ALTPCBNAME=index name  , ALTPCBNUMBER=index PCB number,$
```

where:

psbname

Is the actual name of the Access File member to use. The Access File will contain the name of the actual PSB that IMS will access. If the member does not exist, the following message is (generated) displayed:

```
(EDA4261) PSB MEMBER NOT FOUND: psbname
```

pcb_number

The position within the PSB that points to the database to be processed.

PL1

Indicates whether the PSB was generated with LANG=(PL/I, COBOL or Assembler). YES indicates PL/I. NO indicates COBOL or Assembler.

WRITE

YES - allows SQL update for this database.

NO - does not allow SQL update for this database and is the default.

segname_name

Is / Are the segment name(s) defined in the database definition.

segtype

Identifies the characteristics of the segment's key field. See *SEGTYPE in IMS* on page 13-24.

index name (Optional)

Identifies a secondary index database defined by the 'Procseq=' option within the PSB.

index PCB number (Optional)

Identifies the PCB number of the indexed database.

The member name of an Access File must be the same as the member name of the corresponding Master File.

Note:

All participating files in a join must use the same PSB. Any attempt to join files that require different PSBs generates the following message:

```
(EDA4295) ACCESS POINTS TO DIFFERENT PSBS IN JOIN
```

Example: Sample Access File for DI21PART

The following is a sample Access File corresponding to the DI21PART Master File.

```
PSB=DI21PSB, WRITE=NO PCBNUMBER=2, PL1=NO,,$
SEGNAME=PARTROOT,  KEYTYPE=S2,$
SEGNAME=STANINFO,  KEYTYPE=S2,$
SEGNAME=STOKCOUNT, KEYTYPE=S2,$
SEGNAME=CYCCOUNT,  KEYTYPE=S2,$
SEGNAME=BAKORDR,   KEYTYPE=S2,$
```

Secondary Indexes in IMS

In this section:

If you want to access a segment through a field in a segment that is not its sequence field (primary key), IMS provides the option of creating a secondary index on the field. In some situations, using a secondary index improves performance. The secondary index is itself a separate data source. Each of its records contains a value of the field to be indexed and a pointer to the target segment of the data source record containing that value.

When using a secondary index, IMS locates a record by first reading the *index* data source to retrieve the appropriate pointer and then using the pointer to read the *data* data source.

If a PCB includes the parameter

```
PROCSEQ=indexDBDname
```

the named index is used as the main entry point into the data source.

One Master File and Access File can describe all primary and secondary indexes for a data source. Then, given a request, the adapter can examine all record selection tests to determine the best access path into the data source. The adapter can take advantage of this Auto Index Selection feature if:

- The PSB includes a PCB for each index you may need to access. Each such index PCB must contain a

  ```
  PROCSEQ=indexDBDname
  ```

  parameter that identifies the index DBD.

- The index targets the root segment in the Master File.

In the Master File, prior to the secondary index definitions, you must describe the entire segment of the data source. Every field listed in the DBD is an IMS sequence field or search field, and each secondary index is based on one or more of these fields. You must assign each secondary index field its appropriate alias, as described in *How to Specify IMS Field Attributes* on page 13-25.

Note: IMS allows for system-defined sub-sequence fields to make an index unique. The Master File can completely ignore the presence and length of such fields.

Using an IMS Secondary Index

Example:

Describing a Shared Secondary Index DBD

Describing an Inverted Tree Structure

The CREATE SYNONYM command places information about secondary indexes in the Access File. No auxiliary virtual fields (that is, SKY groups/fields) are created in the Master File.

Note: Existing .sky field suffix-based definitions are supported to ensure compatibility.

Two segments belonging to the same root path might be connected by secondary indexes:

- A target segment has an associated special virtual search field (described in the Access File) to be used in the qualified SSA.
- A source segment has the real base fields to be used in the screening predicates to find the target segment instances.

One target segment may have many secondary indexes based on search fields from different source segments. The secondary index can be based on up to five base fields from the same segment.

Each PCB can be associated with only one index: primary or secondary. A secondary index is defined in the PROCSEQ parameter. When the synonym is created, a primary PCB number is chosen and placed in the Access File PSB statement in the PCBNUMBER parameter. If the PSB contains several PCBs that define identical hierarchies, the primary PCB number is selected according to following rules:

- If all eligible PCBs have a PROCSEQ parameter, the primary PCB number is taken from the CREATE SYNONYM command.

  The root segment is determined, and hierarchical entries processed, in the sequence corresponding to the primary PCB secondary index when an area sweep is performed.

- If at least one of the eligible PCBs does not have a PROCSEQ parameter, the primary PCB number corresponds to the last encountered PCB without a PROCSEQ.

  The root is processed in the sequence corresponding to the physical sequence of segment instances for the non-keyed segment, and in the key sequence for key segments when an area sweep is performed.

Sequential numbers of all other PCBs of identical hierarchies associated with the same DBD are stored in the XDFLD Access File statement in the parameter ALTPCBNUMBER.

Each DBD may be associated with any number of PCBs. Each PCB may have a PROCSEQ parameter to indicate an index key. The segment to which this key refers becomes the tree root segment. Therefore, it is possible to have different hierarchical structures associated with the same database. All identical hierarchies (the only difference is in the PROCSEQ key name) are described by the same Master File. Different Master Files are generated for the various inverted hierarchies.

Access File XDFLD statements are used to describe secondary indexes. There is a one-to-one relationship between XDFLD statements in the Access File and PROCSEQ parameters in the PCB.

There are two types of XDFLD's statements:

- The first type describes secondary indexing based on PROCSEQ.

  The XDFLD belongs to the tree root segment and must have an ALTPCBNUMBER parameter that refers to the appropriate PCB number. Each PROCSEQ parameter can represent an identical hierarchical structure. Multiple PROCSEQ parameters generate multiple XDFLD statements in the Access File. Only one such XDFLD statement is needed to build a qualified SSA in the request.

- The second type describes secondary indexing based on the SENSEG INDICES parameter.

  A SENSEG INDICES parameter may be associated with any segment and must not have an ALTPCBNUMBER parameter in the Access File.

  The Access File may contain XDFLD statements produced from information that is found in the INDICES parameter of the SENSEG macro in a PCB definition. Up to 32 different XDFLD statements associated with a segment can be built from one SENSEG macro. Any number of such XDFLD statements might be used to build a qualified SSA in the request. Both types of secondary indexes can be used simultaneously in the qualified SSA in the request. For related information, see *Defining Subsets of DBD Segments and IMS Fields in Segments* on page 13-21.

Secondary indexes are physically stored in separate index databases. These databases are referenced by the PROCSEQ and INDICES parameters. One index database can be used to store up to 16 secondary indexes. Such a database, and the related indexes, is referred to as a shared secondary index.

Example: Describing a Shared Secondary Index DBD

You can use information from a shared secondary index DBD to create XDFLD statements. The following files illustrate the required entries.

DBD and PSB

```
      DBD NAME=DIX1DBD,ACCESS=(HIDAM,VSAM)
      DATASET   DD1=DIX1KAPL,DEVICE=3380
      SEGM      NAME=PARTROOT,PARENT=0,BYTES=50,PTR=T
      LCHILD    NAME=(SEG1,DIX1DBX0),PTR=INDX
      FIELD     NAME=(PARTKEY,SEQ,U),TYPE=C,BYTES=17,START=1
      FIELD     NAME=PARTFIL1,TYPE=C,BYTES=8,START=18
      FIELD     NAME=PARTNAME,TYPE=C,BYTES=23,START=27
      FIELD     NAME=/SX1
      LCHILD    NAME=(X1SEG,DIX1DBX1),PTR=INDX
      XDFLD     NAME=PARTNAMX,SRCH=PARTNAME,SUBSEQ=/SX1,NULLVAL=BLANK
*
      SEGM      NAME=STANINFO,PARENT=PARTROOT,BYTES=85
      FIELD     NAME=(STANKEY,SEQ),TYPE=C,BYTES=2,START=1
      FIELD     NAME=STANFIL1,TYPE=C,BYTES=16,START=3
      FIELD     NAME=STANTYPE,TYPE=C,BYTES=3,START=19
      FIELD     NAME=STANFIL2,TYPE=C,BYTES=26,START=22
      FIELD     NAME=STANASST,TYPE=C,BYTES=4,START=48
      FIELD     NAME=STANFIL3,TYPE=C,BYTES=2,START=52
      FIELD     NAME=STANVALU,TYPE=C,BYTES=2,START=54
*
      SEGM      NAME=STOKSTAT,PARENT=PARTROOT,BYTES=160
      FIELD     NAME=(STOCKEY,SEQ),TYPE=C,BYTES=16,START=1
      FIELD     NAME=STOKFIL1,TYPE=C,BYTES=4,START=17
      FIELD     NAME=STOKNUMS,TYPE=C,BYTES=9,START=21
```

```
         FIELD    NAME=STOKFIL2,TYPE=C,BYTES=20,START=30
         FIELD    NAME=STOKFREQ,TYPE=C,BYTES=6,START=50
         LCHILD   NAME=(X2SEG,DIX1DBX2),PTR=INDX
         XDFLD    NAME=CYCLNUMX,SRCH=CYCLNUMB,SEGMENT=CYCCOUNT,        X
                  SUBSEQ=/SX1,NULLVAL=BLANK
*
         SEGM     NAME=CYCCOUNT,PARENT=STOKSTAT,BYTES=25
         FIELD    NAME=(CYCLKEY,SEQ),TYPE=C,BYTES=2,START=1
         FIELD    NAME=CYCLNUMB,TYPE=C,BYTES=8,START=3
         FIELD    NAME=CYCLFIL1,TYPE=C,BYTES=4,START=11
         FIELD    NAME=CYCLVALU,TYPE=C,BYTES=7,START=15
         FIELD    NAME=/SX1
*
         SEGM     NAME=BACKORDR,PARENT=STOKSTAT,BYTES=80
         FIELD    NAME=(BACKKEY,SEQ),TYPE=C,BYTES=10,START=1
         FIELD    NAME=BACKREFR,TYPE=C,BYTES=4,START=77
         DBDGEN
         FINISH
         END

DBPCB01  PCB TYPE=DB,DBDNAME=DIX1DBD,PROCOPT=A,KEYLEN=49
   SENSEG NAME=PARTROOT,PARENT=0
   SENSEG NAME=STANINFO,PARENT=PARTROOT
   SENSEG NAME=STOKSTAT,PARENT=PARTROOT
   SENSEG NAME=CYCCOUNT,PARENT=STOKSTAT
   SENSEG NAME=BACKORDR,PARENT=STOKSTAT
DBPCB02  PCB TYPE=DB,DBDNAME=DIX1DBD,PROCOPT=A,KEYLEN=49,              X
               PROCSEQ=DIX1DBX1
   SENSEG NAME=PARTROOT,PARENT=0
   SENSEG NAME=STANINFO,PARENT=PARTROOT
   SENSEG NAME=STOKSTAT,PARENT=PARTROOT
   SENSEG NAME=CYCCOUNT,PARENT=STOKSTAT
   SENSEG NAME=BACKORDR,PARENT=STOKSTAT
DBPCB03  PCB TYPE=DB,DBDNAME=DIX1DBD,PROCOPT=A,KEYLEN=49,              X
               PROCSEQ=DIX1DBX2
   SENSEG NAME=STOKSTAT,PARENT=0
   SENSEG NAME=CYCCOUNT,PARENT=STOKSTAT
   SENSEG NAME=BACKORDR,PARENT=STOKSTAT
   SENSEG NAME=PARTROOT,PARENT=STOKSTAT
   SENSEG NAME=STANINFO,PARENT=PARTROOT
  PSBGEN LANG=ASSEM,PSBNAME=DIX1PSB,CMPAT=YES
 END
```

Master File:

```
FILENAME=DIX_DIX1DBD, SUFFIX=IMS     , $
  SEGMENT=PARTROOT, SEGTYPE=S0, $
    FIELDNAME=PARTKEY, ALIAS=PARTKEY.KEY, USAGE=A17, ACTUAL=A17, $
    FIELDNAME=PARTFIL1, ALIAS=PARTFIL1.IMS, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=FILL1, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=PARTNAME, ALIAS=PARTNAME.IMS, USAGE=A23, ACTUAL=A23, $
    FIELDNAME=FILL2, USAGE=A1, ACTUAL=A1, $

  SEGMENT=STOKSTAT, SEGTYPE=S0, PARENT=PARTROOT, $
    FIELDNAME=STOCKEY, ALIAS=STOCKEY.KEY, USAGE=A16, ACTUAL=A16, $
    FIELDNAME=STOKFIL1, ALIAS=STOKFIL1.IMS, USAGE=A4, ACTUAL=A4, $
    FIELDNAME=STOKNUMS, ALIAS=STOKNUMS.IMS, USAGE=A9, ACTUAL=A9, $
    FIELDNAME=STOKFIL2, ALIAS=STOKFIL2.IMS, USAGE=A20, ACTUAL=A20,$
    FIELDNAME=STOKFREQ, ALIAS=STOKFREQ.IMS, USAGE=A6, ACTUAL=A6, $
    FIELDNAME=FILL1, USAGE=A105, ACTUAL=A105, $

  SEGMENT=CYCCOUNT, SEGTYPE=S0, PARENT=STOKSTAT, $
    FIELDNAME=CYCLKEY, ALIAS=CYCLKEY.KEY, USAGE=A2, ACTUAL=A2, $
    GROUP=CYCLNUMB, ALIAS=CYCLNUMB.IMS, USAGE=A8, ACTUAL=A8, $
     FIELDNAME=CNMB1, USAGE=A3, ACTUAL=A3, $
     FIELDNAME=CNMB2, USAGE=A3, ACTUAL=A3, $
     FIELDNAME=CNMB3, USAGE=A2, ACTUAL=A2, $
    FIELDNAME=CYCLFIL1, ALIAS=CYCLFIL1.IMS, USAGE=A4, ACTUAL=A4, $
    FIELDNAME=CYCLVALU, ALIAS=CYCLVALU.IMS, USAGE=A7, ACTUAL=A7, $
    FIELDNAME=FILL1, USAGE=A4, ACTUAL=A4, $
```

Access File:

```
PSB=DIX1PSB, WRITE=NO, PCBNUMBER=2, PL1=NO, $
  SEGNAME=PARTROOT, KEYTYPE=S2, $
   XDFLD=PARTNAMX, SRCH=PARTNAME, ALTPCBNUMBER=3, $
   XDFLD=PARTFILX, SRCH=PARTFIL1, ALTPCBNUMBER=4, $
  SEGNAME=STOKSTAT, KEYTYPE=S2, $
   XDFLD=CYCLNUMX, SRCH=CNMB1/CNMB3/CNMB2, BASESEG=CYCCOUNT, $
  SEGNAME=CYCCOUNT, KEYTYPE=S2, $
```

Example: Describing an Inverted Tree Structure

This example describes an inverted tree structure that corresponds to PROSEQ DIX1DBX2 in *Describing a Shared Secondary Index DBD* on page 13-47.

Master File:

```
FILENAME=DIX_DIX1DBD, SUFFIX=IMS , $
 SEGMENT=STOKSTAT, SEGTYPE=S0, $
 FIELDNAME=STOCKEY, ALIAS=STOCKEY.KEY, USAGE=A16, ACTUAL=A16, $
 FIELDNAME=STOKFIL1, ALIAS=STOKFIL1.IMS, USAGE=A4, ACTUAL=A4, $
 FIELDNAME=STOKNUMS, ALIAS=STOKNUMS.IMS, USAGE=A9, ACTUAL=A9, $
 FIELDNAME=STOKFIL2, ALIAS=STOKFIL2.IMS, USAGE=A20, ACTUAL=A20, $
 FIELDNAME=STOKFREQ, ALIAS=STOKFREQ.IMS, USAGE=A6, ACTUAL=A6, $
 FIELDNAME=FILL1, USAGE=A105, ACTUAL=A105, $

 SEGMENT=CYCCOUNT, SEGTYPE=S0, PARENT=STOKSTAT, $
 FIELDNAME=CYCLKEY, ALIAS=CYCLKEY.KEY, USAGE=A2, ACTUAL=A2, $
 GROUP=CYCLNUMB, ALIAS=CYCLNUMB.IMS, USAGE=A8, ACTUAL=A8, $
 FIELDNAME=CNMB1, USAGE=A3, ACTUAL=A3, $
 FIELDNAME=CNMB2, USAGE=A3, ACTUAL=A3, $
 FIELDNAME=CNMB3, USAGE=A2, ACTUAL=A2, $
 FIELDNAME=CYCLFIL1, ALIAS=CYCLFIL1.IMS, USAGE=A4, ACTUAL=A4, $
 FIELDNAME=CYCLVALU, ALIAS=CYCLVALU.IMS, USAGE=A7, ACTUAL=A7, $
 FIELDNAME=FILL1, USAGE=A4, ACTUAL=A4, $
 SEGMENT=PARTROOT, SEGTYPE=U, PARENT=STOKSTAT, $
 FIELDNAME=PARTKEY, ALIAS=PARTKEY.KEY, USAGE=A17, ACTUAL=A17, $
 FIELDNAME=PARTFIL1, ALIAS=PARTFIL1.IMS, USAGE=A8, ACTUAL=A8, $
 FIELDNAME=FILL1, USAGE=A1, ACTUAL=A1, $
 FIELDNAME=PARTNAME, ALIAS=PARTNAME.IMS, USAGE=A23, ACTUAL=A23, $
 FIELDNAME=FILL2, USAGE=A1, ACTUAL=A1, $
```

Access File:

```
PSB=DIX1PSB, WRITE=NO, PCBNUMBER=5, PL1=NO, $
 SEGNAME=STOKSTAT, KEYTYPE=S2, $
 XDFLD=CYCLNUMX, SRCH=CNMB1/CNMB3/CNMB2, BASESEG=CYCCOUNT,ALTPCBNUMBER=5,
$
 XDFLD=CYCLVALX, SRCH=CYCLVALU, BASESEG=CYCCOUNT, ALTPCBNUMBER=6, $
 SEGNAME=CYCCOUNT, KEYTYPE=S2, $
 SEGNAME=PARTROOT, KEYTYPE=S2, $
 XDFLD=PARTNAMX, SRCH=PARTNAME, $
 XDFLD=PARTFILX, SRCH=PARTFIL1, $
```

This example uses four secondary indexes:

- **PARTNAMX** is the secondary index used to find the segment PARTROOT by the value of the PARTNAME field. This is a SENSEG-type secondary index. Target and source segments are in the same PARTROOT.
- **PARTFILX** is the secondary index used to find the segment PARTROOT by the value of the PARTFIL1 field. This is a SENSEG-type secondary index. Target and source segments are in the same PARTROOT.
- **CYCLNUMX** is the secondary index used to find the segment STOKSTAT by the value of the CNMB1,CNMB3 and CNMB2 fields. This is a PROCSEQ-type secondary index. The target segment is STOKSTAT; the source segment is CYCCOUNT.
- **CYCLVALX** is the secondary index used to find the segment STOKSTAT by the value of the CYCLVALU field.This is PROCSEQ-type secondary index. The target segment is STOKSTAT; the source segment is CYCCOUNT.

Definition description:

1. Secondary index description associated with the target segment.
2. XDFLD parameter defines the secondary index name used to build the SSA.
3. BASESEG parameter defines the source segment name (if it is not the target segment).
4. SRCH parameter defines the names of one or more (up to 5) base fields, and their order in the secondary index.

Note: A qualified SSA is built for the target segment if at least one eligible screening condition is provided for the associated primary key, search field, or secondary key base source field.

Access File Attributes

In this section:

Describing a Partition to the Server

Describing a Concatenated PCB

An Access File describes the physical attributes of the IMS file, such the name of the PCB, the type of PCB, and any secondary indexes.

Access Files store database access information used to translate report requests into the required IMS direct calls.

An Access File consists of 80-character records in comma-delimited format. A record in an Access File contains a list of attributes and values, separated by commas and terminated with a comma and dollar sign (,$). This list is free-form and spans several lines if necessary. You can specify attributes in any order.

The server reads blank lines, and lines starting with a dollar sign in column one, as comments.

Describing a Partition to the Server

How to:

Describe a Partition in the Access File

IMS limits the size of its data sources. A site that needs a larger data source may be able to create several smaller data sources by partitioning the large data source based on root key values.

Using extended Access File attributes, you can describe a partition to the server, describe how to concatenate the parts, and assign a name to the concatenated PCB. When you issue a request, you can report from the concatenated PCB name or from any of the individual partitions, depending on the file name you reference in the request.

Syntax: How to Describe a Partition in the Access File

A partition assigns each record to a specific data source depending on its root key value. The first partition contains records with the lowest key values; the next partition contains records with higher key values, and so on; the last partition contains records with the highest key values. Each partition is a separate IMS data source and, therefore, has a separate PCB in the PSB.

To describe the key range in each partition to the server, add the LOWVALUE and HIGHVALUE attributes to the appropriate PCB records in the Access File.

```
PCBNAME=mfdname,PCBTYPE=DB,LOWVALUE={value1|0},HIGHVALUE={value2|FF},$
```

where:

mfdname

Is the name of the Master File for one partition of the large data source. (Since the partition is a separate data source with its own DBD, it needs its own PCB and Master File.)

value1

Is the lowest key value, in alphanumeric format, in the partition accessed with Master File *mfdname*. 0 is the default value.

value2

Is the highest key value, in alphanumeric format, in the partition accessed with Master File *mfdname*. Hexadecimal FF is the default value.

Note:

- The LOWVALUE and HIGHVALUE attributes are ignored if you do not define a corresponding concatenation of the PCBs.
- The number of partitions you can define for a data source is limited by the number of PCBs in the PSB.
- The partitioning field must be the key of the root segment.
- The partitioning field must have an alphanumeric format.

If the partitioning field is a group composed of multiple subfields, each subfield value must be alphanumeric, and you must specify the concatenated subfield values in the LOWVALUE and HIGHVALUE attributes. For example, if the root key low value is composed of F1=AAAA, F2=88, and F3=BBB, then LOWVALUE=AAAA88BBB.

Describing a Concatenated PCB

In an Access File, you can concatenate individual PCBs by assigning a name to the concatenation and issuing a request against it.

The PCBs that you concatenate do not have to be partitioned; that is, their Access File records do not have to include the LOWVALUE and HIGHVALUE attributes. However, if they do include the partitioning information, the adapter can examine the request and determine which PCBs satisfy the request. Without the partitioning information, the adapter must access every PCB that participates in the concatenation.

Syntax: How to Describe a Concatenated PCB in the Access File

The format of the Access File for the Adapter for IMS to provide extended support for concatenated databases is as follows:

1. The CONCATPCB statements provide the PCB number and the low and high value range for the root segment key.

   For example:

   ```
   CONCATPCB=4,LOWVALUE=0,HIGHVALUE=0, $
   CONCATPCB=6,LOWVALUE=1,HIGHVALUE=1, $
   .................................................................
   CONCATPCB=12,LOWVALUE=9,HIGHVALUE=9, $
   ```

2. The CONCPOSITION statement provides root segment key substring information.The substring value is compared with the key ranges defined in the CONCATPCB statements. Positions in the key are counted from 0 for the substring operation.

The key value is converted to alphanumeric format prior to the substring operation.The key format is taken from the ACTUAL description. A Packed or Zoned format number is processed as an unsigned one.

Conversion rules for the Packed format:

- Result buffer size for Pn is n*2-1 bytes.
- Converted number is adjusted to the right in the buffer with the leading 0.

Conversion rules for the Integer format:

- Only I1, I2, and I4 formats are supported.
- Result buffer size for I1 is 3 bytes; I2 is 5 bytes; I4 is 10 bytes.

For example:

```
CONCPOSITION=1,LENGTH=1, $
```

Example: Describing a HDAM Concatenated PCB

```
FILENAME=HDAM0, SUFFIX=IMS     , $
  SEGMENT=LANGUAGE, SEGTYPE=S0, $
    FIELDNAME=EMPL6, ALIAS=EMPL6.HKY, USAGE=P8,  ACTUAL=P4, $
    FIELDNAME=LANG6,  ALIAS=LANG6.IMS, USAGE=A15, ACTUAL=A15, $
```

```
PSB=PIHDPSB,PCBNUMBER= 4,PL1=NO,WRITE=NO,$
SEGNAME=LANGUAGE, KEYTYPE=S1  ,$
CONCATPCB=4,LOWVALUE=1,HIGHVALUE=1, $
CONCATPCB=6,LOWVALUE=2,HIGHVALUE=2, $
CONCATPCB=8,LOWVALUE=3,HIGHVALUE=3, $
CONCPOSITION=4,LENGTH=1, $
```

Note: PCBNUMBER in the PSB statement refers to the PCB associated with the root segment.

In the following request, we associate the 5th digit in the key (offset 4 in the key buffer) with the PCB to schedule.:

```
TABLE FILE HDAM0
PRINT *
WHERE EMPL6 EQ 9336
END
```

PCB number 8 will be used for the numeric qualifier 9336. The extracted value is 3 (since packed number 9336 is unpacked to the 7 bytes buffer and the result value is 0009336).

Example: Describing a HIDAM Concatenated PCB

EMPDB01, EMPDB02, and EMPDB03 are HIDAM data sources that illustrate partitioning and concatenation of PCBs in the Access File. The following topics illustrate:

- The EMPDB01, EMPDB02, and EMPDB03 DBDs for the three partitions of the data source, and, since these are HIDAM data sources, the EMPDBIX1, EMPDBIX2, and EMPDBIX3 DBDs for the index data sources.
- A PSB with three PCBs, each corresponding to one partition of the data source.
- The EMPDB01, EMPDB02, and EMPDB03 Master Files for the individual partitions, and the EMPDBJ Master File for the concatenation.
- The Access File corresponding to the PSB. It includes the LOWVALUE and HIGHVALUE attributes for each PCB and the CONCATNAME record to define the concatenation.

EMPDB01 DBD

Data source DBD:

```
PRINT NOGEN
DBD       NAME=EMPDB01,ACCESS=(HIDAM,VSAM)
DATASET   DD1=EMPDB01,DEVICE=3380,BLOCK=4096,SCAN=5
  *
SEGM NAME=EMPINFO,PTR=H,PARENT=0,BYTES=212
  FIELD NAME=(SSNALPHA,SEQ,U),BYTES=9,START=18,TYPE=C
  LCHILD NAME=(SEGIX1,EMPDBIX1),PTR=INDX
  FIELD NAME=SEQFIELD,BYTES=2,START=1,TYPE=P
  FIELD NAME=REVSEQ,BYTES=6,START=3,TYPE=C
  FIELD NAME=SSN,BYTES=3,START=15,TYPE=P
  FIELD NAME=EMPID,BYTES=12,START=27,TYPE=C
  FIELD NAME=LNAME,BYTES=12,START=39,TYPE=C
  FIELD NAME=FNAME,BYTES=12,START=51,TYPE=C
  FIELD NAME=ADMDATE,BYTES=8,START=72,TYPE=C
  *
DBDGEN
FINISH
END
```

Index DBD:

```
PRINT NOGEN
DBD       NAME=EMPDBIX1,ACCESS=INDEX
DATASET   DD1=EMPDBIX1,DEVICE=3380
  *
SEGM NAME=SEGIX1,PARENT=0,BYTES=9
  FIELD NAME=(SSNIX,SEQ,U),BYTES=9,START=1
  LCHILD NAME=(EMPINFO,EMPDB01),INDEX=SSNALPHA
DBDGEN
FINISH
END
```

EMPDB02 DBD

Database DBD:

```
PRINT NOGEN
DBD      NAME=EMPDB02,ACCESS=(HIDAM,VSAM)
DATASET  DD1=EMPDB02,DEVICE=3380,BLOCK=4096,SCAN=5
  *
SEGM NAME=EMPINFO,PTR=H,PARENT=0,BYTES=212
  FIELD NAME=(SSNALPHA,SEQ,U),BYTES=9,START=18,TYPE=C
  LCHILD NAME=(SEGIX2,EMPDBIX2),PTR=INDX
  FIELD NAME=SEQFIELD,BYTES=2,START=1,TYPE=P
  FIELD NAME=REVSEQ,BYTES=6,START=3,TYPE=C
  FIELD NAME=SSN,BYTES=3,START=15,TYPE=P
  FIELD NAME=EMPID,BYTES=12,START=27,TYPE=C
  FIELD NAME=LNAME,BYTES=12,START=39,TYPE=C
  FIELD NAME=FNAME,BYTES=12,START=51,TYPE=C
  FIELD NAME=ADMDATE,BYTES=8,START=72,TYPE=C
  *
DBDGEN
FINISH
END
```

Index DBD:

```
PRINT NOGEN
DBD      NAME=EMPDBIX2,ACCESS=INDEX
DATASET  DD1=EMPDBIX2,DEVICE=3380
  *
SEGM NAME=SEGIX2,PARENT=0,BYTES=9
  FIELD NAME=(SSNIX,SEQ,U),BYTES=9,START=1
  LCHILD NAME=(EMPINFO,EMPDB02),INDEX=SSNALPHA
DBDGEN
FINISH
END
```

EMPDB03 DBD

Database DBD:

```
PRINT NOGEN
DBD       NAME=EMPDB03,ACCESS=(HIDAM,VSAM)
DATASET   DD1=EMPDB03,DEVICE=3380,BLOCK=4096,SCAN=5
  *
SEGM NAME=EMPINFO,PTR=H,PARENT=0,BYTES=212
  FIELD NAME=(SSNALPHA,SEQ,U),BYTES=9,START=18,TYPE=C
  LCHILD NAME=(SEGIX3,EMPDBIX3),PTR=INDX
  FIELD NAME=SEQFIELD,BYTES=2,START=1,TYPE=P
  FIELD NAME=REVSEQ,BYTES=6,START=3,TYPE=C
  FIELD NAME=SSN,BYTES=3,START=15,TYPE=P
  FIELD NAME=EMPID,BYTES=12,START=27,TYPE=C
  FIELD NAME=LNAME,BYTES=12,START=39,TYPE=C
  FIELD NAME=FNAME,BYTES=12,START=51,TYPE=C
  FIELD NAME=ADMDATE,BYTES=8,START=72,TYPE=C
  *
DBDGEN
FINISH
END
```

Index DBD:

```
PRINT NOGEN
DBD       NAME=EMPDBIX3,ACCESS=INDEX
DATASET   DD1=EMPDBIX3,DEVICE=3380
  *
SEGM NAME=SEGIX3,PARENT=0,BYTES=9
  FIELD NAME=(SSNIX,SEQ,U),BYTES=9,START=1
  LCHILD NAME=(EMPINFO,EMPDB03),INDEX=SSNALPHA
DBDGEN
FINISH
END
```

PSB to Access the EMPDB Data Sources

```
PCB        TYPE=TP,MODIFY=YES,EXPRESS=YES
PCB        TYPE=TP,EXPRESS=NO,MODIFY=YES,SAMETRM=YES
PCB        TYPE=DB,DBDNAME=EMPDB01,PROCOPT=GO,KEYLEN=9
SENSEG     NAME=EMPINFO,PARENT=0
PCB        TYPE=DB,DBDNAME=EMPDB02,PROCOPT=GO,KEYLEN=9
SENSEG     NAME=EMPINFO,PARENT=0
PCB        TYPE=DB,DBDNAME=EMPDB03,PROCOPT=GO,KEYLEN=9
SENSEG     NAME=EMPINFO,PARENT=0
PSBGEN LANG=COBOL,PSBNAME=EMPPSBJ,CMPAT=YES
END
```

Master Files to Access the EMPDB Data Sources

The EMPDB01, EMPDB02, and EMPDB03 Master Files each correspond to a PCB for one individual partition of the data source; the EMPDBJ Master File corresponds to the concatenation of the three partitions.

EMPDB01 Master File

```
FILE=EMPDB01,SUFFIX=IMS,$
SEGNAME=EMPINFO,PARENT=,SEGTYPE=S1,$
FIELD=SEQFIELD      ,ALIAS=SEQFIELD.IMS     ,USAGE=P6    ,ACTUAL=P2  ,$
FIELD=REVSEQ        ,ALIAS=REVSEQ.IMS       ,USAGE=A6    ,ACTUAL=A6  ,$
FIELD=SALARY        ,ALIAS=                 ,USAGE=P8    ,ACTUAL=P4  ,$
FIELD=OT_HR_PAY     ,ALIAS=                 ,USAGE=P5    ,ACTUAL=P2  ,$
FIELD=SSN           ,ALIAS=SSN.IMS          ,USAGE=P9    ,ACTUAL=P3  ,$
FIELD=SSNALPHA      ,ALIAS=SSNALPHA.KEY     ,USAGE=A9    ,ACTUAL=A9  ,$
FIELD=EMPID         ,ALIAS=EMPID.IMS        ,USAGE=A12   ,ACTUAL=A12,$
FIELD=ELAST_NAME    ,ALIAS=ELNAME.IMS       ,USAGE=A12   ,ACTUAL=A12,$
FIELD=EFIRST_NAME   ,ALIAS=EFNAME.IMS       ,USAGE=A12   ,ACTUAL=A12,$
FIELD=DATE_OF_BIRTH ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=RACE          ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=ADMIT_DATE    ,ALIAS=ADMDATE.IMS      ,USAGE=A08   ,ACTUAL=A08,$
FIELD=ADMIT_TYPE    ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=DISPOSITION   ,ALIAS=                 ,USAGE=A02   ,ACTUAL=A02,$
FIELD=TRANSFER_DATE ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=ALLERGY1      ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY2      ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY3      ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY4      ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=HOUSING       ,ALIAS=                 ,USAGE=A03   ,ACTUAL=A03,$
FIELD=RPR           ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=URIN          ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=PRACTITIONAR  ,ALIAS=                 ,USAGE=A02   ,ACTUAL=A02,$
FIELD=SHIFT         ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=PATINET_ID    ,ALIAS=                 ,USAGE=P12   ,ACTUAL=P4  ,$
FIELD=WHO_ADDED     ,ALIAS=                 ,USAGE=A10   ,ACTUAL=A10,$
FIELD=DATE_ADDED    ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=WHO_EDITED    ,ALIAS=                 ,USAGE=A10   ,ACTUAL=A10,$
FIELD=DATE_EDITED   ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=STATION_ID    ,ALIAS=                 ,USAGE=A12   ,ACTUAL=A12,$
FIELD=DIABETIC      ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=DIALYSIS      ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
```

EMPDB02 Master File

```
FILE=EMPDB02,SUFFIX=IMS,$
SEGNAME=EMPINFO,PARENT=,SEGTYPE=S1,$
FIELD=SEQFIELD       ,ALIAS=SEQFIELD.IMS     ,USAGE=P6    ,ACTUAL=P2  ,$
FIELD=REVSEQ         ,ALIAS=REVSEQ.IMS       ,USAGE=A6    ,ACTUAL=A6  ,$
FIELD=SALARY         ,ALIAS=                 ,USAGE=P8    ,ACTUAL=P4  ,$
FIELD=OT_HR_PAY      ,ALIAS=                 ,USAGE=P5    ,ACTUAL=P2  ,$
FIELD=SSN            ,ALIAS=SSN.IMS          ,USAGE=P9    ,ACTUAL=P3  ,$
FIELD=SSNALPHA       ,ALIAS=SSNALPHA.KEY     ,USAGE=A9    ,ACTUAL=A9  ,$
FIELD=EMPID          ,ALIAS=EMPID.IMS        ,USAGE=A12   ,ACTUAL=A12,$
FIELD=ELAST_NAME     ,ALIAS=ELNAME.IMS       ,USAGE=A12   ,ACTUAL=A12,$
FIELD=EFIRST_NAME    ,ALIAS=EFNAME.IMS       ,USAGE=A12   ,ACTUAL=A12,$
FIELD=DATE_OF_BIRTH  ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=RACE           ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=ADMIT_DATE     ,ALIAS=ADMDATE.IMS      ,USAGE=A08   ,ACTUAL=A08,$
FIELD=ADMIT_TYPE     ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=DISPOSITION    ,ALIAS=                 ,USAGE=A02   ,ACTUAL=A02,$
FIELD=TRANSFER_DATE  ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=ALLERGY1       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY2       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY3       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY4       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=HOUSING        ,ALIAS=                 ,USAGE=A03   ,ACTUAL=A03,$
FIELD=RPR            ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=URIN           ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=PRACTITIONAR   ,ALIAS=                 ,USAGE=A02   ,ACTUAL=A02,$
FIELD=SHIFT          ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=PATINET_ID     ,ALIAS=                 ,USAGE=P12   ,ACTUAL=P4  ,$
FIELD=WHO_ADDED      ,ALIAS=                 ,USAGE=A10   ,ACTUAL=A10,$
FIELD=DATE_ADDED     ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=WHO_EDITED     ,ALIAS=                 ,USAGE=A10   ,ACTUAL=A10,$
FIELD=DATE_EDITED    ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=STATION_ID     ,ALIAS=                 ,USAGE=A12   ,ACTUAL=A12,$
FIELD=DIABETIC       ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=DIALYSIS       ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
```

EMPDB03 Master File

```
FILE=EMPDB03,SUFFIX=IMS,$
SEGNAME=EMPINFO,PARENT=,SEGTYPE=S1,$
FIELD=SEQFIELD       ,ALIAS=SEQFIELD.IMS     ,USAGE=P6    ,ACTUAL=P2  ,$
FIELD=REVSEQ         ,ALIAS=REVSEQ.IMS       ,USAGE=A6    ,ACTUAL=A6  ,$
FIELD=SALARY         ,ALIAS=                 ,USAGE=P8    ,ACTUAL=P4  ,$
FIELD=OT_HR_PAY      ,ALIAS=                 ,USAGE=P5    ,ACTUAL=P2  ,$
FIELD=SSN            ,ALIAS=SSN.IMS          ,USAGE=P9    ,ACTUAL=P3  ,$
FIELD=SSNALPHA       ,ALIAS=SSNALPHA.KEY     ,USAGE=A9    ,ACTUAL=A9  ,$
FIELD=EMPID          ,ALIAS=EMPID.IMS        ,USAGE=A12   ,ACTUAL=A12,$
FIELD=ELAST_NAME     ,ALIAS=ELNAME.IMS       ,USAGE=A12   ,ACTUAL=A12,$
FIELD=EFIRST_NAME    ,ALIAS=EFNAME.IMS       ,USAGE=A12   ,ACTUAL=A12,$
FIELD=DATE_OF_BIRTH  ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=RACE           ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=ADMIT_DATE     ,ALIAS=ADMDATE.IMS      ,USAGE=A08   ,ACTUAL=A08,$
FIELD=ADMIT_TYPE     ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=DISPOSITION    ,ALIAS=                 ,USAGE=A02   ,ACTUAL=A02,$
FIELD=TRANSFER_DATE  ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=ALLERGY1       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY2       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY3       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY4       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=HOUSING        ,ALIAS=                 ,USAGE=A03   ,ACTUAL=A03,$
FIELD=RPR            ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=URIN           ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=PRACTITIONAR   ,ALIAS=                 ,USAGE=A02   ,ACTUAL=A02,$
FIELD=SHIFT          ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=PATINET_ID     ,ALIAS=                 ,USAGE=P12   ,ACTUAL=P4  ,$
FIELD=WHO_ADDED      ,ALIAS=                 ,USAGE=A10   ,ACTUAL=A10,$
FIELD=DATE_ADDED     ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=WHO_EDITED     ,ALIAS=                 ,USAGE=A10   ,ACTUAL=A10,$
FIELD=DATE_EDITED    ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=STATION_ID     ,ALIAS=                 ,USAGE=A12   ,ACTUAL=A12,$
FIELD=DIABETIC       ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=DIALYSIS       ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
```

EMPDBJ Master File

```
FILE=EMPDBJ,SUFFIX=IMS,$
SEGNAME=EMPINFO,PARENT=,SEGTYPE=S1,$
FIELD=SEQFIELD       ,ALIAS=SEQFIELD.IMS     ,USAGE=P6    ,ACTUAL=P2  ,$
FIELD=REVSEQ         ,ALIAS=REVSEQ.IMS       ,USAGE=A6    ,ACTUAL=A6  ,$
FIELD=SALARY         ,ALIAS=                 ,USAGE=P8    ,ACTUAL=P4  ,$
FIELD=OT_HR_PAY      ,ALIAS=                 ,USAGE=P5    ,ACTUAL=P2  ,$
FIELD=SSN            ,ALIAS=SSN.IMS          ,USAGE=P9    ,ACTUAL=P3  ,$
FIELD=SSNALPHA       ,ALIAS=SSNALPHA.KEY     ,USAGE=A9    ,ACTUAL=A9  ,$
FIELD=EMPID          ,ALIAS=EMPID.IMS        ,USAGE=A12   ,ACTUAL=A12,$
FIELD=ELAST_NAME     ,ALIAS=ELNAME.IMS       ,USAGE=A12   ,ACTUAL=A12,$
FIELD=EFIRST_NAME    ,ALIAS=EFNAME.IMS       ,USAGE=A12   ,ACTUAL=A12,$
FIELD=DATE_OF_BIRTH  ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=RACE           ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=ADMIT_DATE     ,ALIAS=ADMDATE.IMS      ,USAGE=A08   ,ACTUAL=A08,$
FIELD=ADMIT_TYPE     ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=DISPOSITION    ,ALIAS=                 ,USAGE=A02   ,ACTUAL=A02,$
FIELD=TRANSFER_DATE  ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=ALLERGY1       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY2       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY3       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=ALLERGY4       ,ALIAS=                 ,USAGE=A15   ,ACTUAL=A15,$
FIELD=HOUSING        ,ALIAS=                 ,USAGE=A03   ,ACTUAL=A03,$
FIELD=RPR            ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=URIN           ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=PRACTITIONAR   ,ALIAS=                 ,USAGE=A02   ,ACTUAL=A02,$
FIELD=SHIFT          ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=PATINET_ID     ,ALIAS=                 ,USAGE=P12   ,ACTUAL=P4  ,$
FIELD=WHO_ADDED      ,ALIAS=                 ,USAGE=A10   ,ACTUAL=A10,$
FIELD=DATE_ADDED     ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=WHO_EDITED     ,ALIAS=                 ,USAGE=A10   ,ACTUAL=A10,$
FIELD=DATE_EDITED    ,ALIAS=                 ,USAGE=A08   ,ACTUAL=A08,$
FIELD=STATION_ID     ,ALIAS=                 ,USAGE=A12   ,ACTUAL=A12,$
FIELD=DIABETIC       ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
FIELD=DIALYSIS       ,ALIAS=                 ,USAGE=A01   ,ACTUAL=A01,$
```

Example: Concatenating Partitioned PCBs

The following example illustrates how to concatenate partitioned PCBs:

```
PSB=EMPPSBJ, WRITE=NO, PCBNUMBER=4, PL1=NO, $
 SEGNAME=EMPINFO, KEYTYPE=S2, $
PSB=EMPPSBJ, WRITE=NO, PCBNUMBER=5, PL1=NO, $
PSB=EMPPSBJ, WRITE=NO, PCBNUMBER=6, PL1=NO, $
CONCATPCB=4, LOWVALUE=000000001, HIGHVALUE=000001667, $
CONCATPCB=5, LOWVALUE=000001668, HIGHVALUE=000003334, $
CONCATPCB=6, LOWVALUE=000003335, HIGHVALUE=000005000, $
```

Consider the following request. (The key field is named SSNALPHA):

```
SELECT SSNALPHA ...
FROM EMPDBJ
WHERE SSNALPHA = ‘000001775’;
```

The adapter satisfies the request using the EMPDB02 PCB only. If the WHERE clause had requested key values less than 000001775 (rather than equal to 000001775), the adapter would have used the EMPDB01 and EMPDB02 PCBs.

Note:

- If a retrieval request has no WHERE clause, or if the WHERE clause references a non-key field, the adapter accesses all PCBs.
- If any PCB listed in a concatenation lacks the LOWVALUE and HIGHVALUE key range specification, that PCB always participates in the retrieval.

Migrating From an Existing MVS Server

Migrating from an existing MVS server environment to OS/390 and z/OS consists of the following steps:

1. Duplicate any IMS SET commands from the existing server profile into the new edasprof.prf file using the Web Console *Workspace/Edit Files* option.
2. Restart the server.

If you have not already configured the server for DBCTL, configure it now; otherwise the server is ready for use.

Invoking IMS Stored Procedures

In this section:

This topic contains overview, configuration, and installation information for the following IMS/TM procedure:

- **CALLIMS LU6.2 for IMS/TM.** This procedure connects to IMS/TM using an IMS/APPC connection from a server for MVS.
- **CALLITOC OTMA for IMS/TM.** This procedure is available on all servers beginning with Version 5 Release 1.0.

The CALLIMS and CALLITOC procedures for IMS/TM hide the complexity of accessing IMS/TM transactions from the client application. The client application uses any major *standard* access protocol to connect to a server and invoke the adapter to access the IMS/TM transaction. The standard protocols include ODBC, JDBC, OLE DB, or any enabled client or connector.

The Adapter for IMS allows any client or connector to invoke a transaction running under the control of an IMS/TM transaction-processing monitor. It provides a means of building on existing applications that perform transaction processing against IMS/TM, to create new Web or client/server applications while reusing existing IMS transactions and program logic.

For related information, refer to the *API Reference* manual for API method calls and the *Stored Procedure* manual for details about writing Dialogue Manager procedures.

CALLIMS Procedure for IMS/TM

The CALLIMS procedure is part of the server for MVS. It is attached to and extended from any compatible API environment, including client applications and Hub Servers.

The CALLIMS procedure uses LU6 communications from a server for MVS to execute IMS/TM transactions, and passes the output to any client that connects to that server.

CALLITOC OTMA Procedure for IMS/TM

Starting with Version 5 Release 1.0, the OTMA procedure for IMS will be available for servers for MVS, Windows NT, and UNIX. It is attached to and extended from any compatible API environment, including client applications and Hub Servers.

The OTMA procedure fully utilizes the IMS TOC functionality. The IMS TCP/IP Connector (ITOC) and its equivalent product, IMS Connect, is available with IMS Version 7.1 and higher, and allows client TCP/IP communications to and from one or more IMS/TM regions. TOC and IMS Connect are IBM's implementation for allowing the execution of IMS/TM transactions from TCP/IP clients.

Transaction Processing With CALLIMS or CALLITOC

Example:

Processing a Transaction With CALLIMS

Processing a Transaction With CALLITOC OTMA

The following steps are performed when a client application calls an IMS/TM transaction using either the CALLIMS or CALLITOC procedure. The difference between the CALLIMS and the CALLITOC procedures is the communications method used to connect to the IMS/TM region from the client application. Also, CALLIMS requires a server for MVS to initiate the IMS/TM request. Any client on any platform can connect to the server for MVS.

1. The client application issues the API method call, EDARPC, to run a Dialogue Manager procedure residing on a server (either a Hub Server or a Full-Function Server).

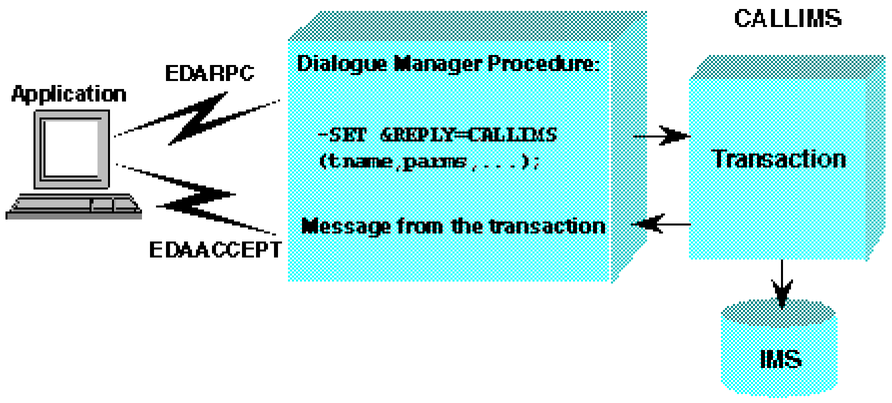
   Parameters are sent as part of the calling sequence for use by the procedure.

2. The Dialogue Manager procedure contains the command -SET, which executes CALLIMS or CALLITOC.
3. CALLIMS or CALLITOC establishes a connection to the IMS/TM environment and invokes the transaction, passing an IMS message as a parameter.
4. The IMS transaction receives and then processes the input message. It then passes one IMS message back to the server. The CALLIMS or CALLITOC procedure returns the message to the client application.
5. The client application issues the method call, EDAACCEPT, to accept the message.

For details on the syntax and use of API method calls, see the *API Reference* manual.

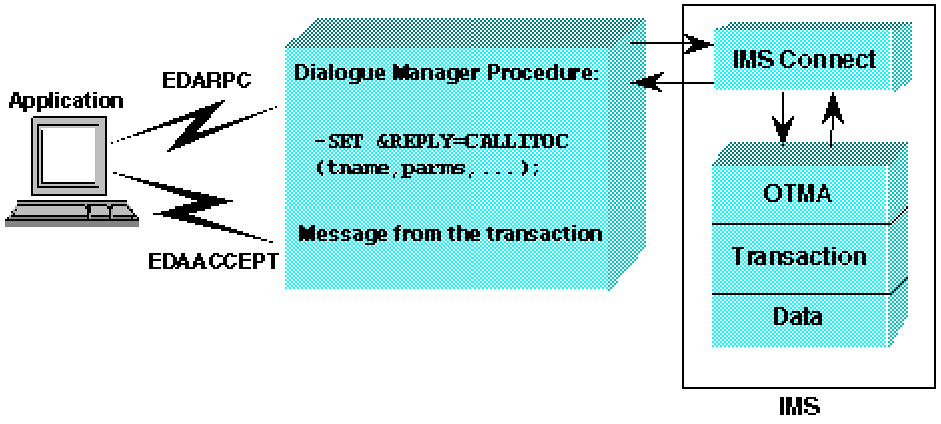
Example: Processing a Transaction With CALLIMS

The following figure illustrates transaction processing as initiated by the client application running a CALLIMS request.



Example: Processing a Transaction With CALLITOC OTMA

The following figure illustrates transaction processing as initiated by the client application running a CALLITOC OTMA request.



Using CALLIMS and CALLITOC

To call an IMS/TM transaction, execute either CALLIMS or CALLITOC from a Dialogue Manager procedure.

- CALLIMS invokes an IMS/TM transaction through the use of APPC/IMS. CALLIMS requires IMS Version 4.1 or higher (IMS/TM) and APPC/MVS.
- CALLITOC invokes an IMS/TM transaction through the use of IMS TOC or IMS Connect. CALLITOC requires IMS Version 5.1 or higher (IMS/TM) and a TOC or IMS connection to an IMS OTMA address space.

A sample RPC called CALLIMS is supplied within the server for MVS. It executes the IMS/TM PART transaction for verification and installation purposes.

A sample RPC called CALLIMSC is supplied within all servers in future releases. It also executes the IMS/TM PART transaction for verification and installation purposes.

Reference: Differences Between CALLIMS and CALLITOC

There are several differences between the CALLIMS and the CALLITOC procedures.

- The communications method used to connect to the IMS/TM region from the client application. CALLIMS uses LU6.2, CALLITOC uses TCP/IP to the mainframe using IMS Connect.
- CALLIMS requires a server for MVS to initiate the IMS/TM request.
- For CALLIMS, you must run a separate link job to allow CALLIMS to communicate with APPC/MVS. See the *Server for OS/390 and z/OS Configuration and Operations* manual for the steps needed to configure the Transaction Server for IMS.

How Data Is Returned With CALLIMS and CALLITOC

Example:

Viewing Message Codes From CALLIMS or CALLITOC

CALLIMS and CALLITOC return codes and data to the client application in a series of 72-byte messages. To retrieve the messages, the client application issues the method call EDAACCEPT.

Repeated calls to EDAACCEPT retrieve all messages from CALLIMS and CALLITOC.

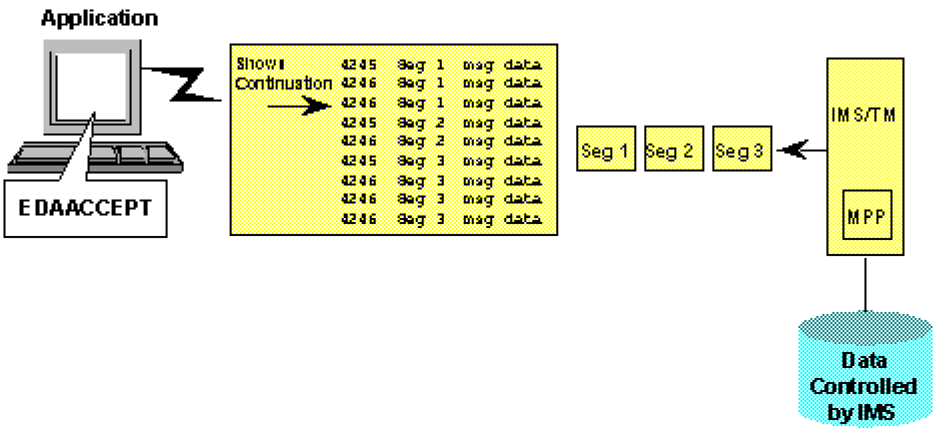
The client application must check the message code in the field scb.msg_code. (The session control block contains several fields pertaining to message processing. For example, scb.msg_code contains a message code and scb.msg_text contains the message text.)

A code of 4245 indicates that the message is coming from CALLIMS or CALLITOC.

For the client application to detect multiple output segments, the Adapter for IMS returns a message code of 4246 following the 4245, until the end of the segment. At that time, the message code is again 4245, indicating the beginning of a new segment.

Example: Viewing Message Codes From CALLIMS or CALLITOC

The following figure illustrates the message codes from CALLIMS or CALLITOC when there are multiple output segments. The client application normally changes the received format to a format acceptable to the end user.



Executing CALLIMS and CALLITOC

How to:

Execute CALLIMS

Execute CALLITOC OTMA

Example:

Using -SET to Execute CALLIMS

Using -SET to Execute CALLITOC

CALLIMS or CALLITOC is executed from a Dialogue Manager procedure with the command -SET.

Any of the variables described in the syntax that follows may be implicitly set in the Dialogue Manager procedure using the command -DEFAULTS, or explicitly set on the CALLIMS or CALLITOC call. The variables work the same way, whether set implicitly or explicitly. If both ways are used, the explicit setting takes precedence over the implicit setting. For examples of implicit settings, see *Using -SET to Execute CALLITOC* on page 13-74 and *Using -SET to Execute CALLIMS* on page 13-73.

Syntax: How to Execute CALLIMS

The dash (-) in the syntax below is used to allow multiple lines for a parameter list in a Dialogue Manager command, as in -SET.

```
-SET &REPLY = CALLIMS(&SYM,&TP,&MLEN,&MSG,&DELIM,&OPT,&UID,
- &PW,&SG,&PLU,&LMODE,&HEXCONV,'A1') ;
```

where:

&SYM

Is the symbolic destination name. This is a key value for the side information data set configuration file defined to APPC/MVS. APPC/MVS supports a symbolic destination name for determining the default APPLID for the IMS/TM region, with a log mode and transaction name. If a value is supplied for &SYM, either implicitly or explicitly, then &TP, &PLU, and &LMODE may be left blank. This field must be eight (8) characters in length.

&TP

Is the name of the IMS MPP transaction. This field must be eight (8) characters in length. Set &TP to blanks if you supply a value for &SYM.

&MLEN

Is the length of the IMS MPP message for the transaction. This is set with the Dialogue Manager LENGTH function in a –SET command.

&MSG

Is the input message for the IMS transaction. This message must exactly match the layout expected by the transaction.

&DELIM

Is a non-blank character string used to delimit individual segments in this IMS message. Each delimited segment has its own SEND. This variable is designed for MPP transactions that expect to retrieve multiple segments from the message queue. The length of &DELIM must be four (4) characters.

&OPT

Specifies whether CALLIMS waits for a response. Possible values are:

REPL assumes synchronous operation. CALLIMS waits for a return code or other response from IMS/TM.

NORP sends the message, then immediately returns control to the server. CALLIMS does not wait for a response.

The length of &OPT must be four (4) characters.

The following parameters are optional. If not used, the parameters must be padded with blanks.

&UID

Is the user ID that invokes the IMS MPP transaction. The length of &UID must be eight (8) characters.

&PW

Is the password used to invoke the IMS MPP transaction. The length of &PW must be eight (8) characters.

&SG

Is the security group that the user ID belongs to or is part of. The length of &SG must be eight (8) characters.

&PLU

Is the LU6.2 APPLID for the IMS/TM region. The length of &PLU must be eight (8) characters. Set &PLU to blanks if you supply a value for &SYM.

&LMODE

Is the log mode table entry name. The length of &LMODE must be eight (8) characters. Set &LMODE to blanks if you supply a value for &SYM.

HEXCONV

A value of ON will truncate the message returned after any hex character data. OFF is the default.

'A1'

Sets the format of the response to alphanumeric. It is required by -SET.

Syntax: How to Execute CALLITOC OTMA

The dash (-) in the syntax below is used to allow multiple lines for a parameter list in a Dialogue Manager command, as in -SET.

```
-?SET &REPLY = CALLITOC(&HOST, &PORT, &DSID, &TPNAME, &MLEN, &MESSAGE,
-? &USERID, &RUSERID, &RGROUP, &PASSWD, &DELIM, &HEXCONV, &OPTION, &OTMAEX 'A1');
```

where:

&HOST

Is the symbolic destination name. TCP/IP is the host address where IMS Connect or ITOC is running. *&host* is the value in the HOSTNAME=*host* in the HWS configuration file.

&PORT

Is the port number that IMS Connect or ITOC is listening on. *&port* is the value in the PORTID=*port* in the HWS configuration file.

`&DSID`

Is the value in the DATASTORE ID=*dsid* in the HWS configuration file.

`&TPNAME`

Is the name of the IMS MPP transaction. This field must be eight (8) characters in length.

`&MLEN`

Is the length of the IMS MPP message for the transaction. This is set with the Dialogue Manager LENGTH function in a –SET command.

`&MESSAGE`

Is the input message for the IMS transaction. This message must exactly match the layout expected by the transaction.

`&USERID`

Is the ITOC user ID. This user ID is reserved for future use.

`&RUSERID`

Is the user ID that is validated by RACF based on the RGROUP value.

`&RGROUP`

Is the XCF group validated by RACF. *&rgroup* is the value in the GROUP=*rgroup* in the HWS configuration file.

`&PASSWD`

Is the RACF password based on the RUSERID value.

`&DELIM`

Is a non-blank character string used to delimit individual segments in this IMS message. Each delimited segment has its own SEND. This variable is designed for MPP transactions that expect to retrieve multiple segments from the message queue. The length of &DELIM must be four (4) characters.

`&HEXCONV`

A value of ON will truncate the message returned after any hex character data. OFF is the default.

`&OPTION`

Specifies whether CALLITOC waits for a response. Possible values are:

`REPL` assumes synchronous operation. CALLITOC waits for a return code or other response from IMS/TM.

`NORP` sends the message, then immediately returns control to the server. CALLITOC does not wait for a response.

The length of &OPT must be four (4) characters.

`&OTMAEX`

Specifies which user message exit to use:

- *SAMPLE* indicates HWSSMPL0
- *SAMPL1* indicates HWSSMPL1

The sample user exits enable users to assign their own message formats to fit their business needs. See the IBM documentation for further information about sample user exits.

'A1'

Sets the format of the response to alphanumeric. It is required by -SET.

Example: Using -SET to Execute CALLIMS

The following is a sample Dialogue Manager procedure that uses the command -SET to execute CALLIMS. It is member CALLIMS of the data set.

In this example, values for variables are padded with blanks as required. The command -DEFAULTS enables you to specify default values in the Dialogue Manager procedure, but may be overridden by a client application.

```
-DEFAULTS &SYM    = '        '
-DEFAULTS &TP     = 'PART    '
-DEFAULTS &MSG    = 'AN960C10'
-DEFAULTS &DELIM  = '    '
-DEFAULTS &OPT    = 'REPL'
-DEFAULTS &UID    = '        '
-DEFAULTS &PW     = '        '
-DEFAULTS &SG     = '        '
-DEFAULTS &PLU    = 'SCMI41AX'
-DEFAULTS &LMODE  = 'LU62APPC'
-SET &MLEN  = &MSG.LENGTH;
-SET &REPLY =
CALLIMS(&SYM,&TP,&MLEN,&MSG,&DELIM,&OPT,&UID,&PW,&SG,&PLU,&LMODE,'A1');
```

Example: Using -SET to Execute CALLITOC

The following is a sample Dialogue Manager procedure that uses the command -SET to execute CALLITOC. For a server for MVS, it is member CALLIMS of the data set.

In this example, values for variables are padded with blanks as required.

```
-DEFAULT   &HOST=    'IBIMVS.IBI.COM ';
-DEFAULT   &PORT=     6683;
-DEFAULT   &DSID=    'IMS61          ';
-DEFAULT   &TPNAME=  'PART           ';
-DEFAULT   &OPTION=  'REPL           ';
-DEFAULT   &MESSAGE= 'AN960C10       ';
-DEFAULT   &USERID=  '               ';
-DEFAULT   &RUSERID= '               ';
-DEFAULT   &RGROUP=  'IMSGRP61       ';
-DEFAULT   &PASSWD=  '               ';
-DEFAULT   &DELIM=   '               ';
-DEFAULT   &HEXCONV= '               ';
-DEFAULTS  &OTMAEX = '*SAMPLE*'
-SET &MLEN = &MESSAGE.LENGTH;

-SET &REPLY = CALLITOC(&HOST,
-                      &PORT,
-                      &DSID,
-                      &TPNAME,
-                      &MLEN,
-                      &MESSAGE,
-                      &USERID,
-                      &RUSERID,
-                      &RGROUP,
-                      &PASSWD,
-                      &DELIM,
-                      &HEXCONV,
-                      &OPTION,
-                      &OTMAEX,
-                      'A1') ;
```

Installing CALLIMS and CALLITOC

Example:

Installing CALLIMS

Installing CALLITOC

Once you have created a Dialogue Manager procedure that includes CALLIMS or CALLITOC for IMS/TM OTMA, you must place it in the appropriate libraries for a server for MVS or a directory for a non-MVS server.

Note:

- CALLIMS is run from a server for MVS only. Therefore, CALLIMS is also placed in a library allocated to EDARPC on the server. The CALLIMS or CALLITOC load module is placed in a library allocated to STEPLIB on the server. If a function executes CALLITOC from a server for MVS, the procedure must also be placed in a library allocated to ddname EDARPC.
- CALLITOC runs from Windows NT or UNIX as well as MVS. The following is an example of the Windows NT directories that include the CALLITOC procedure as well as the CALLIMSC RPC function within a Version 5 Release 1.0 server:

  ```
  IBI\svr51\ffs\user\CALLITOC.DLL
  ```

  ```
  IBI\srv51\ffs\catalog\CALLIMSC.FEX
  ```

CALLIMS and CALLITOC are executed as a Dialogue Manager procedures. A Dialogue Manager procedure is referred to as a Remote Procedure Call (RPC) or a FOCEXEC. This procedure must be made available on a server for MVS.

- **CALLIMS.** It is recommended that you assign the procedure to a library allocated to the ddname EDARPC. A sample procedure is located in the EDARPC.DATA library, member CALLIMS. This procedure is modified and run, or it is used as a base for other procedures. You must first generate an APPC connection for CALLIMS before it is run. For an illustration, see *Installing CALLIMS* on page 13-76.
- **CALLITOC.** To run CALLITOC from a server for MVS, it is recommended that you assign the procedure to a library allocated to the ddname EDARPC. A sample procedure is located in the EDARPC.DATA library, member CALLIMSC. This procedure is modified and run or it is used as a base for other procedures.

  CALLITOC runs from Windows NT or UNIX as well as MVS. For an illustration, see *Installing CALLITOC* on page 13-76.

Example: Installing CALLIMS

The following JCL is located in member GENEAPPC in the library EDACTL.DATA. It must be modified and run. Make sure that the load module CALLIMS that is created in SYSLMOD is made available to the EDASERVER STEPLIB ahead of any load libraries.

```
//*********************************************************************
//* NAME:     GENEAPPC JCL
//*
//* FUNCTION: LINKEDIT APPC/MVS STUBS INTO CALLIMS
//*
//* PROC SYMBOLIC PARAMETERS:
//*       1. qualif MUST BE CHANGED TO THE HIGH LEVEL QUALIFIER
//*           USED FOR THE PROGRAM DATASETS UNLOAD FROM THE MEDIA
//*
//*********************************************************************

//APPROC   PROC PREFIX='qualif'
//*
//LKED     EXEC PGM=IEWL,PARM='RENT,LIST,NOXREF,LET'
//SYSLIB    DD DISP=SHR,DSN=SYS1.CSSLIB <- change this to your APPC
                                           library that contains member ATBPBI
//SYSIN     DD DISP=SHR,DSN=&PREFIX..EDALIB.LOAD <- EDALIB libraries from
                                                    the install
//SYSLMOD   DD DISP=SHR,DSN=&PREFIX..EDALIB.LOAD
//MAINTAIN DD DISP=SHR,DSN=&PREFIX..EDALIB.DATA
//SYSUT1    DD UNIT=SYSDA,SPACE=(100,(50,50))
//SYSPRINT DD SYSOUT=A
//*
//APPROC    PEND
//GENFAPPC EXEC APPROC
//LKED.SYSLIN    DD *
  INCLUDE SYSIN(CALLIMS)
  INCLUDE SYSLIB(ATBPBI)
  INCLUDE MAINTAIN(CALLIMS)
  NAME CALLIMS(R)
/*
```

Example: Installing CALLITOC

The following is an example of the Windows NT directories that include the CALLITOC procedure as well as the CALLIMSC RPC function within a Version 5 Release 1.0 server:

```
IBI\svr51\ffs\user\CALLITOC.DLL
```

```
IBI\srv51\ffs\catalog\CALLIMSC.FEX
```

There are no pre-installation requirements for the CALLITOC procedure.

Storing Multiple Messages In a Server File for Later Queries

If you are using a front-end application, such as PowerBuilder, which cannot handle multiple messages, a method is needed to convert the messages into an answer set. This is especially important when using CALLIMS or CALLITOC, which only return messages.

Example: Storing Multiple Messages for Later Execution

The following stored procedure executes a CALLIMS procedure that returns five (5) messages. Instead of sending the messages directly to the client, the stored procedure stores the messages in a file. A subsequent SELECT is performed to extract data, as an answer set, from the file. The stored procedure statements required for saving messages in the file are in **bold** letters.

```
-SET &EMGSRV = 'FILE';
DYNAM ALLOC FILE EMGFILE DA qualif.EMGFIL SPACE 2,2 TRACKS UNIT SYSDA -
RECFM FB LRECL 80 BLKSIZE 1600
SET EMGSRV=&EMGSRV
SET PAUSE=OFF
-RUN
-DEFAULT &SYM   = '        '
-DEFAULT &TP    = 'PART    '
-DEFAULT &MSG   = 'AN960C10'
-DEFAULT &DELIM= '        '
-DEFAULT &OPT   = 'REPL    '
-DEFAULT &UID   = '        '
-DEFAULT &PW    = '        '
-DEFAULT &SG    = '        '
-DEFAULT &PLU   = 'SCMI41AX'
-DEFAULT &LMODE= 'LU62APPC'
-SET &MLEN = &MSG.LENGTH ;
-SET &REPLY = CALLIMS(&SYM,&TP,&MLEN,&MSG,&DELIM,&OPT,&UID,
-  &PW,&SG,&PLU,&LMODE,'A1') ;
DYNAM FREE DDN EMGFILE
-RUN
DYNAM ALLOC FILE EMGSRV1 DA qualif.EMGFIL SHR REU
-RUN
SQL
SELECT COL1 FROM EMGSRV1;
TABLE
ON TABLE PCHOLD FORMAT ALPHA
END
DYNAM FREE DDN EMGSRV1
-RUN
```

For the SELECT COL1 FROM EMGSRV1 statement to work, a Master File has to be predefined for the file. The following is the Master File, EMGSRV1, used for this example:

```
FILENAME=EMGSRV1,SUFFIX=FIX
SEGNAME=ORIGIN,SEGTYPE=S1
 FIELDNAME=COL1,FIRST40,A40,$
 FIELDNAME=COL2,LAST40,A40,$
```

The following is the output, from CALLIMS, which is placed into the file:

```
(FOC4245)  :      Part...........     AN960C10; Desc........... WASHER
(FOC4246)  :
(FOC4245)  :      Proc Code......           74; Inv Code.......        2
(FOC4245)  :      Make Dept......        12-00; Plan Rev Num...
(FOC4245)  :      Make Time......           63; Comm Code......       14
```

Note: The second line is a FOC4246 message line, which indicates a continuation of the previous line. The sample stored procedure returns the first 40 bytes of all five (5) records to the client. The stored procedure can be set to return the entire message.

CHAPTER 14

Using the Adapter for IMS Transactions

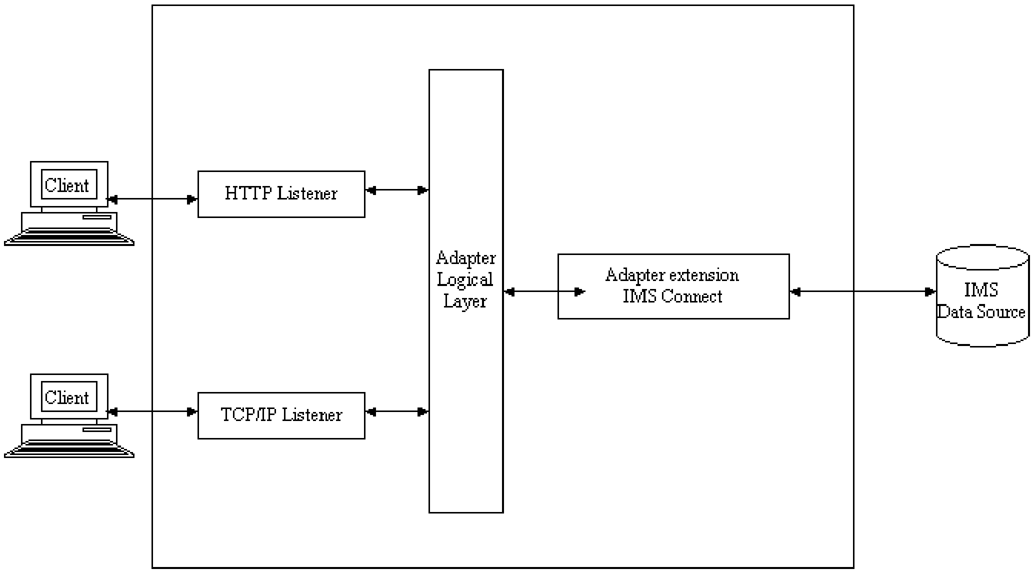
Topics:

The Adapter for IMS Transactions processes input parameters, creates and sends requests to the IMS Transaction Server, and creates an answer set based on the received response.

Note: Throughout this chapter, the IMS Transaction Manager is referred to as IMS.

Preparing the IMS Transactions Environment

The following diagram illustrates the basic flow of the Adapter for IMS Transactions:



This diagram illustrates how the client application communicates with the Application Adapter Server using the TCP/IP or HTTP protocol. The communication to the IMS region uses IMS Connect.

To provide for transparent execution of IMS transactions, metadata describing the program input and output areas is held on the server. This metadata can be derived from COBOL copy book entries, if available.

A Web Console is available as the primary configuration and maintenance tool. It is used to add or change adapter communication parameters and in creating or maintaining the metadata associated with the programs to be executed.

Supported Platforms and Release Information for IMS Transactions

The following platforms are supported:

- Microsoft Windows using IMS Connect /OTMA (IMS Version 5 or higher).
- UNIX using IMS Connect /OTMA (IMS Version 5 or higher).
- OS/390 and z/OS using OTMA.

Note:

- IBMs IMS Connect provides high-performance communications for IMS between one or more TCP/IP or local/390 clients and one or more IMS systems.
- IMS(TM) Open Transaction Manager Access (OTMA) is a transaction-based, connectionless client/server protocol. Though easily generalized, its implementation is specific to IMS in an MVS(TM) sysplex environment. The domain of the protocol is restricted to the domain of the MVS Cross-System Coupling Facility (XCF).
- OTMA addresses the problem of connecting a client to a server so that the client can support a large network, or a large number of sessions, while maintaining high performance.

Configuring the Adapter for IMS Transactions

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to IMS, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).

- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to IMS takes place when the first query that references that connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the last SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the last SET CONNECTION_ATTRIBUTES command.

For detailed instructions about declaring connection attributes for your operating system, see *Configuring the Adapter on Windows and UNIX* on page 14-4 or *Configuring the Adapter on OS/390 and z/OS* on page 14-8.

Configuring the Adapter on Windows and UNIX

The Adapter for IMS Transactions must be configured in order to gain access to IMS programs, create metadata and build the repository. The HWS command *viewhws* can provide some of the information required for configuration. The setup information becomes the content of the of the Adapter for IMS Transactions communications nodes.

To configure the Application Adapter Server for IMS access, perform these steps:

1. Configure the Adapter for TCP/IP on Windows or UNIX.
2. Create Synonyms.

Before you begin this configuration process, you must start the Web Console.

Procedure: How to Declare Connection Attributes From the Web Console on Windows and UNIX

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *Procedures* group folder, then expand the *IMS TRAN* adapter folder and click a connection. The Add IMS TRAN to Configuration pane opens.
3. Provide valid values for all input fields.

Parameter	**Description**
`CONNECTION NAME`	1-8 character logical name. Select a name that reflects the IMS region you are configuring the connection for.
`HOST`	DNS name of the host LPAR of the target IMS region. You can also code the four part IP address.
`PORT`	Port number on which the IMS Connect is listening. This value may be obtained from viewhws. See *Output from Viewhws* on page 14-7.
`DATASTORE`	ID of the IMS region that provides the transaction processing. This value may be obtained from viewhws. See *Output from Viewhws* on page 14-7.
`XCFGROUP`	IMS group name defined in the IMS startup JCL. This value may be obtained from viewhws.
`SECURITY`	There are two methods by which a user can be authenticated when connecting to an IMS Application: • **Explicit.** If you require all connected users to connect to IMS using the same security profile, select *Explicit* and provide a User ID and Password. • **Password Passthru.** If Password Passthru is selected, the user ID and password sent from the client application will be used to connect to IMS. If Explicit security is selected, two additional input fields are shown for User and Password. Enter values that are valid for logging on to IMS. See *Security on Microsoft Windows and UNIX* on page 14-7 for more information on the different security models available.
`User`	A user ID that is valid for an IMS logon.

Parameter	Description
`Password`	The password associated with the user ID above.
`Select profile`	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure* to continue.

Once the adapter is configured, the connection name appears in the navigation pane under Configured. If you left-click the connection name, several options become available:

- **Add connection.** This option allows you to add another connection.
- **View Settings.** This option displays the current default connection.
- **Change settings.** This option enables you to revise conversion, optimization, session, and other settings in one place.
- **Remove.** This option removes the connection from the server.

Following the configuration, refer to *Creating Synonyms* on page 14-12 for information about how to create metadata.

Syntax: How to Declare Connection Attributes Manually on Windows and UNIX

```
ENGINE IMSTRAN SET CONNECTION_ATTRIBUTES connection /,: "HOST dns_name
COM PORT port DATASTORE IMS_region XCFGROUP IMS_group"
```

where:

`IMSTRAN`

Indicates the Adapter for IMS Transactions.

`connection`

1-8 character logical name. Select a name that reflects the IMS region you are configuring the connection for.

`dns_name`

DNS name of the host LPAR of the target IMS region. You can also code the four part IP address.

`port`

Port number on which the IMS Connect is listening. This value may be obtained from viewhws.

`IMS_region`

ID of the IMS region that provides the transaction processing. This value may be obtained from viewhws.

`IMS_group`

IMS group name defined in the IMS startup JCL. This value may be obtained from viewhws.

For information about security options, see *Security on Microsoft Windows and UNIX* on page 14-7.

Example: Declaring Connection Attributes

```
ENGINE IMSTRAN SET CONNECTION_ATTRIBUTES IMS8C /,: "HOST IBIMVS.IBI.COM
PORT 6686 DATASTORE IMS8C XCFGROUP IMSGRP8C"
```

Example: Output from Viewhws

```
IEE600I        REPLY TO 01 IS;VIEWHWS
HWSC0001I      HWS ID=HWSB Racf=N
HWSC0001I      Maxsoc=2000 Timeout=8888 23
HWSC0000I      *IMS CONNECT READY* HWSB
HWSC0001I      Datastore=IMS7B Status=ACTIVE
HWSC0001I          Group=IMSGRP7B Memb
HWSC0001I      Datastore=IMS6A Status=ACTIVE
HWSC0001I          Group=IMSGRP6A Member=HWSMEM6A
HWSC0001I          Target Member=TMEM6A
HWSC0001I      Datastore=IMS7A Status=NOT ACTIVE
HWSC0001I          Group=IMSGRP7A Member=HWSMEM7A
HWSC0001I          Target Member=TMEM7A
HWSC0001I          Port=6685 Status=ACTIVE
HWSC0001I          No active Clients
```

Reference: Security on Microsoft Windows and UNIX

If the adapter is deployed on Microsoft Windows or UNIX, starting the server without security is recommended. The client application should provide the User ID and Password to be used for the connection to IMS. This User ID and Password is passed through to IMS for validation. The server does not validate the User ID and Password with the operating system.

If the **Explicit** option is selected and security information is entered in the adapter configuration window, it overrides this security mode and a single User ID and Password are used for all connected users.

If the adapter is started with security, and **Password Passthru** is selected, the local security profiles must be synchronized with those of the IMS platform, thereby creating additional security maintenance overhead.

Configuring the Adapter on OS/390 and z/OS

How to:

Declare Connection Attributes From the Web Console on OS/390 and z/OS

Declare Connection Attributes Manually on OS/390 and z/OS

Reference:

Security on OS/390 and z/OS

Example:

Output from /DIS OTMA

Declaring Connections Attributes on OS/390 and z/OS

Configuring the adapter consists of specifying connection and authentication information for each connection on OS/390 and z/OS.

Procedure: How to Declare Connection Attributes From the Web Console on OS/390 and z/OS

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *Procedures* group folder, then expand the *IMS TRAN* adapter folder and click a connection. The Add IMS TRAN to Configuration pane opens.
3. Provide valid values for all input fields.

Parameter	Description
`CONNECTION NAME`	1-8 character logical name. Select a name that reflects the IMS region you are configuring the connection for.
`XCFMEMB`	Target member name. This value may be obtained from /DIS OTMA. See *Output from /DIS OTMA* on page 14-10.
`XCFGROUP`	Target member group name. This value may be obtained from /DIS OTMA. See *Output from /DIS OTMA* on page 14-10.
`XCFUSMEMB`	User ID.
`RACFGROUP`	RACF group.

Parameter	Description
`SECURITY`	There are two methods by which a user can be authenticated when connecting to an IMS Application: • **Explicit.** If you require all connected users to connect to IMS using the same security profile, select *Explicit* and provide a User ID and Password. • **Password Passthru.** If Password Passthru is selected, the user ID and password sent from the client application will be used to connect to IMS. If Explicit security is selected, two additional input fields are shown for User and Password. Enter values that are valid for logging on to IMS. See *Security on Microsoft Windows and UNIX* on page 14-7 for more information on the different security models available.
`User`	Is a user ID that is valid for an IMS logon.
`Password`	Is the password associated with the user ID above.
`Server Profile`	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure* to continue.

Once the adapter is configured, the connection name appears in the navigation pane under Configured. If you left-click the connection name, several options become available:

- **Add connection.** This option allows you to add another connection.
- **View Settings.** This option displays the current default connection.
- **Change settings.** This option enables you to revise conversion, optimization, session, and other settings in one place.
- **Remove.** This option removes the connection from the server.

Following the configuration, refer to *Creating Synonyms* on page 14-12 for information about how to create metadata.

Example: Output from /DIS OTMA

```
R 20,/DIS OTMA
IEE600I REPLY TO 20 IS;/DIS OTMA
DFS000I      GROUP/MEMBER      XCF-STATUS   USER-STATUS    SECURITY
IM8C
DFS000I      IMSGRP8C
IM8C
DFS000I      -TMEM8C           ACTIVE       SERVER         NONE
IM8C
DFS000I      -HWSMEM8C         ACTIVE       ACCEPT TRAFFIC
IM8C
```

Reference: Security on OS/390 and z/OS

If the adapter is deployed on OS/390 and z/OS, you must start the server with security turned on (the default).

The client application must provide a valid User ID and Password that is authenticated by the Adapter for IMS Transactions with calls to security packages (RACF, ACF2 and Top Secret). If the User ID and Password are valid, the subsequent connection to IMS is Trusted.

Syntax: How to Declare Connection Attributes Manually on OS/390 and z/OS

```
ENGINE IMSTRAN SET CONNECTION_ATTRIBUTES connection /,: "XCFMEMB
 target_member_name XCFGROUP target_member_group_name XCFUSMEMB userid
 RACFGROUP racf_group"
```

where:

connection

1-8 character logical name. Select a name that reflects the IMS region you are configuring the connection for.

target_member_name

Name of the target member. This value may be obtained from /DIS OTMA.

target_member_group_name

Name of the target member group. This value may be obtained from /DIS OTMA.

userid

User ID.

racf_group

RACF group.

For information about security options, see *Security on OS/390 and z/OS* on page 14-10.

Example: Declaring Connections Attributes on OS/390 and z/OS

```
ENGINE IMSTRAN SET CONNECTION_ATTRIBUTES IMS8C /,: "XCFMEMB TMEM8C
XCFGROUP IMSGRP8C XCFUSMEMB EDAERH RACFGROUP EDA"
```

Overriding the Default Connection

Once the connections have been defined, the connection named in the last SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET CONNECTION_ATTRIBUTES command.

You can also include the CONNECTION attribute in the Access File of the business component specified in the current query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION attribute is automatically included in the Access File. This attribute supersedes the default connection.

Syntax: How to Change the Default Connection

```
ENGINE IMSSTRAN SET DEFAULT_CONNECTION connection
```

where:

`IMSTRAN`

Indicates the Adapter for IMS Transactions.

`connection`

Is the connection name specified in a previously issued SET CONNECTION_ATTRIBUTES command. If this connection name has not been previously declared, a FOC44221 message is issued.

Note: If you use the SET DEFAULT_CONNECTION command more than once, the connection_name specified in the last command will be the active one.

Example: Selecting CON1 as the Default Connection

The following example selects the previously defined connection named CON1 as the default IMS connection:

```
ENGINE IMSSTRAN SET DEFAULT_CONNECTION CON1
```

Managing IMS Transaction Metadata

When the server invokes a transaction or procedure, its needs to know how to build the request, what parameters to pass, and how to format an answer set from the response. For each transaction the server will execute, you must create a synonym that describes the layout of the request/response area.

Creating Synonyms

Synonyms define unique names (or aliases) for each transaction or procedure that is accessible from the server. Synonyms are useful because they hide the underlying transaction or procedure from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows the input parameters and the response layout to be moved while allowing client applications to continue functioning without modification. For example, moving a transaction or procedure from a test region to production. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym From the Web Console on Windows and UNIX

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The right pane displays selection options for the adapter.
4. Under Collection of Cobol definitions, enter a directory name to display all members in the directory path.

   If you wish to limit retrieval, you can enter a File Name and/or File Extension:

   - In the File Name box, enter a full name or a partial name with a wildcard symbol '%'. A full name returns just that entry. A name with a wildcard symbol may return many entries.
   - In the File Extension box, enter an extension to limit the filetypes returned.
5. Click *Submit*.

   The Create Synonym pane (Step 2of 2) opens.
6. Type the name of the synonym.
7. Type the name of the IMS transaction program.

   The Cobol Copybook(s) appear on the screen. You can choose the same Copybook or different Copybooks for input and output parameters.

   Note: There can be no more than one set of input/output Copybooks per synonym.
8. Enter the following additional parameters, as required:

Parameter	**Description**
Select Application	Select an application directory. The default value is baseapp.
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

9. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

   When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

10. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

11. Optionally, select *Customize options* to customize how the COBOL FD is translated. For details, about these options, see *Customization Options for COBOL File Descriptions* on page 14-18. If you do not select the check box, default translation settings are applied.

12. Click *Create Synonym*. The result of the Create Synonym process is a Master File, which is created and added under the specified application directory.

13. A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Procedure: How to Create a Synonym From the Web Console on OS/390 and z/OS

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The right pane displays selection options for the adapter.

4. Under File System Selection, choose one of the following options from the drop-down list:

   - *Fully qualified PDS name* to indicate a partitioned dataset on MVS.

     In the input boxes provided, type a PDS name preceded by // and a Member name containing the location of the COBOL FD source. If you wish, you can filter the member name using a wildcard character (%).

     or

   - *Absolute HFS directory pathname* to indicate a hierarchical file structure on USS.

     In the input boxes provided, type a Directory name to specify the HFS location that contains the COBOL FD and a File name and File extension. If you wish, you can filter the file and extension using a wildcard character (%).

5. Click *Submit*. The Create Synonym pane (Step 2 of 2) opens:

   If selected, the filtered COBOL definition is displayed at the top of the pane.

6. Type the name of the synonym.

7. Type the name of the IMS transaction program.

8. Enter the following parameters, as required:

Parameter	**Description**
Select Application	Select an application directory. baseapp is the default value.
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Parameter	**Description**
Input Segment Definition	Is the name of the COBOL FD file that contains information about input parameters. (Omit this entry if the transaction does not have input parameters.) There can be no more than one set of input COBOL Copybooks per synonym. You can choose the same Copybook or different Copybooks for input and output parameters.
Output Segment Definition	Is the name of the COBOL FD file that contains information about output parameters. (Omit this entry if the transaction does not have output parameters.) There can be no more than one set of output Copybooks per synonym. You can choose the same Copybook or different Copybooks for input and output parameters.

9. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

10. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

11. Optionally, select *Customize options* to customize how the COBOL FD is translated. For details, about these options, see *Customization Options for COBOL File Descriptions* on page 14-18. If you do not select the check box, default translation settings are applied.

12. Click *Create Synonym*. The result of the Create Synonym process is a Master File, which is created and added under the specified application directory.

13. A status window displays the message:

```
All Synonyms Created Successfully
```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: Customization Options for COBOL File Descriptions

You can customize how a COBOL FD is translated by selecting *Customize options* in the Synonym creation pane. The following options are added to the right pane:

Parameter	**Definition**
On Error	Choose *Continue* to continue generating the Master File when an error occurs. Choose *Abort* to stop generating the Master File when an error occurs. Choose *Comment* to produce a commented Master File when an error occurs. Continue is the default value.
Hyphens as	Choose *No* to remove all hyphens in the COBOL name from the Master File field names. Choose *Yes* to replace all hyphens in the COBOL name with the underscore character. *Yes* is the default value.
Redefines	You may treat COBOL REDEFINE fields in one of three ways: Choose *Segments* to describe REDEFINE fields as segments in the Master File. Segments is the default value. Choose *Comments* to describe REDEFINE fields as comments in the Master File. Choose *None* to exclude REDEFINE fields altogether.
Occurs as	Choose *Segments* to describe OCCURS structures as segments, or *Field* otherwise. Segments is the default value.
Alignment	Choose *Yes* to insert slack bytes into a record to ensure alignment of numeric fields. Choose *No* to generate Master Files without alignment of slack bytes. No is the default value.

Parameter	**Definition**
Number of Hyphens to skip	FD Translator removes characters from the left, up to and including the Nth hyphen (depending on the value of N chosen in the menu). A value of *0* will retain the entire COBOL name. *All* means all prefixes will be removed. *0* is the default value.
Order fields	Choose *Yes* to generate Order fields in a Master File. Choose *No* to generate a Master File without Order fields. *No* is the default value.
Level 88 as	Choose *Comment* to include COBOL Level 88 fields as comments in the Master Files. Choose *Skip* to exclude level 88 fields. *Skip* is the default value.
Zoned Numeric Fields	Sets how zoned numeric values will be stored.
Numeric Fields Edit Options	
Zeroes	Choose *Suppress* to suppress printing of the digit zero for a field whose value is zero. Choose *Display* to display leading zeroes, for example, 00124. Choose *None* for no formatting.
Negative value	Choose *Bracket* to bracket negative values, for example, (1234). Choose *Credit* to credit negative values, for example, 1234 CR. Choose *None* for no formatting.
Dollar Sign	Choose *Floating* to display a floating dollar sign and commas, for example, $1,1234. Choose *Fixed* to display a fixed dollar sign and commas, for example, $ 1,1234. Choose *None* for no formatting.

Parameter	**Definition**
Separate Thousands	Choose *Comma* to include commas where appropriate. Choose *None* for no formatting.

For additional information about customization options, see Appendix D, *Translating COBOL File Descriptions*.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '| ', '&', ' (', ') ', ' <', '> ', '' ', ' =', '''

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Example: Master File for IMS TRAN Adapter

```
FILENAME=IMSTPART, SUFFIX=IMSTRAN , $
  SEGMENT=SEG1, SEGTYPE=S0, $
$  GROUP=PARTKEY_INPUT, USAGE=A50, ACTUAL=A50, $
    FIELDNAME=PARTKEY, USAGE=A50, ACTUAL=A50, $
  SEGMENT=SEG11, SEGTYPE=S0, PARENT=SEG1, $
$  GROUP=PARTROOT_OUTPUT, USAGE=A268, ACTUAL=A268, $
    FIELDNAME=FILLER, USAGE=A3, ACTUAL=A3, $
    FIELDNAME=RECTYPE, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=FILLER, USAGE=A22, ACTUAL=A22, $
    FIELDNAME=PARTNUM, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=FILLER, USAGE=A18, ACTUAL=A18, $
    FIELDNAME=DESCRIPTION, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=FILLER, USAGE=A26, ACTUAL=A26, $
    FIELDNAME=PROCCODE, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=FILLER, USAGE=A18, ACTUAL=A18, $
    FIELDNAME=INVCODE, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=FILLER, USAGE=A26, ACTUAL=A26, $
    FIELDNAME=MAKEDEPT, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=FILLER, USAGE=A18, ACTUAL=A18, $
    FIELDNAME=PREVNO, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=FILLER, USAGE=A26, ACTUAL=A26, $
    FIELDNAME=MAKETIME, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=FILLER, USAGE=A18, ACTUAL=A18, $
    FIELDNAME=COMMCODE, USAGE=A8, ACTUAL=A8, $
```

SEG1, the input segment, was generated from the following COBOL FD:

```
01  PARTKEY-INPUT.
           05 PARTKEY             PIC X(50).
```

SEG11, the output segment, was generated from the following COBOLFD:

```
01  PARTROOT-OUTPUT.
          05 FILLER         PIC X(3).
          05 RECTYPE        PIC X(1).
          05 FILLER         PIC X(22).
          05 PARTNUM        PIC X(12).
          05 FILLER         PIC X(18).
          05 DESCRIPTION    PIC X(20).
          05 FILLER         PIC X(26).
          05 PROCCODE       PIC X(12).
          05 FILLER         PIC X(18).
          05 INVCODE        PIC X(8).
          05 FILLER         PIC X(26).
          05 MAKEDEPT       PIC X(12).
          05 FILLER         PIC X(18).
          05 PREVNO         PIC X(8).
          05 FILLER         PIC X(26).
          05 MAKETIME       PIC X(12).
          05 FILLER         PIC X(18).
          05 COMMCODE       PIC X(8).
```

Reference: Access File Attributes

Keyword	Description
`SEGNAME`	Is the name of the input segment in the Master File.
`CONNECTION`	Indicates the connection_name as previously specified in a SET CONNECTION_ATTRIBUTES command. Defaults to the default connection.
`TRANSACTION`	Is the name of transaction to be executed.

Example: Access File for Transaction/Program

```
SEGNAME=SEG1, CONNECTION=IMS8C, TRANSACTION=part, $
```

Note: Do not change the FLOAT or INTEGER parameter values.

Invoking an IMS Transaction

How to:

Invoke an IMS Transaction Using TABLE

Invoke an IMS Transaction Using SELECT

Invoke an IMS Transaction Using EX

Example:

Invoking the IMSTRAN Transaction

To invoke a IMS transaction, you issue a Data Manipulation Language (DML) request, also known as a TABLE command, an SQL SELECT statement, or an EX command. (DML is iWay's internal retrieval language.) For information about the Data Manipulation Language, see the *iWay Stored Procedure Reference* manual.

Syntax: How to Invoke an IMS Transaction Using TABLE

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
   .
   .
   .
END
```

where:

`synonym`

Is the synonym of the IMS transaction you want to invoke.

`parameter`

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the transaction does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

`*`

Indicates that you want to display all indicated parameters.

`IF/WHERE`

Is used if you want to pass values to input parameters.

`in-parameter`

Is the name of an input parameter to which you want to pass a value.

`value`

Is the value you are passing to an input parameter.

Syntax: How to Invoke an IMS Transaction Using SELECT

```
SQL
SELECT [parameter [parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
   .
   .
   .
END
```

where:

`synonym`

Is the synonym of the IMS transaction you want to invoke.

`parameter`

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the transaction does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

`*`

Indicates that you want to display all indicated parameters.

`WHERE`

Is used if you want to pass values to input parameters.

You must specify the value of each parameter on a separate line.

`in-parameter`

Is the name of an input parameter to which you want to pass a value.

`value`

Is the value you are passing to an input parameter.

Syntax: How to Invoke an IMS Transaction Using EX

```
[app_name_space/]syn_name  1=parm1_val,..., N=parmN_val
```

```
[app_name_space/]syn_name  prm1_name=prm1_val,... ,
  prmN_name=prmN_val
```

```
[app_name_space/]syn_name [, parm1_val [,...[, parmN_val]]]
```

where:

app_name_space

Is the apps directory name under which the synonyms are stored. This value is optional.

syn_name

Is the user friendly name of a repository entry that represents the object to be executed in the vendor environment.

parmname

Is the name of the parameter taken from the input metadata description.

1=parm1_val
N=parmN_val
prm1_name
prm1_val
parmN_val

Are the parameter values that match the input metadata description.

Example: Invoking the IMSTRAN Transaction

The following requests produce identical output:

```
TABLE FILE IMSTPART
PRINT PARTNUM DESCRIPTION PROCCODE INVCODE MAKEDEPT PREVNO MAKETIME
COMMCODE
IF PARTKEY EQ 'an960c10'
END
```

```
SQL
SELECT
PARTNUM,DESCRIPTION,PROCCODE,INVCODE,MAKEDEPT,PREVNO,MAKETIME,COMMCODE
FROM IMSTPART
WHERE PARTKEY='an960c10'
END
```

The output is:

```
PARTNUM  DESCRIPTION PROCCODE INVCODE MAKEDEPT REVNO MAKETIME COMMCODE
-------  ----------- -------- ------- -------- ----- -------- --------
AN960C10 WASHER      74       2       1        2-00  63       14
```

The following EX commands invoke the IMSTRAN transaction:

```
EX IMSTPART 'AN960C10'
```

```
EX IMSTPART PARTKEY='AN960C10'
```

```
EX IMSTPART 1='AN960C10'
```

The IMSTPART transaction returns the data associated with PARTKEY 'AN960C10'.

```
FILE=SQLOUT ,SUFFIX=FIX ,$ SEGNAME=SQLOUT ,$ FIELD=FILLER , E2 ,A3 ,A3 ,
,$ FIELD=RECTYPE , E3 ,A1 ,A1 , ,$ FIELD=FILLER , E4 ,A22 ,A22 , ,$
FIELD=PARTNUM , E5 ,A12 ,A12 , ,$ FIELD=FILLER , E6 ,A18 ,A18 , ,$
FIELD=DESCRIPTION , E7 ,A20 ,A20 , ,$ FIELD=FILLER , E8 ,A26 ,A26 , ,$
FIELD=PROCCODE , E9 ,A12 ,A12 , ,$ FIELD=FILLER , E10 ,A18 ,A18 , ,$
FIELD=INVCODE , E11 ,A8 ,A8 , ,$ FIELD=FILLER , E12 ,A26 ,A26 , ,$
FIELD=MAKEDEPT , E13 ,A12 ,A12 , ,$ FIELD=FILLER , E14 ,A18 ,A18 , ,$
FIELD=PREVNO , E15 ,A8 ,A8 , ,$ FIELD=FILLER , E16 ,A26 ,A26 , ,$
FIELD=MAKETIME , E17 ,A12 ,A12 , ,$ FIELD=FILLER , E18 ,A18 ,A18 , ,$
FIELD=COMMCODE , E19 ,A8 ,A8 , ,$ Part........... AN960C10;
Desc........... WASHER Proc Code...... 74; Inv Code....... 2 Make
Dept...... 12-00; Plan Rev Num... Make Time...... 63; Comm Code...... 14
```

CHAPTER 15

Using the Adapter for Information Manager

Topics:

These topics provide an overview of the Adapter for InfoMan. They also provide hardware and software requirements and configuration instructions. This information supplements the *Server for MVS* manuals. In many instances, you are advised to refer to the server manuals for specific details on the server environment in which this adapter operates.

Adapter for InfoMan

The Adapter for InfoMan enables you to access IBM's Information/Management databases using the industry standard Structured Query Language (SQL). You may access Information/Management data from the full array of server supported platforms, environments, and front-end tools.

You do not have to use the standard IBM Interactive System Productivity Facility (ISPF) interface to view retrieved data. You have the option of organizing reports using server front-end reporting facilities.

IBM's Information/Management

In this section:

IBM's Information/Management product provides a well-developed environment for tracking large system hardware and software problems, Program Temporary Fixes (PTFs), and change control. At many sites, the use of Information/Management has been expanded to include more general-purpose systems. Although Information/Management provides native query and reporting facilities, many users have suggested that Information/Management data should be available for querying and reporting from other IBM and third-party tools and utilities.

To provide this flexibility, iWay Software, in conjunction with IBM, has built an adapter for Information/Management, called the Adapter for InfoMan. This adapter allows any server-enabled application program to access an IBM Information/Management database by way of ANSI SQL.

The Adapter for InfoMan is one of many existing adapters that provide access to over 70 relational and non-relational data sources. The server can also perform joins with these heterogeneous data sources using one SQL statement.

Information/Management Access

By using the native Information/Management Application Program Interface (API), the Adapter for InfoMan provides the most efficient combination of server and Information/Management resources. SQL WHERE predicate search arguments are passed to Information/Management to exploit its fast indexed retrieval characteristics. Extended criteria searches are performed by the server to ensure full ANSI WHERE processing expected by most SQL front-ends.

Adapter for InfoMan Functional Overview

The Adapter for InfoMan interfaces to Information/Management through the Information/Management low-level API. This API allows application transactions such as creating, updating, retrieving, and deleting records. There is also a high-level API, which converts function calls to low-level API function calls.

Note: The high-level API is not used by the Adapter for InfoMan.

The Adapter for InfoMan uses the Programming Interface Communications Area (PICA), which is a control structure, to interface with the low-level API. The adapter inputs specific values for the application name or user ID (PICAUSRN), the privilege class name (PICACLSN), and the session member name (PICASESS).

The Adapter for InfoMan also provides trace information in allocations named FSTRACE and FSTRACE4. When tracing is enabled with FSTRACE, you can watch the Information/Management environment control transactions as the Adapter for InfoMan issues them to the Information/Management database. This trace information is useful for debugging purposes.

The Adapter for InfoMan interfaces seamlessly with the server. It allows many users access from client platforms, such as UNIX, and Windows. With the Adapter for InfoMan you benefit from having Information/Management data appear in your respective client applications.

How the Server Works With Information/Management

This topic describes how the server uses Information/Management S-words. An S-word, or structured word, is a keyword that identifies the contextual meaning of a collected item of data in Information/Management.

The server uses Information/Management PIDT views as table or file names, and maps Information/Management S-words to column or field defaults. You can assign more friendly column names to S-words during the AUTOIMAN definition process. Also, you can define virtual columns to provide functions and data types not present in the Information/Management database.

InfoMan Hardware and Software Requirements

Hardware Requirements

The following hardware specifications must be met to use the Adapter for InfoMan:

- An IBM or IBM-compatible mainframe, supporting MVS/ESA, Release 4.3 or higher.
- 98K of space, required by the specific modules that constitute the Adapter for InfoMan in *qualif*.EDALIB.LOAD.

Software Requirements

The following minimum software levels are required for the operation of the Adapter for InfoMan:

- Server for MVS, Release 3.1.6 or higher, plus applicable PTFs.
- IBM Information/Management, Version 6, Release 1 for MVS/ESA.
- Adapter for InfoMan.

Configuring the Adapter for InfoMan

In this section:

Server JCL for IBM Information/Management

How to:

Enable the Adapter for InfoMan

The Adapter for InfoMan is installed as part of the standard server installation procedure. Complete Steps 1, 2, and 3 after the server distribution tape has been unloaded. For information regarding the installation procedure, refer to the *iWay Server Installation* manual.

Server JCL for IBM Information/Management

The Adapter for InfoMan can be enabled in a server. The following is an example of server JCL containing the appropriate allocations for an IBM Information/Management database.

Syntax: How to Enable the Adapter for InfoMan

```
//JOBCARD
//*****************************************************
//*
//*  Server JCL
//*
//*  with InfoMan Data Adapter libraries.
//*
//*****************************************************
//EDASERVE  EXEC PGM=SSCTL,
//STEPLIB     DD  DISP=SHR,DSN=qualif.EDALIB.LOAD
   .
   .
   .
//            DD  DISP=SHR,DSN=INFOFVT.V6R1M0.BLX1.LOAD
//            DD  DISP=SHR,DSN=INFOFVT.V6R1M0.VSAMLOAD
//            DD  DISP=SHR,DSN=INFOFVT.V6R1M0.FIXLOAD
//            DD  DISP=SHR,DSN=INFOFVT.V6R1M0.SBLMMOD1
//            DD  DISP=SHR,DSN=INFOFVT.V6R1M0.SESSLOAD
   .
   .
   .
//MASTER      DD  DISP=SHR,DSN=qualif.MASTER.DATA
//ACCESS      DD  DISP=SHR,DSN=qualif.ACCESS.DATA
//*FSTRACE    DD  SYSOUT=*,DCB=(LRECL=132,BLKSIZE=132,RECFM=F)
//*FSTRACE4   DD  SYSOUT=*,DCB=(LRECL=132,BLKSIZE=132,RECFM=F)
```

where:

`STEPLIB`

Is the allocation for the server load library, *qualif*.EDALIB.LOAD. Allocate this library before any IBM Information/Management data sets or other system load libraries.

All libraries in STEPLIB must be APF-authorized in order for the server load modules to run.

If your installation uses non-APF authorized libraries, you must allocate only *qualif*.EDALIB.LOAD (which must be APF-authorized) in STEPLIB, and all non-APF authorized libraries under ddname EDALIB.

`qualif`

Is the high-level qualifier for the data set.

`MASTER`

Is the allocation for the *qualif*.MASTER.DATA data set, in which Master Files are found.

The AUTOIMAN facility will create Master Files to describe IBM Information/ Management data to the server.

`ACCESS`

Is the allocation for the *qualif*.ACCESS.DATA data set, in which Access Files are found.

The AUTOIMAN facility will create Access Files to describe IBM Information/Management data to the server.

`FSTRACE`

If uncommented, logs all trace output from the Adapter for InfoMan. FSTRACE creates an entry for each retrieved record, and is a useful method for logging all communication with the IBM Information/Management database.

`FSTRACE4`

If uncommented, logs only trace output summaries. FSTRACE4 displays Program Interface Argument Table (PIAT) criteria, and the number of hits to the IBM Information/Management database (that is, the number of records that satisfied the search criteria). It is effective in filtering out the additional output that FSTRACE provides.

Defining the Adapter for InfoMan User ID and Session ID

The Adapter for InfoMan interfaces with the low-level InfoMan Application API. Since the low-level API acquires application parameters from the Program Interface Communications Area (PICA), the Adapter for InfoMan inputs certain pre-defined parameters to PICA.

PICA Parameters

PICA parameters of special interest to the Information/Management database administrators include:

- **PICAUSRN:** Application ID or User ID
- **PICASESS:** Session Member Name
- **PICACLSN:** Privilege Class Name
- **PICADBID:** Database ID
- **PICATINT:** Transaction Time Interval

The Information/Management database uses the application ID PICAUSRN in conjunction with the privilege class name PICACLSN to establish the level of user accessibility permitted by the database.

If you specify a PICAUSRN (application ID) and/or PICACLSN (privilege class name) in an Access File, the Adapter for InfoMan uses it. If you do not, the Adapter for InfoMan inputs a PICAUSRN with a default eight-character value of SAMPID, and a PICACLSN with a default eight-character value of MASTER. During installation, these default values must be defined and recognized by the Information/Management database. The AUTOIMAN facility automatically creates the Access File with the specified PICAUSRN and PICACLSN values.

The PICASESS (session member name) is also defined in the Access File. If you do not specify a PICASESS value in the Access File, the default session name BLGSES00 is used. In order for your Information/Management database to recognize the default session name, you can define the default session BLGSES00 to your Information/Management database with full MASTER class privileges. The next topic explains how to set up the default user ID, class name, and session ID in the Information/Management database.

The PICATINT (transaction time interval) is the maximum time in seconds for a transaction to complete in the IBM API. Specify this parameter in the Access File.

```
TINT=x
```

where:

x

Is the transaction time interval in seconds. 0 is the default value.

Procedure: How to Set Up the Adapter for InfoMan User ID, Privilege Class Name, and Session ID

The following steps are required to set up the default Adapter for InfoMan user ID (or application ID), the default privilege class name, and the default session ID.

1. Define SAMPID as an eligible user ID accessible and available to the Information/ Management database. The Adapter for InfoMan will interface to the low-level API with:

   ```
   PICAUSRN='SAMPID'
   ```

   You may use the Information/Management ISPF panels to do this. SAMPID must also be a member of the MASTER class with full privileges.

2. Define session BLGSES00 as a session with full privileges using the Information/ Management ISPF panels. This session is a CSECT, and can be created, assembled, and link-edited by your Information/Management administrator. The session name BLGSES00 must be recognized and defined in the Information/Management database.

3. The session member BLGSES00 must be in a load library concatenation sequence available to Information/Management. Place the assembled and link-edited BLGSES00 member in STEPLIB DD of your server JCL.

Overriding the Default PICA Values in an Access File

You can override the default PICA values by specifying them in an Access File. The following table shows the PICA parameters, their Access File keywords, and their default values.

PICA Parameter	Access File Keyword	Default Value
PICAUSRN	USRN	'SAMPID'
PICASESS	SESS	'BLGSES00'
PICACLSN	CLSN	'MASTER'
PICADBID	DBID	5
PICATINT	TINT	0

To change a default in an Access File, edit the text following its keyword in the Access File. The following is an example of an Access File with some override values:

```
DBID=5,ITABLE=TS0032I,RTABLE=TS0032R,TINT=600,
SESS=MYSESS1,USRN=JOHN,CLSN=MASTER,$
```

In this example, DBID (database ID) and CLSN (privilege class name) specify the default values. SESS (session member name) has an override value that replaces BLGSES00, USRN (user ID) has an override value that replaces SAMPID, and the maximum transaction internal is set to 600 seconds.

Additional InfoMan Access File Parameters

You can include an IGNORE= parameter in the Access File to specify the processing action for database errors.

`IGNORE=ALL`	Allows all database errors to be processed as if they were not problematic. This is not recommended.
`IGNORE=NONE`	Allows no database errors to pass through. A message is issued, relevant data is logged in FSTRACE, and processing is halted.
`IGNORE=TRUNC`	Allows only field truncation errors to be passed through and the truncated data is returned. A message is logged in FSTRACE, and processing continues.

These errors indicate logical errors in the INFOMAN database. For more information, see the *IBM API* manual.

InfoMan Access Control

The following topics describe InfoMan access control:

- Server Security
- IBM Information/Management Database Security

In any computer system, it is important that data be secured from unauthorized access. Both InfoMan and the server provide security mechanisms to ensure that users have access to only those objects for which they have authorization.

Server Security in InfoMan

To activate server security, you must enable the EXTSEC parameter in the EDASERVE DD data set member as follows:

```
EXTSEC=ON
```

This setting will default to the security available from your system environment. For more information on server security, see the *iWay Server Installation* manual.

IBM Information/Management Database Security

At the IBM Information/Management database level, security is set by defining a privileged class and assigning users to it. A system administrator establishes the display, create, delete, or copy function and the data record type. For more information on how to define user IDs to a privileged class, see your System Administrator or IBM's *The Information/Management User's Guide for MVS/ESA, Version 6, Release 1*.

AUTOIMAN Configuration File

In this section:

The AUTOIMAN configuration file, IMANCONF, contains information relevant to the PIDT Selection Library, the Master File, and the Access File. Customize this file according to the directions that follow.

Note: If ddname FOCEXEC is allocated to concatenated partitioned data sets, IMANCONF must reside in the first partitioned data set.

Editing IMANCONF

The IMANCONF configuration file consists of three lines:

```
1. INFOFVT.V6R1M0.SBLMFMT
2. qualif.MASTER.DATA
3. qualif.ACCESS.DATA
```

Line 1 is the PIDT Selection Library. INFOFVT.V6R1M0.SBLMFMT is the default provided with Information/Management. You may keep this library as the default, or enter a different PIDT Selection Library name.

Lines 2 and 3 are the partitioned data sets (PDSs) for the Master and Access Files. In both lines, substitute your user ID for *qualif*. The PDS names can remain as shown, or be changed. Remember that only cataloged PDSs can be referenced.

Note: Only the first 35 characters of each of the lines will be read

Maintaining Multiple Configuration Files in InfoMan

In order to maintain multiple IMANCONF configuration files, you must copy the IMANCONF configuration file into your own IMANCONF partitioned data set. Follow the editing instructions in *Editing IMANCONF* on page 15-11 to customize the file.

Other InfoMan Editing Options

The three lines of data that comprise the IMANCONF configuration file are the permanent AUTOIMAN defaults, and also appear on the AUTOIMAN screen, Master File Generation Facility for InfoMan Main Menu. You can change the PIDT Selection Library and the Master and Access Files on the Main Menu. However, the changes you make on the Main Menu are effective only for the duration of the AUTOIMAN session, and do not become permanent AUTOIMAN defaults.

Describing InfoMan Data Sources

The following topics describe InfoMan data sources:

- Executing AUTOIMAN
- Working With AUTOIMAN
- Master File Generation Facility in InfoMan

To access an existing table or view using the server, you must first describe it in a Master File and an Access File. The AUTOIMAN facility enables the creation and cataloging of Master Files and Access Files automatically. AUTOIMAN prompts you for Retrieval and Inquiry Information/Management Program Interface Data Table (PIDT) names, which are used to create the Master and Access Files.

The AUTOIMAN facility gives you the ability to do the following:

- Select fields to be incorporated into a Master File.
- Change the usage format, which determines the way data displays in a report, and how it is manipulated when making calculations for a report. This is useful for date fields and character fields that hold numeric data. For example, a priority field that has a usage format of A2 may be changed in AUTOIMAN to any integer format for summing or averaging purposes.
- Change the default field name to a unique and more meaningful field name. By default, AUTOIMAN suffixes a number (preceded by an underscore) to the Information/ Management P-word to make it unique. AUTOIMAN makes it possible for you to overwrite this field name with a more meaningful name.

  Note: A P-word (prefix word) is a keyword used in a search argument that identifies which field in the database corresponds with the data being searched.

Executing AUTOIMAN

To execute AUTOIMAN, you must:

1. Run the EDAAUTO list *qualif*.EDACTL.DATA(EDAAUTO). The main menu for execution of the auto procedures opens.

```
Instructions: Place cursor on the type of File Description you wish
to translate and then press ENTER.  Use PF3 to Quit.

                            DB2

                            IDMS

                            ADABAS

                            DATACOM

                            INFOMAN

                            Cobol FD

                            NOMAD
```

2. Choose INFOMAN from the main menu for the execution of the auto procedures.

Working With AUTOIMAN

In this section:

Movement Within Screens in InfoMan

Help in InfoMan

On a color monitor, AUTOIMAN screens display the following colors:

Screen element	Color
Input fields	White
Fixed fields	Light green
F or PF key instructions	Pale yellow
Screen headings	Pale yellow
Messages within a screen	Light green
Messages between screens	Red

Movement Within Screens in InfoMan

Movement on one screen is controlled by the tab key, the four directional arrow keys, and the programmed function (F or PF) keys. Use the *Enter* key to go from one screen to another.

When you press *F12* on any screen, the previous screen displays. Press *F3* if you want to exit AUTOIMAN.

Note: In this manual, programmed function keys are referred to as F keys.

Help in InfoMan

Numerous Help screens are available in AUTOIMAN. To invoke Help, move the cursor to a field that you want information on and press *F1*. To exit a help screen and return to your previous location, press *Enter*.

Note: On the Main Menu, you can obtain Help only on an input field. On all other screens in AUTOIMAN, you can invoke help from anywhere on the screen.

Master File Generation Facility in InfoMan

When you select InfoMan on the EDAAUTO selection screen, the Master File Generation Facility for InfoMan Main Menu displays. You specify environment parameters on this menu.

```
AUTOIMAN        Master File Generation Facility for InfoMan
                                  Main Menu

Retrieval PIDT               ==========>      (Leaving PIDT blank will
Inquiry   PIDT               ==========>       display selection list)

Master/Access File Name      ==========>            (Optional)
Replace existing MFD,Access File ?   ==>  N          (Y/N)

DBID                         ==========>  5
Session Member               ==========>  BLGSES00
Application ID               ==========>  SAMPID
Class Name                   ==========>  MASTER
PIDT Selection Library       ==========>  INFOFVT.V6R1M0.SBLMFMT
Master File Destination      ==========>  qualif.MASTER.DATA
Access File Destination      ==========>  qualif.ACCESS.DATA

F1=Help   F3=Exit
```

On this menu you can obtain field-specific help by pressing *F1*. To exit AUTOIMAN, press *F3*.

If you press Enter on this menu without having entered a valid Retrieval PIDT and Inquiry PIDT, the PIDT Selection Panel is displayed, to assist you in selecting PIDT names. When you have completed entry of PIDTs and other necessary information on this menu and pressed Enter, the Field Selection for Retrieval PIDT Name screen appears.

The following table provides field descriptions for the AUTOIMAN Main Menu.

Field	**Description**
Retrieval PIDT	You must enter the Retrieval PIDT. If you do not, a selection list, from which you may select a Retrieval PIDT, displays after you press *Enter*. If you enter a Retrieval PIDT, it must exist in the PIDT Selection Library listed on the Main Menu. Enter the correct library if the default library name generated by AUTOIMAN is inappropriate. The next field, the Inquiry PIDT, follows the same rules. Additionally, you must enter both PIDTs to continue, or the PIDT Selection Panel displays.
Inquiry PIDT	You must enter the Inquiry PIDT. If you do not, a selection list, from which you may select an Inquiry PIDT, displays after you press *Enter*. If you enter an Inquiry PIDT, it must exist in the PIDT Selection Library listed on the Main Menu. Enter the correct library if the default library name generated by AUTOIMAN is inappropriate. The preceding field, the Retrieval PIDT, follows the same rules. Additionally, you must enter both PIDTs to continue, or the PIDT Selection Panel appears.
Master/Access File Name	You can specify a name for the Master and Access File or leave this field blank. If blank, the name will default to the specified Retrieval PIDT name.
Replace Existing MFD, Access File?	This message displays if you select a Retrieval PIDT for which a Master File and Access File already exist. You can enter Y or N (the default). If you enter Y, the new Master File and Access File will replace the existing file.

Field	Description
DBID	This one-character database identifier field is required. It is used as input to the PICADBID field of the PICA block when the adapter is communicating to the Information/ Management database. 5 is the default value, which indicates a read access type to the Information/Management database. If you want to access another Information/ Management type database with other identifiers, you can do so by specifying its DBIDs in this field.
Session Member	This is a seven- to eight-character session member name used as input to the PICASESS field of the PICA block when the adapter is communicating to the Information/ Management database. Specify the session member appropriate for your retrieval PIDT. This session member must be assembled, linked, and defined to your Information/ Management database.
Application ID	This is a one- to eight-character name that identifies the user application to the Information/Management database. It will be used as input to the PICAUSRN field of the PICA block when the adapter communicates to the Information/ Management database. Specify the application ID appropriate for your retrieval PIDT. This application ID must be valid in your Information/Management database.
Class Name	This is a one- to eight-character privilege class name used by the application when executing a transaction. It is used as input to the PICACLSN field of the PICA block when the adapter communicates to the Information/Management database. Specify the appropriate privilege class for your application ID. This privilege class name must be valid in your Information/Management database.
PIDT Selection Library	This library is the partitioned data set containing all Retrieval and Inquiry PIDTs. AUTOIMAN can access only PIDTs that are in the library specified here.
Master File Destination	This is the partitioned data set that stores the new Master File. The partitioned data set must be cataloged.
Access File Destination	This is the partitioned data set that stores the new Access File. The partitioned data set must be cataloged.

PIDT Selection Panel in InfoMan

In this section:

If you do not specify a valid Retrieval PIDT and Inquiry PIDT on the Main Menu, the PIDT Selection Panel appears:

```
AUTOIMAN                    PIDT Selection Panel                Page  1

_ BLGOZ11  _ BLMPMHF  _ BLMPRRU  _ BLMZZ41  _ BTNRPRCR _ BTNRZZ3A
_ BLGOZ14  _ BLMPMHS  _ BLMPRSC  _ BLMZZ42  _ BTNRPRPR _ BTNRZZ3B
_ BLGPRAL  _ BLMPMSC  _ BLMPRSF  _ BLMZZ43  _ BTNRPR10 _ BTNRZZ31
_ BLGPRBK  _ BLMPMSF  _ BLMPRSPS _ BLMZZ44  _ BTNRPR20 _ TS0AF8C
_ BLGPRGR  _ BLMPRAR  _ BLMPRSR  _ BLMZZ45  _ BTNRPR30 _ TS0AF8CP
_ BLGPROZ  _ BLMPRCD  _ BLMPRSY  _ BLMZZ46  _ BTNRPR40 _ TS0AF8I
_ BLGPRUS  _ BLMPRCH  _ BLMZZ12  _ BTNCHSLR _ BTNRPR50 _ TS0AF8IP
_ BLGRPR10 _ BLMPRCPS _ BLMZZ13  _ BTNPRCH  _ BTNRPR60 _ TS0AF8R
_ BLGRPR20 _ BLMPRCR  _ BLMZZ14  _ BTNPRSC  _ BTNRPR70 _ TS0AF8RP
_ BLGRPR30 _ BLMPRCS  _ BLMZZ21  _ BTNPRSLR _ BTNRZZ10 _ TS0AF8U
_ BLGR8FLO _ BLMPRFH  _ BLMZZ22  _ BTNRCH10 _ BTNRZZ14 _ TS0AF8UP
_ BLGR8FOU _ BLMPRFS  _ BLMZZ23  _ BTNRCH20 _ BTNRZZ16 _ TS0B0CC
_ BLGR8PLA _ BLMPRHC  _ BLMZZ31  _ BTNRCH30 _ BTNRZZ2A _ TS0B0CCP
_ BLGR8POL _ BLMPRHF  _ BLMZZ32  _ BTNRCH40 _ BTNRZZ2B _ TS0B0CI
_ BLGR8STE _ BLMPRHS  _ BLMZZ33  _ BTNRCH45 _ BTNRZZ2C _ TS0B0CIP
_ BLGR8SUR _ BLMPRPC  _ BLMZZ34  _ BTNRCH50 _ BTNRZZ21 _ TS0B0CR
_ BLGZZ11  _ BLMPRPPS _ BLMZZ35  _ BTNRGROA _ BTNRZZ24 _ TS0B0CRP
_ BLMPMHC  _ BLMPRPR  _ BLMZZ36  _ BTNRPERS _ BTNRZZ25 _ TS0B0CU

Enter R (Retrieval) and I (Inquiry) F1=Help  F3=Exit  F5=Rfind
F6=Search  F7=Backward  F8=Forward  F12=Cancel
```

This panel lists all of the PIDTs found in the PIDT Selection Library shown on the Main Menu.

Help is accessed with *F1*. You can exit AUTOIMAN with *F3*. You can return to the previous screen, the Main Menu, using *F12*. To page forward, press *F8*. To page backward, press *F7*. To locate PIDTs, *F6* and *F5* are available. They are explained in detail in *Using the F6 Search and F5 Rfind Keys on the PIDT Selection Panel* on page 15-18.

The screen's page number is in the top right corner.

Entering Data on the PIDT Selection Panel

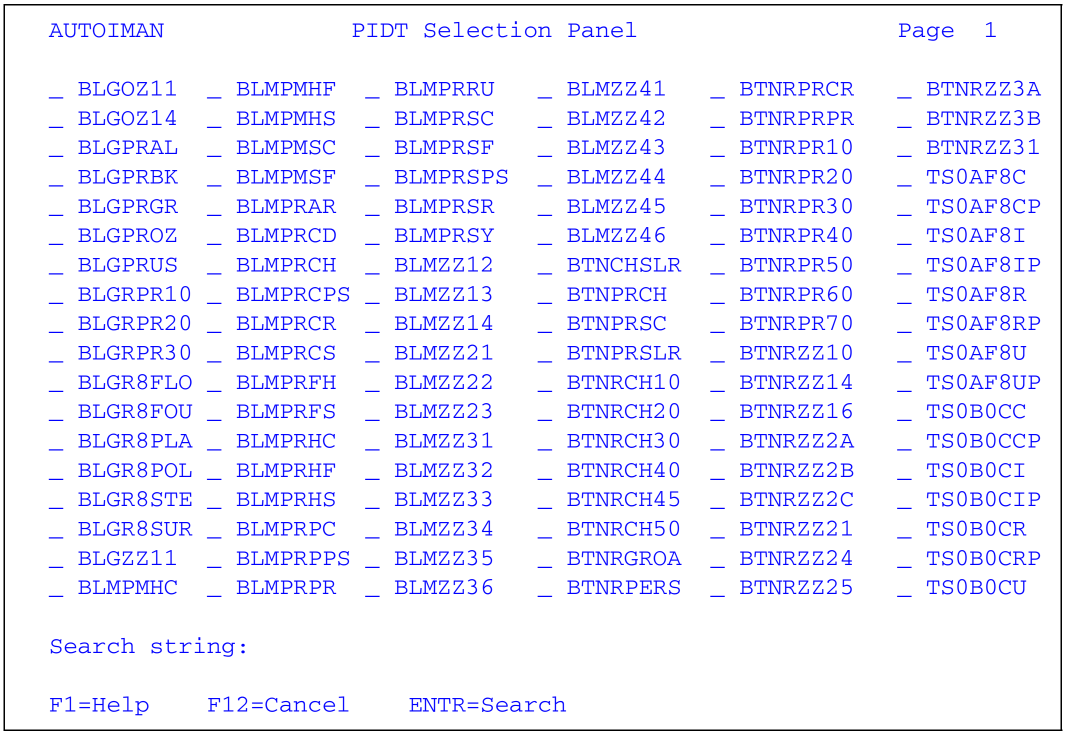
The PIDT Selection Panel displays Retrieval and Inquiry PIDT names. You can specify a Retrieval PIDT by placing an R before its name, and an Inquiry PIDT by placing an I before its name (the R and I are not case-sensitive).

Note that if you enter either a Retrieval or Inquiry PIDT on the Main Menu, the PIDT name displays on the PIDT Selection Panel with either an R or an I before it. You may use this choice or select another.

Press *Enter* to continue.

Using the F6 Search and F5 Rfind Keys on the PIDT Selection Panel

When you select Search (*F6*), the library names remain on screen and a field appears at the bottom of the screen where you can enter a search string.



```
AUTOIMAN                PIDT Selection Panel                     Page   1

_ BLGOZ11   _ BLMPMHF   _ BLMPRRU   _ BLMZZ41   _ BTNRPRCR   _ BTNRZZ3A
_ BLGOZ14   _ BLMPMHS   _ BLMPRSC   _ BLMZZ42   _ BTNRPRPR   _ BTNRZZ3B
_ BLGPRAL   _ BLMPMSC   _ BLMPRSF   _ BLMZZ43   _ BTNRPR10   _ BTNRZZ31
_ BLGPRBK   _ BLMPMSF   _ BLMPRSPS  _ BLMZZ44   _ BTNRPR20   _ TS0AF8C
_ BLGPRGR   _ BLMPRAR   _ BLMPRSR   _ BLMZZ45   _ BTNRPR30   _ TS0AF8CP
_ BLGPROZ   _ BLMPRCD   _ BLMPRSY   _ BLMZZ46   _ BTNRPR40   _ TS0AF8I
_ BLGPRUS   _ BLMPRCH   _ BLMZZ12   _ BTNCHSLR  _ BTNRPR50   _ TS0AF8IP
_ BLGRPR10  _ BLMPRCPS  _ BLMZZ13   _ BTNPRCH   _ BTNRPR60   _ TS0AF8R
_ BLGRPR20  _ BLMPRCR   _ BLMZZ14   _ BTNPRSC   _ BTNRPR70   _ TS0AF8RP
_ BLGRPR30  _ BLMPRCS   _ BLMZZ21   _ BTNPRSLR  _ BTNRZZ10   _ TS0AF8U
_ BLGR8FLO  _ BLMPRFH   _ BLMZZ22   _ BTNRCH10  _ BTNRZZ14   _ TS0AF8UP
_ BLGR8FOU  _ BLMPRFS   _ BLMZZ23   _ BTNRCH20  _ BTNRZZ16   _ TS0B0CC
_ BLGR8PLA  _ BLMPRHC   _ BLMZZ31   _ BTNRCH30  _ BTNRZZ2A   _ TS0B0CCP
_ BLGR8POL  _ BLMPRHF   _ BLMZZ32   _ BTNRCH40  _ BTNRZZ2B   _ TS0B0CI
_ BLGR8STE  _ BLMPRHS   _ BLMZZ33   _ BTNRCH45  _ BTNRZZ2C   _ TS0B0CIP
_ BLGR8SUR  _ BLMPRPC   _ BLMZZ34   _ BTNRCH50  _ BTNRZZ21   _ TS0B0CR
_ BLGZZ11   _ BLMPRPPS  _ BLMZZ35   _ BTNRGROA  _ BTNRZZ24   _ TS0B0CRP
_ BLMPMHC   _ BLMPRPR   _ BLMZZ36   _ BTNRPERS  _ BTNRZZ25   _ TS0B0CU

Search string:

F1=Help    F12=Cancel    ENTR=Search
```

You can enter up to eight characters in the search string. It can contain all or any part of a PIDT name. However, you cannot include blanks. After inputting the string, press *Enter* to begin the search. AUTOIMAN searches the PIDT list for the first occurrence of the string.

If the search is not successful, the message *Search string not found* displays on the lower portion of the screen. To continue using the AUTOIMAN search facility, check that you entered the string correctly, or enter another string.

If the search is successful, the first PIDT containing the search string displays in the top left corner of the screen, and the bottom of the screen changes back to its appearance on the first PIDT Selection Panel. You can then use the *F5* (Rfind) key.

You can specify Retrieval and Inquiry PIDTs on this screen, as described in *Entering Data on the PIDT Selection Panel* on page 15-18.

The AUTOIMAN Rfind (*F5*) feature is an extension of search. After a search, each time you press *F5*, the next PIDT that contains the search string appears in the top left corner of the page.

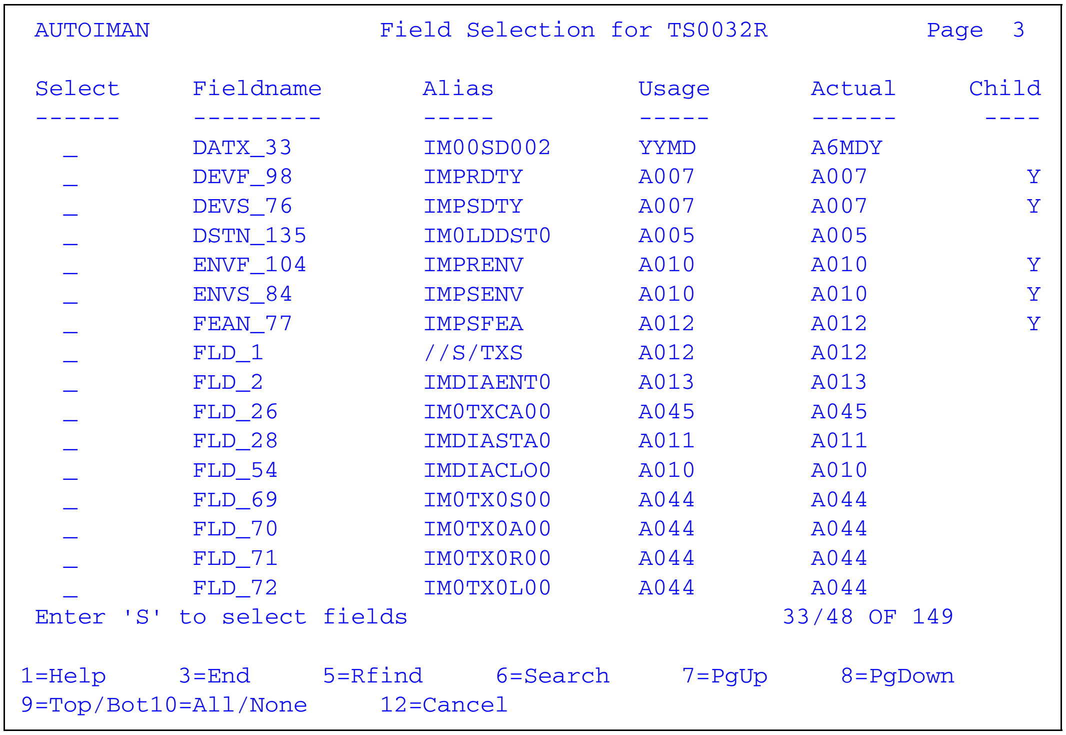
Rfind works independently of the actual screen image. It searches from the location of the last matching PIDT, and forwards the screen, as requested, until it has found all occurrences of the string. The message *Search string not found* displays if additional occurrences cannot be found.

If you need to see the name of the string you are *refinding*, press *F6* for Search. The search string input field displays the most recently searched string. To continue *refinding,* press *F12* to cancel the search screen, and then press *F5* for Rfind.

Field Selection for Retrieval PIDT Name in InfoMan

The Field Selection screen displays field data read from the selected Retrieval PIDT. The Retrieval PIDT name is shown in the screen title.

On this screen you select the fields to include in the new Master File. You can edit the Fieldname and Usage fields.



```
 AUTOIMAN                 Field Selection for TS0032R           Page  3

 Select     Fieldname        Alias          Usage        Actual      Child
 ------     ---------        -----          -----        ------       ----
   _        DATX_33          IM00SD002      YYMD         A6MDY
   _        DEVF_98          IMPRDTY        A007         A007            Y
   _        DEVS_76          IMPSDTY        A007         A007            Y
   _        DSTN_135         IM0LDDST0      A005         A005
   _        ENVF_104         IMPRENV        A010         A010            Y
   _        ENVS_84          IMPSENV        A010         A010            Y
   _        FEAN_77          IMPSFEA        A012         A012            Y
   _        FLD_1            //S/TXS        A012         A012
   _        FLD_2            IMDIAENT0      A013         A013
   _        FLD_26           IM0TXCA00      A045         A045
   _        FLD_28           IMDIASTA0      A011         A011
   _        FLD_54           IMDIACLO0      A010         A010
   _        FLD_69           IM0TX0S00      A044         A044
   _        FLD_70           IM0TX0A00      A044         A044
   _        FLD_71           IM0TX0R00      A044         A044
   _        FLD_72           IM0TX0L00      A044         A044
 Enter 'S' to select fields                            33/48 OF 149

1=Help     3=End     5=Rfind     6=Search     7=PgUp     8=PgDown
9=Top/Bot10=All/None     12=Cancel
```

F1 is Help. *F12* is Cancel, which takes you to the previous screen. *F3* ends your AUTOIMAN session. *F8* (PgDown) scrolls down the list, and *F7* (PgUp) scrolls up. *F9* is a toggle key and jumps either to the beginning or the end of the list. *F10*, also a toggle key, selects either all of the fields or none.

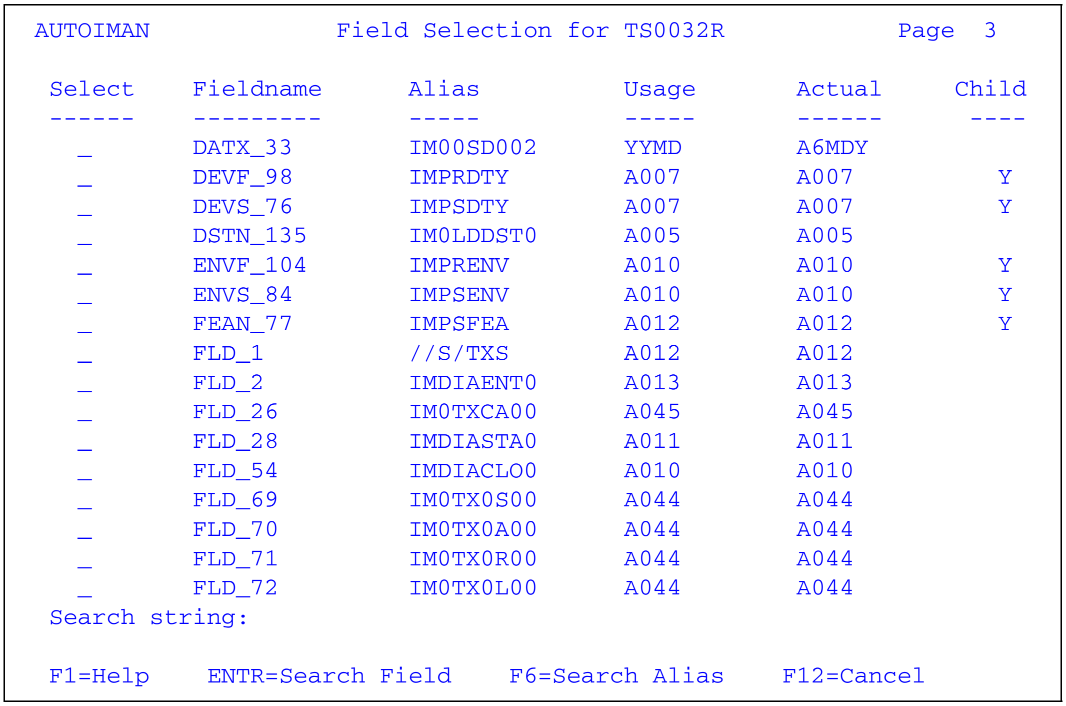
The screen's page number displays in the top right corner. In the lower right corner, the number of fields on the screen, and their relationship to the total number of fields in the file, are shown. For example, 1/16 OF 149 means that the file contains 149 fields, and that fields 1 through 16 appear on the screen.

Using the F6 Search and F5 Rfind Keys on the Field Selection Screen in InfoMan

F6 Search and *F5* Rfind on the Field Selection screen can help you locate field names or aliases. Their usage is similar to the PIDT Selection Panel Search and Rfind facilities.

F6 Search in InfoMan

When you select *F6* for Search, the field names remain on screen, but at the bottom of the screen a field displays, where you can enter a search string.



```
 AUTOIMAN                    Field Selection for TS0032R            Page  3

  Select     Fieldname      Alias         Usage       Actual      Child
  ------     ---------      -----         -----       ------       ----
    _        DATX_33        IM00SD002     YYMD        A6MDY
    _        DEVF_98        IMPRDTY       A007        A007            Y
    _        DEVS_76        IMPSDTY       A007        A007            Y
    _        DSTN_135       IM0LDDST0     A005        A005
    _        ENVF_104       IMPRENV       A010        A010            Y
    _        ENVS_84        IMPSENV       A010        A010            Y
    _        FEAN_77        IMPSFEA       A012        A012            Y
    _        FLD_1          //S/TXS       A012        A012
    _        FLD_2          IMDIAENT0     A013        A013
    _        FLD_26         IM0TXCA00     A045        A045
    _        FLD_28         IMDIASTA0     A011        A011
    _        FLD_54         IMDIACLO0     A010        A010
    _        FLD_69         IM0TX0S00     A044        A044
    _        FLD_70         IM0TX0A00     A044        A044
    _        FLD_71         IM0TX0R00     A044        A044
    _        FLD_72         IM0TX0L00     A044        A044
  Search string:

  F1=Help     ENTR=Search Field     F6=Search Alias     F12=Cancel
```

You can enter up to nine characters in the search string. It can contain all or any part of a field name or an alias. However, you cannot include blanks. After you specify the string, to begin the search, press *Enter* to search for a file name, or press *F6* to search for an alias. AUTOIMAN searches the list of field names or aliases for the first occurrence of the string.

If the search is not successful, the message *Search string not found* displays on the lower portion of the screen. To continue using the AUTOIMAN search facility, check that you entered the string correctly, or enter another string.

If the search is successful, the first field name or alias containing the search string displays in the top left corner of the screen, and the bottom of the screen changes back to its appearance on the first Field Selection for PIDT Name screen. You can then use the *F5* (Rfind) key.

F5 Rfind in InfoMan

The AUTOIMAN Rfind feature is an extension of search. After a search, each time you press *F5*, the next PIDT that contains the search string appears in the top left corner of the page.

Rfind works independently of the actual screen image. It searches from the location of the last matching PIDT, and forwards the screen, as requested, until it has found all occurrences of the string. The message *Search string not found* displays if additional occurrences cannot be found.

If you need to see the name of the string you are *refinding*, press *F6* for Search, and the search string input field displays the most recently searched string. To continue *refinding*, press *F12* to Cancel the search screen, and then press *F5* for Rfind.

Making Changes in Field Descriptions in InfoMan

You can change the Fieldname and the Usage format on the Field Selection screen, but you cannot change the Alias, Actual, and Child columns.

Field	Description
Fieldname	You may enter a new Fieldname of up to 15 characters. Duplicate field names should not be used. A duplicate field name will not cause an error in AUTOIMAN, but could cause unpredictable results when working with the newly created Master File.
Usage	You may enter a new Usage field of up to six characters. For non-date fields, it may be any alphanumeric or integer format.

Selection in InfoMan

On the Field Selection panel, enter an S in the Select column to select a field. *F10* is a toggle key, and can be used to select all of the fields, or none.

You can select only field names with aliases. If you select a field name that does not have an alias, a message appears.

Note: You must select at least one non-child field in order to continue working in AUTOIMAN.

Press *Enter* to continue.

Retrieval PIDT Name Confirmation in InfoMan

In this section:

Changing Your Selections in InfoMan

Confirming Your Choices in InfoMan

The confirmation screen displays the choices you have made during your AUTOIMAN session and permits you to either confirm them or change them. AUTOIMAN displays the Retrieval PIDT name in the screen heading.

```
AUTOIMAN                    TS0032R Confirmation

Total fields selected :    8
Child fields selected :    2

Master File   :   AAA
   Location   :   qualif.MASTER.DATA

Access File   :   AAA
   Location   :   EDABXV.ACCESS.DATA
   DBID               =   5               Session Member   =   BLGSES00
   Inquiry PIDT       =   TS0032I         Application ID   =   SAMPID
   Retrieval PIDT     =   TS0032R         Class Name       =   MASTER

ENTR=Confirm     PF12=Cancel
```

Only the *F12* key (Cancel) is active, which takes you to the previous screen. To confirm your selections, press *Enter*.

Changing Your Selections in InfoMan

The following table provides descriptions of the fields on the Retrieval PIDT Name Confirmation panel.

Field	Description
Total Fields Selected	If you want to change the total number of fields you have selected, press *F12* to return to the Field Selection screen, and reselect your choices.
Child Fields Selected	Of the total number of fields selected, this field displays the number of child fields. On the Field Selection for Retrieval PIDT Name screen, child fields are identified with a Y in the Child column. In the newly created Master File, each child field is placed in its own segment.
Master File Name and Location	To change the Master File name and its location, use the *F12* key to return to the AUTOIMAN Main Menu.
Access File Name and Location	To change the Access File name and its location, use the *F12* key to return to the AUTOIMAN Main Menu.
DBID, Session Member, Application ID, Class Name	To change the DBID, Session Member, Application ID, and Class Name, use the *F12* key to return to the AUTOIMAN Main Menu.

Confirming Your Choices in InfoMan

To confirm your choices, press *Enter*. The message

```
Generating Master/Access File Description
```

displays, and AUTOIMAN returns to the Main Menu.

CHAPTER 16

Using the Adapter for Informix

Topics:

The Adapter for Informix allows applications to access Informix data sources. The adapter converts application requests into native Informix statements and returns optimized answer sets to the requesting application. This adapter has read/write capabilities, which enables the adapter to insert the data from an application to the data source.

Preparing the Informix Environment

In this section:

The Adapter for Informix minimally requires the installation of the Informix Client SDK software. The Informix Client SDK software allows you to connect to a local or remote Informix database server.

In order to support a greater range of Informix DBMS releases, the current server release uses Informix Client SDK in its Adapter for Informix. All the current Informix Client SDK's support connectivity to wide range of Informix databases. For example, SDK 2.6 will support connectivity to Informix releases 5.1 through 9.3.

Note: Always check the Informix SDK documentation for supported releases of Informix databases.

The following environment settings apply when configuring the server with the Adapter for Informix.

Procedure: How to Set Up the Environment on Windows

On Windows, the Informix environment is set up during the installation of the Informix Server and Informix Client SDK.

Procedure: How to Set Up the Environment on UNIX

1. Identify the Informix Client SDK using the UNIX environment variable $INFORMIXDIR. For example, to set the home directory for the Informix software to /usr/informix930, specify:

   ```
   INFORMIXDIR=/usr/informix930
   export INFORMIXDIR
   ```

2. Identify the Default Informix Server using the UNIX environment variable $INFORMIXSERVER:

   ```
   INFORMIXSERVER=server_name
   export INFORMIXSERVER
   ```

3. Specify the path to the Informix shared library using the UNIX environment variable $LD_LIBRARY_PATH. For example:

   ```
   LD_LIBRARY_PATH=$INFORMIXDIR/lib:$LD_LIBRARY_PATH
   export LD_LIBRARY_PATH
   ```

4. Specify the format of the DATE fields using the UNIX environment variable $DBDATE. For example, if the user wants the DATE to be returned, in the following format 01/08/2001:

   ```
   DBDATE=MDY4/
   export DBDATE
   ```

Accessing a Remote Informix Server

Using the standard rules for deploying the Informix Client, the server supports connections to:

- Local Informix database servers.
- Remote Informix database servers. To connect to a remote Informix database server, the sqlhosts file on the source machine must contain an entry pointing to the target machine and the listening process must be running on the target machine.

Informix SQL ID (for Informix SE only)

When you access Informix SE, you are identified by your UNIX login ID. The UNIX ID becomes the owner ID for any Informix objects (such as tables or indexes) created with immediate SQL commands. Immediate commands are non-parameterized SQL statements that are passed directly to the data source. The UNIX ID is also used as the sole authorization ID for Informix GRANT and REVOKE statements.

Other types of requests, such as SQL SELECTs, sponsor an Informix search for the necessary privileges to act on the particular tables and views in a data source using the UNIX ID. All Informix security rules are respected.

The following table shows the UNIX privileges required in order to specify certain SQL statements.

Statement	Requires
SELECT	Read permission for the Informix data and index files. UNIX write and execute (search) privileges for the data source directory.
UPDATE	Write permission for the Informix data and index files. UNIX write and execute (search) privileges for the data source directory.

ANSI-Compliant Data Sources

If you create a data source to be ANSI-compliant, owner naming is enforced by Informix. Therefore, to preserve any lowercase owner name, you must enclose it in double quotation marks. Otherwise, Informix will default to uppercase for the owner name. For example:

```
ENGINE SQLINF
SELECT * FROM "user1".test
END
```

For additional information, see the *Informix Guide to SQL*.

XA Support

Read/write applications accessing Informix data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see Appendix A, *XA Support*.

Configuring the Adapter for Informix

In this section:

Declaring Connection Attributes

Overriding the Default Connection

Controlling the Connection Scope

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

Example:

Declaring Connection Attributes

In order to connect to an Informix database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Informix database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Informix Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	**Description**
Connection name	Logical name used to identify this particular set of connection attributes.
Datasource	Informix server name, as defined by the dbservername field in the sqlhosts file.
Security	There are three methods by which a user can be authenticated when connecting to Informix: **Explicit.** The user ID and password are explicitly specified for each connection and passed to Informix, at connection time, for authentication. **Password passthru.** The user ID and password received from the application are passed to Informix, at connection time, for authentication. This option requires that the server be started with security off. **Trusted.** The adapter connects to Informix as a Windows login using the credentials of the Windows user impersonated by the server data access agent.
User	Name by which the user is known to Informix.
Password	Password associated with the user name.
Database name	Informix database name. For Informix SE, the entire path to the location of the database files including the database name must be specified.

Attribute	**Description**
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Informix, at connection time, for authentication.

```
ENGINE [SQLINF] SET CONNECTION_ATTRIBUTES [connection]
 [datasource]/userid,password ;dbname
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Informix, at connection time, for authentication. This option requires that the server be started with security off.

```
ENGINE [SQLINF] SET CONNECTION_ATTRIBUTES [connection] [datasource]/;dbname
```

Trusted authentication. The adapter connects to Informix as a Windows login using the credentials of the Windows user impersonated by the server data access agent.

```
ENGINE [SQLINF] SET CONNECTION_ATTRIBUTES [connection] [datasource]/,;dbname
```

where:

`SQLINF`

Indicates the Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is a logical name (or a data source name) used to identify this particular set of attributes.

If you plan to have only one connection to Informix, this parameter is optional. If not specified, the local database server serves as the default connection.

`datasource`

Is the name of the database server as defined by the dbservername field in the sqlhosts file.

If datasource is not specified, the Informix default server (value of the INFORMIXSERVER environment variable) is used. You can also specify the default server with two consecutive single quotation marks.

userid

Is the primary authorization ID by which you are known to Informix.

password

Is the password associated with the primary authorization ID.

dbname

Also referred to as schema, is the name of the Informix database used for this connection. The database name, including path, must be enclosed in single quotation marks.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allows the application to access the Informix database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Informix database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

The following SET CONNECTION_ATTRIBUTES command connects to a local Informix database server using operating system authentication:

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES /,
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLINF] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLINF`

Indicates the Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Informix database server named SAMPLENAME as the default Informix database server:

```
ENGINE SQLINF SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLINF] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLINF`

Indicates the Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. This value is the default.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Informix Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Informix data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Informix table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

   Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

   - **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.

   - **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.

5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.

6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note

that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave prefix and suffix fields blank.

7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value. See *Controlling the Mapping of Large Character Data Types* for more information.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLINF [AT connection]
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLINF

Indicates the Data Adapter for Informix.

AT *connection*

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR "edaqa".nf29004 DBMS SQLINF AT conn_inf
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLINF ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4,     I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=qaeda:"edaqa".nf29004,
CONNECTION=conn_inf,KEYS=1,$
```

Reference: Access File Keywords

Attribute	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the Informix table name. The table name can be fully qualified as follows: `TABLENAME=[[`*`database`*`.]`*`owner`*`.]`*`table`*
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local Informix database server. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following table lists how the server maps Informix data types. Note that you can:

- Control the mapping of large character data types.
- Control the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Informix Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
CHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 32767.
CHAR VARYING (*m*,*r*)	A(*m*,*r*)	A(*m*,*r*)	*m* is an integer between 1 and 255 (254 with index). *r* is an integer between 0 and 255, and less than m. 0 is the default value.
LVARCHAR	A2048	A2048	
NCHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 32767.
NVARCHAR	A255	A255	
DECIMAL (*p*,*s*)	P33.33	P(*p*,*s*)	*p* is an integer between 1 and 32. 16 is the default value for p. *s* is an integer between 0 and p. 0 is the default value for s.
FLOAT (*n*)	D20.2	D8	*n* is an integer between 1 and 14.
DOUBLE PRECISION (*n*)	D20.2	D8	*n* is an integer between 1 and 14.
INT	I11	I4	
INT8	P20	P10	
MONEY(*p*,*s*)	P18.2	P9	*p* is an integer between 1 and 32. 16 is the default value for p. *s* is an integer between 0 and p. 2 is the default value for s.

Informix Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
SERIAL	I11	I4	
SERIAL8	P20	P10	
SMALLINT	I6	I4	
BYTE	BLOB	BLOB	
TEXT	A32767	A32767	
BLOB			Not supported for the current release of the server.
CLOB			Not supported for the current release of the server.
DATE	YYMD	DATE	
DATETIME	HYYMD	HYYMD	
INTERVAL YEAR (*year_precision*) TO MONTH	A20	A20	
INTERVAL DAY (*day_precision*) TO SECOND			
(*fractional_seconds_precision*)	A20	A20	
BOOLEAN	A1	A1	The only values that can be inserted are T or F.

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Informix data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Informix Data Type	**Remarks**	**LONGCHAR ALPHA or BLOB**		**LONGCHAR TEXT**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
CHAR (*n*)	*n* is an integer between 1 and 32767	A*n*	A*n*	TX50	TX
NCHAR (*n*)	*n* is an integer between 1 and 32767	A*n*	A*n*	TX50	TX
LVARCHAR		A2048	A2048	TX50	TX
TEXT		A32767	A32767	TX50	TX

Syntax: How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLINF] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

`SQLINF`

Indicates the Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

`ALPHA`

Maps the Informix data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). ALPHA is the default value.

`TEXT`

Maps the Informix data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

`BLOB`

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Informix data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

For OS/390 and z/OS, maps the Informix data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Informix data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Informix Data Type	**Remarks**	**VARCHAR ON**		**VARCHAR OFF**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
LVARCHAR (*n*)		A2048V	A2048V	A2048	A2048
NVARCHAR (*n*)		A255V	A255V	A255	A255

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLINF] SET VARCHAR {ON|OFF}
```

where:

`SQLINF`

Indicates the Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Informix data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (A*n*V).

`OFF`

Maps the Informix data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). OFF is the default value.

Changing the Precision and Scale of Numeric Columns

How to:

Override Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLINF] SET CONVERSION RESET
ENGINE [SQLINF] SET CONVERSION format RESET
ENGINE [SQLINF] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLINF] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLINF`

Indicates the Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER fields to 7:

```
ENGINE SQLINF SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLINF SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER fields to the default:

```
ENGINE SQLINF SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLINF SET CONVERSION RESET
```

Customizing the Informix Environment

The Adapter for Informix provides several parameters for customizing the environment and optimizing performance.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLINF] SET PASSRECS {ON|OFF}
```

where:

`SQLINF`

Indicates the Adapter for Informix. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLINF] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLINF`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

`setting`

Is the optimization setting. Valid values are as follows:

`OFF` instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

`ON` instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

`FOCUS` passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

`SQL` passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLINF set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLINF set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

How to:

Optimize Requests Containing Virtual Fields

Example:

Using IF-THEN_ELSE Optimization Without Aggregation

Using IF-THEN_ELSE Optimization With Aggregation

Using IF-THEN_ELSE Optimization With a Condition That is Always False

Reference:

SQL Limitations on Optimization of DEFINE Expressions

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLINF] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLINF`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLINF SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLINF SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLINF SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Calling an Informix Stored Procedure Using SQL Passthru

Example:

Calling a Stored Informix Procedure

Informix Stored Procedure

Informix stored procedures are supported using SQL Passthru. These procedures need to be developed within Informix using the CREATE PROCEDURE command.

Example: Calling a Stored Informix Procedure

The supported syntax to call a stored procedure is shown below. It is recommended that you use the syntax below instead of the previously supported CALLINF syntax.

```
ENGINE SQLINF
EX SAMPLE PARM1,PARM2,PARM3...;
TABLE ON TABLE PCHOLD
END
```

The server supports invocation of stored procedures written according to the following rules:

- Only scalar input parameters are allowed, but not required.
- Output parameters are not allowed.

Example: Informix Stored Procedure

```
CREATE PROCEDURE SAMPLE()
RETURNING CHAR(11), CHAR(20), CHAR(15), DATE, CHAR(1);
DEFINE a1 CHAR(11);
DEFINE b1 CHAR(20);
DEFINE c1 CHAR(15);
DEFINE d1 DATE;
DEFINE e1 CHAR(1);
FOREACH entry FOR SELECT ssn5, last_name5, first_name5, birthdate5, sex5
INTO
a1,b1,c1,d1,e1 FROM 'edaqa'.NF29005
RETURN a1,b1,c1,d1,e1 WITH RESUME;
CONTINUE FOREACH;
END FOREACH
END PROCEDURE;
```

The following syntax is used to call the above stored procedure:

```
SQL SQLINF
EX SAMPLE;
TABLE ON TABLE PCHOLD
END
```

CHAPTER 17

Using the Adapter for Ingres

Topics:

The Adapter for Ingres allows applications to access Ingres data sources. The adapter converts application requests into native Ingres statements and returns optimized answer sets to the requesting application.

Preparing the Ingres Environment

Currently, the Ingres II environment is supported on UNIX and OpenVMS.

Procedure: How to Set Up the Environment on UNIX

The following environment variables must be set in this environment:

Location where Ingres II was installed.

```
II_SYSTEM = /rdbms/ing250
```

Installation code for Ingres II.

```
II_INSTALLATION = Ia
```

Path to Ingres II shared libraries.

```
LIBPATH=/rdbms/ing250/ingres/lib
```

Procedure: How to Set Up the Environment on OpenVMS

Check with the System Administrator to see if system logicals have been set up for Ingres II.

Configuring the Adapter for Ingres

In this section:

Overriding the Default Connection

How to:

Declare Connection Attributes From the Web Console

Declaring Connection Attributes Manually

Change the Default Connection

Example:

Selecting the Default Connection

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	Description
Database Name	Name of the Ingres II database you wish to connect to. There is no default database; a value must be entered.
User	Valid Ingres II username.
Password	Password that identifies the entered user name.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declaring Connection Attributes Manually

To connect to Ingres II using SET commands, issue the following:

```
ENGINE SQLING SET DATABASE dbname
ENGINE SQLING SET USER userid/password
ENGINE SQLING SET OPTIONs -R"xxxxxx", -G"yyyyyy"
```

where:

dbname

Is the name of the database.

userid

Is the user ID of the user.

password

Is the password in the Ingres II database. This is required only if the password is turned on in the database.

xxxxxx

Is the Ingres II role ID.

yyyyyy

Is the Ingres II group ID.

Overriding the Default Connection

Once connections have been defined, the connection named in the last SET DATABASE command serves as the default connection. You can override this default by re-issuing the SET DATABASE command to connect to a different database. The Adapter for Ingres does not support multiple data connections.

Syntax: How to Change the Default Connection

```
ENGINE [SQLING] SET DATABASE dbname
```

where:

`SQLING`

Indicates the Adapter for Ingres. You can omit this value if you previously issued the SET SQLENGINE command.

dbname

Is the name of the database to which you are connecting.

Note: You must terminate the session with one database before you switch to another one. Use the following command to terminate the session:

```
ENGINE SQLING END SESSION
```

Example: Selecting the Default Connection

The following SET DATABASE command selects the Ingres database server named SAMPLENAME as the default Ingres database server:

```
ENGINE SQLING SET DATABASE SAMPLENAME
```

Managing Ingres Metadata

In this section:

Creating Synonyms

Data Type Support

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Ingres data types.

Creating Synonyms

How to:

Create a Synonym Using the Web Console

Create a Synonym Manually

Example:

Using CREATE SYNONYM

Reference:

Managing Synonyms

Access File Keywords

Synonyms define unique names (or aliases) for each Ingres table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

   Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

   - **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.

   - **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.

5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.

6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.

7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.

Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLING [NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLING

Indicates the Data Adapter for Ingres.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLING AT DSN_A
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION ,SUFFIX=SQLING ,$
SEGNAME=SEG1_4 ,SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Name of the table or view. This value can include a location or owner name as follows: `TABLENAME=[`*`location.`*`][`*`owner.`*`]`*`tablename`*
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following chart provides information about the default mapping of Ingres II data types to server data types:

Ingres II Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
CHAR (*n*)	A(256)	A(256)	
VARCHAR (*n*)	A(256)	A(256)	
LONG VARCHAR			Not supported
TEXT A(254) A(254)	A(254)	A(254)	
DATE	HYYMDS	HYYMDS	
DATETIME			Not supported
MONEY	P18.2	P18.2	
FLOAT	D20.2	D20.2	
INTEGER	I11	I4	
SMALLINT	I6	I4	
DECIMAL(*p*,*s*)	P11	P4	
BYTE			Not supported
BYTE VARYING			Not supported
LONG BYTE			Not supported

CHAPTER 18

Using the Adapter for Interplex

Topics:

The Adapter for Interplex allows applications to access Interplex data sources. The adapter converts application requests into native Interplex statements and returns optimized answer sets to the requesting application.

Preparing the Interplex Environment

Prior to configuring the Adapter for Interplex, the following must be installed:

- A middleware product on the Unisys platform.
- An ODBC driver on UNIX or Windows/NT.

Configuring the Adapter for Interplex

In this section:

Declaring Connection Attributes

Overriding the Default Connection

Controlling the Connection Scope

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

Example:

Declaring Connection Attributes

In order to connect to the Interplex database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Interplex database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Interplex Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	**Description**
Data source	Data source name configured using the ODBC Driver Manager on Windows platforms or defined in the odbc.ini file on UNIX platforms.
User	Primary authorization ID by which you are known to Interplex data source.
Password	Password associated with the primary authorization ID.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

```
ENGINE [SQLIPX] SET CONNECTION_ATTRIBUTES [datasource]/userid,password
```

where:

`SQLIPX`

Indicates the Adapter for Interplex. You can omit this value if you previously issued the SET SQLENGINE command.

`datasource`

Is the name of the Interplex data source you wish to access.

`userid`

Is the primary authorization ID by which you are known to Interplex.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Interplex database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLIPX SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLIPX] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLIPX`

Indicates the Adapter for Interplex. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Interplex database server named SAMPLENAME as the default Interplex database server:

```
ENGINE SQLIPX SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLIPX] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLIPX`

Indicates the Adapter for Interplex. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Interplex Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Interplex data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Interplex table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

   Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

   - **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
   - **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.

7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Note: Due to the nature of the DMS database (hierarchical model), Select and Update are readily available using InterPlex SQL. Insert and Delete are not automatic, but require a programmer to supply business rule code via a copy procedure.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLIPX [AT connection|AT '']
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLIPX

Indicates the Data Adapter for Interplex.

AT *connection*

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

AT ' '

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local database server.

NOCOLS

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLIPX AT DSN_A
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLIPX ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the Interplex table. The value assigned to this attribute can include the name of the owner (also known as schema). and the database link name as follows: `TABLENAME=[`*`owner.`*`]`*`table`*
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

How to:

Override Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLIPX] SET CONVERSION RESET
ENGINE [SQLIPX] SET CONVERSION format RESET
ENGINE [SQLIPX] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLIPX] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLIPX`

Indicates the Adapter for Interplex. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

precision

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

scale

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLIPX SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLIPX SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLIPX SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLIPX SET CONVERSION RESET
```

Customizing the Interplex Environment

The Adapter for Interplex provides several parameters for customizing the environment and optimizing performance.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLIPX] SET PASSRECS {ON|OFF}
```

where:

`SQLIPX`

Indicates the Adapter for Interplex. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

In this section:

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

How to:

Example:

Reference:

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLIPX] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLIPX`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

OFF instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

ON instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

FOCUS passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

SQL passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLIPX set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLIPX set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

How to:

Optimize Requests Containing Virtual Fields

Example:

Using IF-THEN_ELSE Optimization Without Aggregation

Using IF-THEN_ELSE Optimization With Aggregation

Using IF-THEN_ELSE Optimization With a Condition That is Always False

Reference:

SQL Limitations on Optimization of DEFINE Expressions

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLIPX] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLIPX`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLIPX SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLIPX SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLIPX SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

CHAPTER 19

Using the Adapter for JDBC

Topics:

The Adapter for JDBC allows applications to access JDBC data sources. The adapter converts application requests into native JDBC calls and returns optimized answer sets to the requesting application.

Preparing the JDBC Environment

In order to use the Adapter for JDBC, you must install the JDBC driver for whichever data source you would like to access. For example, if you want to access a UniData database, you must install a JDBC driver for UniData.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the JDBC Driver files using the environment variable $CLASSPATH. For example, to set the location of the JDBC Driver files to /usr/driver_files, specify:

   ```
   CLASSPATH=/usr/driver_files
   export CLASSPATH
   ```

2. Identify the installation directory of the Java Development Kit using the environment variable $JDK_HOME. For example, if you want to set the location of the Java Development Kit to /usr/java, specify:

   ```
   JDK_HOME=/usr/java
   export JDK_HOME
   ```

3. Identify the installation directory of the Java Virtual Machine using the environment variable $LD_LIBRARY_PATH. For example, if you want to set the location of the java virtual machine to /usr/j2sdk1.4.2_01/jre/lib/i386/server, specify:

   ```
   LD_LIBRARY_PATH=/usr/j2sdk1.4.2_01/jre/lib/i386/server
   export LD_LIBRARY_PATH
   ```

Note: If the server is running with security on, the LD_LIBRARY_PATH variable will be ignored. In this case, you must use IBI_LIBPATH.

Configuring the Adapter for JDBC

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a JDBC data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one JDBC data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	**Description**
Connection name	Arbitrary connection name to be used to reference the connection.
URL	Location URL for the JDBC data source.
User	Primary authorization ID by which you are known to the target database.
Password	Password associated with the primary authorization ID.
Driver name	Name for the JDBC driver.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLJDBC SET CONNECTION_ATTRIBUTES connection 'URL'/userid,password
```

where:

`SQLJDBC`

Indicates the Adapter for JDBC. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name.

`URL`

Is the URL to the location of the JDBC data source.

`userid`

Is the primary authorization ID by which you are known to the target database.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to a books database using the JDBC Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLJDBC SET CONNECTION_ATTRIBUTES CON1
  'jdbc:transoft://rediron6:7000/books.udd'/MYUSER,PASS
```

Overriding the Default Connection

How to:

Change the Default Connection

Example:

Selecting the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE [SQLJDBC] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLJDBC`

Indicates the Adapter for JDBC. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.

- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects CON1 as the default connection.

```
ENGINE SQLJDBC SET DEFAULT_CONNECTION CON1
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLJDBC] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLJDBC`

Indicates the Adapter for JDBC. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing JDBC Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLJDBC to identify the Adapter for JDBC.

Syntax: How to Identify the Adapter for JDBC

```
FILE[NAME]=file, SUFFIX=SQLJDBC [,$]
```

where:

file

Is the file name for the Master File. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

`SQLJDBC`

Is the value for the Adapter for JDBC.

Accessing Database Tables

If you choose to access a remote third-party table using JDBC, you must locally install the RDBMS' JDBC Driver.

The iWay Server can access third-party database tables across the network JDBC. You must specify a URL for the data source and, possibly, a user ID and/or password for the database you are accessing. You can define these parameters in either the server's global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.
10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)

Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLJDBC [AT connection]
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLJDBC

Indicates the Data Adapter for JDBC.

`AT connection`

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

The following example creates a synonym named nf29004 for an object named NF29004.

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLJDBC AT CON1
```

The synonym creates the following Master File and Access File.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLJDBC ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the JDBC table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: `TABLENAME=[owner.]table[@databaselink]`
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=connection` CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first n fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

How to:

Override Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLJDBC] SET CONVERSION RESET
ENGINE [SQLJDBC] SET CONVERSION format RESET
ENGINE [SQLJDBC] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLJDBC] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLJDBC`

Indicates the Adapter for JDBC. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLJDBC SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLJDBC SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLJDBC SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLJDBC SET CONVERSION RESET
```

Customizing the JDBC Environment

The Adapter for JDBC provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to JDBC.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE [SQLJDBC] SET TIMEOUT {nn|0}
```

where:

`SQLJDBC`

Indicates the Adapter for JDBC. You can omit this value if you previously issued the SET SQLENGINE command.

`nn`

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLJDBC] SET PASSRECS {ON|OFF}
```

where:

`SQLJDBC`

Indicates the Adapter for JDBC. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

In this section:

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

How to:

Optimize Requests

Example:

SQL Requests Passed to the RDBMS With Optimization OFF

SQL Requests Passed to the RDBMS With Optimization ON

Reference:

SQL Generation in Optimization Examples

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLJDBC] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLJDBC`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

OFF instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

ON instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

FOCUS passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

SQL passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLJDBC set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLJDBC set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

How to:

Optimize Requests Containing Virtual Fields

Example:

Using IF-THEN_ELSE Optimization Without Aggregation

Using IF-THEN_ELSE Optimization With Aggregation

Using IF-THEN_ELSE Optimization With a Condition That is Always False

Reference:

SQL Limitations on Optimization of DEFINE Expressions

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLJDBC] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLJDBC`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLJDBC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLJDBC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLJDBC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

CHAPTER 20

Using the Adapter for Lawson

Topics:

The Adapter for Lawson allows applications to access Lawson data sources.

The Adapter for Lawson automatically applies the appropriate Lawson security rules as defined in the Lawson LAUA repository. The adapter transparently generates dynamic DBA statements and/or dynamic WHERE clauses.

Preparing the Server Environment for Adapter Configuration

How to:

Extract the Lawson Security Information

Copy the Lawson Security Information Flat Files to the Server

Run glawfils.fex

To prepare the server environment for adapter configuration, perform the following:

1. Verify that the server is installed.
2. On the machine where Lawson is running, extract the Lawson security information into flat files using the lawsdmp.sh utility.
3. On the server, create a lawsec directory and add it to the path.
4. Copy the Lawson security information flat files to the server using the lawscpy utility.
5. Run the glawfils.fex procedure to change the flat files into a format the server can understand.

Every time you change Lawson security information, you need to update the server files. See *Updating Lawson Security Information* on page 20-7.

Procedure: How to Extract the Lawson Security Information

Perform the following steps on the machine where Lawson is running:

1. On the Lawson System, create the ibi_officer user ID, or any user ID with Security Officer capabilities. (This ID must be allowed to run the Lawson RNGDBDUMP command.)
2. Copy lawsdmp.sh, lawsdmpu.sh, and lawsfil.txt from the etc directory of EDAHOME into the home directory of the ibi_officer.

   EDAHOME is the location of the files that run your server. The following table shows the default locations for EDAHOME.

Operating system	EDAHOME default directory location and name
Windows	ibi\srv52\home
UNIX and OS/400	/home/iadmin/ibi/srv52/home

   Note: For more information on EDAHOME, see the *iWay Server Administration* manual.
3. Change the execute attribute of lawsdmp.sh and of lawsdmpu.sh to *On*.
4. Schedule or run lawsdmp.sh (or lawsdmpu.sh) to extract the latest changes to the Lawson Security System.

Procedure: How to Copy the Lawson Security Information Flat Files to the Server

1. Start the server and the Web Console.
2. Open the Configure Application Path window (click either *Metadata* or *Procedures* in the Web Console menu, then click *Configure Application Path*).
3. Create a directory named lawsec in the APPROOT directory by clicking *New directory*. This is the Lawson Security Directory.

   APPROOT is a directory that contains applications and sample files. The following table shows the default locations for APPROOT.

Operating System	Default Location
Windows	ibi\apps
UNIX, OS/400	home/iadmin/ibi/apps

   Note: For more information on the Web Console and APPROOT, see the *iWay Server Administration* manual.
4. Include lawsec in the APPPATH by dragging the folder next to its name into the APPPATH box and clicking *Set APP PATH*.
5. Copy the following files from the etc directory in EDAHOME to the lawsec directory:

   On UNIX: copy lawscpy.sh, lawsftp.dat, and lawscpy.bat.

   On Windows: copy lawsftp.dat and lawscpy.bat.
6. Edit the following files:

   On UNIX: edit lawscpy.sh to set the execute attribute to On, and to specify the correct source and target directories.

   On Windows: edit lawscpy.bat to specify the correct source and target directories.

   Note: The source directory is the home directory of ibi_officer that you created in Step 1 of the previous section. The target directory is lawsec.
7. Edit lawsftp.dat to indicate the hostname, user ID, and password of the remote machine that contains the extracts from running the lawsdmp.sh script.

8. If the server with the Adapter for Lawson is on the same server as the Lawson System, run lawscpy.sh (on UNIX) or lawscpy.bat (on Windows) to copy the extracted files from the lawsdmp run to the Lawson Security Directory (the lawsec directory in APPROOT) of the server.

   If it is not on the same server, run ftp with lawsftp.dat as input to ftp the extracted files to the Lawson Security Directory (the lawsec directory in APPROOT), as follows:

   ```
   ftp -niv < lawsftp.dat
   ```

Procedure: How to Run glawfils.fex

1. Copy glawcopr.fex, glawfils.fex, and glawincl.fex (glaw*.fex) from the catalog directory in EDAHOME to the lawsec directory.
2. If you haven't already done so, start the server and the Web Console.
3. Click *Procedures* in the Web Console menu, and then expand the lawsec directory.
4. Edit glawincl.fex by clicking it and clicking *Edit* in the pop-up menu.
   a. Set EDAPATH, TEMP and &FTMDIR to the Lawson Security Directory (the lawsec directory in APPROOT).
   b. Set &HOMECAT to the catalog directory in EDAHOME, where lawsflds.* resides.
   c. Set &PLATFORM to PC if Lawson System is on Windows and to UNIX if Lawson System is on a UNIX system.

   When finished, click *Save*.
5. Run glawfils.fex by clicking it, and then clicking *Run*.

This process creates the LAW* FOCUS files needed by the Adapter for Lawson to apply the necessary filtering to queries against the Lawson database files.

Configuring the Adapter for Lawson

How to:

Configure the Adapter for Lawson From the Web Console

Reference:

Declaring Lawson Connection Attributes From the Web Console

Once you have completed preparing the server environment, you are ready to configure the Adapter for Lawson. You can configure the adapter in one of two ways:

- Using the Web Console.
- Manually editing the Declare Connections Attributes in the global server profile, edasprof.prf.

Procedure: How to Configure the Adapter for Lawson From the Web Console

To configure the adapter from the Web Console:

1. Start the server, open the Web Console, and click *Data Adapters*.

   The Configuring Data Adapters page opens.

2. If necessary, expand the Add folder.
3. Expand the Lawson folder, and click the Lawson icon.
4. In the right pane, enter a DBMS Suffix and a Default Product Line attribute, and click *Configure*.

   The following reference item describes these attribute values.

Reference: Declaring Lawson Connection Attributes From the Web Console

Attribute	Description
DBMS Suffix	Specify the suffix of the database that Lawson is using. Possible values are: `SQLORA` - Oracle `SQLMSS` - Microsoft SQL Server `SQLINF` - Informix `SQLDB2` - DB2
Default Product Line	This value is customized by the site at installation of the Lawson system.

When you add the connection attributes, the following takes place:

1. Adds the following lines to edasprof.prf:

```
SET USER=LAWSON
ENGINE LAWSON SET LAW DBMS dbmssuffix
ENGINE LAWSON SET LAW PRODLINE lawidi
```

where:

dbmssuffix

Is the suffix for the database used by the Lawson installation.

lawidi

Is the default Lawson Product Line to access.

Note: If you change the value for SET USER, you will not be able to access Lawson data.

2. Adds the following line to edaserve.cfg:

```
law_access=y
```

Managing Metadata

Once the Adapter for Lawson is configured, set up the metadata for the underlying database management system. See the documentation for the appropriate DBMS adapter.

Updating Lawson Security Information

How to:

Update Lawson Security Information

The Adapter for Lawson works behind the scenes to control end users' access to the data in the Lawson system. When a user makes a request against a Lawson table, the Adapter for Lawson asks the following questions:

- Is this a Lawson File?
- Is this a Lawson User?
- Does the user have access to this table?

Once these questions have been answered, the Adapter for Lawson does the following:

- Dynamically generates DBA for any Application Security System Code, Record Level security, File Security, and Field Security. For more information on DBA, see the *Describing Data* manual.
- Generates WHERE clauses for any Lawson Company and Process Level, Record Security, and Field Security restrictions. These WHERE clauses are appended to any requests for data made to the Lawson data.

Important: Automatic Passthru has been disabled against Lawson data. You must set Automatic Passthru off with the following command:

```
SET APT = OFF
```

For more information on Automatic Passthru, see your iWay documentation.

To update the Lawson security information, you must update the server files as described in the following procedure.

Procedure: How to Update Lawson Security Information

1. Run lawsdmp.sh (or lawsdmpu.sh) to extract the latest changes to the Lawson Security System in the ibi_officer user ID home directory.
2. If the server with the Adapter for Lawson is in the same server as the Lawson System, run lawscpy.sh (on UNIX) or lawscpy.bat (on Windows) to copy the extracted files from the lawsdmp run to the Lawson Security Directory (the lawsec directory in APPROOT) of the server. You can find these files in the lawsec directory.

   If not on the same server, run ftp with lawsftp.dat as input to ftp the extracted files to the Lawson Security Directory (the lawsec directory in APPROOT). This should be done as follows:

   ```
   ftp -niv < lawsftp.dat
   ```
3. Start the server and the Web Console.
4. Click *Procedures* in the Web Console menu, and then expand the lawsec directory.
5. Run glawfils.fex by clicking it, and then click *Run*.

CHAPTER 21

Using the Adapter for Microsoft Access

Topics:

The Adapter for Microsoft Access allows applications to access Microsoft Access data sources. The adapter converts application requests into native Microsoft Access statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the Microsoft Access Environment

The Adapter for Microsoft Access minimally requires the installation of ODBC 32-bit Driver Manager. The configuration of a system data source name allows you to connect to a local or remote Microsoft Access data source.

Procedure: How to Set Up the Environment on Windows

There is no environment set up needed to access Microsoft Access.

Configuring the Adapter for Microsoft Access

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Example:

In order to connect to an Microsoft Access database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Microsoft Access database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Microsoft Access Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	**Description**
Data source	Data source name configured using the ODBC Driver Manager.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

Trusted authentication.

```
ENGINE [SQLMAC] SET CONNECTION_ATTRIBUTES datasource
```

Explicit authentication. For password protected data sources:

```
ENGINE [SQLMAC] SET CONNECTION_ATTRIBUTES datasource/,[password]
```

where:

`SQLMAC`

Indicates the Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

`datasource`

Is the name of the Microsoft Access data source you wish to access.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Microsoft Access database server named SAMPLESERVER with a password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile. The following SET CONNECTION_ATTRIBUTES command connects to the Microsoft Access database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLMAC SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

Overriding the Default Connection

How to:

Change the Default Connection

Example:

Selecting the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLMAC] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLMAC`

Indicates the Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Microsoft Access database server named SAMPLENAME as the default Microsoft Access database server:

```
ENGINE SQLMAC SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLMAC] SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLMAC`

Indicates the Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Microsoft Access Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft Access data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Microsoft Access table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value. See *Controlling the Mapping of Large Character Data Types* for more information.
11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.
12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```
14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.

Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLMAC [AT connection|AT '']
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

`SQLMAC`

Indicates the Data Adapter for Microsoft Access.

`AT connection`

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`AT ' '`

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local database server.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLMAC AT MACDSN
END
```

Generated Master File nf29004.mas

```
FILE=NF29004, SUFFIX=SQLMAC ,$
SEGNAME=NF29004, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4,     I11, I4,  MISSING=ON ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I11, I4,  MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=NF29004,TABLENAME=NF29004,
CONNECTION=MACDSN,KEYS=1, WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the Microsoft Access table name.
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following chart provides information about the default mapping of Microsoft Access data types to server data types:

Microsoft Access Data Types	Data Type		Remarks
	USAGE	**ACTUAL**	
TEXT (*n*) in (1...255)	A255	A255	
MEMO (*n*) in (1...65,535)	A*m* TX50	A*m* TX	*m*=2**n*.
AUTONUMBER (Long Integer)	I6	I4	
YES/NO	I11	I4	
OLE OBJECT (drawings, waveform audio files, bitmapped graphics)	BLOB	BLOB	Supported via API
HYPERLINK	A*m* TX50	A*m* TX	*m*=2**n*.
CURRENCY (range of -922337203685477.5808 to +922337203685477.5808)			

Microsoft Access Data Types	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
General	P21.4	P10	
Currency	P21.4	P10	
Euro	P21.4	P10	
Fixed	P21.4	P10	
Standard	P21.4	P10	
Percent	P21.4	P10	
Scientific	P21.4	P10	
NUMBER (Subtypes)			
Byte (range 0 to 255)	I6	I4	
Integer (range of 32,768 to 32,767)	I6	I4	
Long Integer (range of 2,147,483,648 to 2,147,483,647)	I11	I4	
Single (range of 3.4*1038 to +3.4*1038)	D20.2	D8	
Double (range of 1.797*10308 to +1.797*10308)	D20.2	D8	
Decimal (-1028 to +1028-1)	P19	P10	
Replication ID	A38	A38	
DATE/TIME			
General Date	HYYMDS	HYYMDS	
Long Date	HYYMDS	HYYMDS	
Medium Date	HYYMDS	HYYMDS	
Short Date	HYYMDS	HYYMDS	
Long Time	HYYMDS	HYYMDS	

Microsoft Access Data Types	Data Type		Remarks
	USAGE	**ACTUAL**	
Medium Time	HYYMDS	HYYMDS	
Short Time	HYYMDS	HYYMDS	

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Microsoft Access data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Microsoft Access Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
CHAR (*n*)	*n* is an integer between 1 and 2000	A*n*	A*n*	TX50	TX
VARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*	A*n*	TX50	TX
VARCHAR2 (*n*)	*n* is an integer between 1 and 4000	A*n*	A*n*	TX50	TX
RAW (*n*)	*n* is an integer between 1 and 2000 *m* = 2 * *n*	A*m*	A*m*	TX50	TX

Syntax: How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLMAC] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

`SQLMAC`

Indicates the Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

`ALPHA`

Maps the Microsoft Access data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). ALPHA is the default value.

`TEXT`

Maps the Microsoft Access data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

`BLOB`

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Microsoft Access data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

For OS/390 and z/OS, maps the Microsoft Access data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as binary large object (BLOB).

Changing the Precision and Scale of Numeric Columns

How to:

Override Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLMAC] SET CONVERSION RESET
ENGINE [SQLMAC] SET CONVERSION format RESET
ENGINE [SQLMAC] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLMAC] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLMAC`

Indicates the Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	**MAX Precision**
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLMAC SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLMAC SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLMAC SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLMAC SET CONVERSION RESET
```

Customizing the Microsoft Access Environment

The Adapter for Microsoft Access provides several parameters for customizing the environment and optimizing performance.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLMAC] SET PASSRECS {ON|OFF}
```

where:

`SQLMAC`

Indicates the Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLMAC] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLMAC`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

OFF instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

ON instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

FOCUS passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

SQL passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLMAC set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLMAC set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLMAC] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLMAC`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLMAC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLMAC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLMAC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Specifying Block Size for Retrieval Processing

How to:

Specify the Block Size for Array Retrieval

Specify the Block Size for Insert Processing

The Adapter for Microsoft Access supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Specify the Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE [SQLMAC] SET FETCHSIZE n
```

where:

`SQLMAC`

Indicates the Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Syntax: How to Specify the Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE [SQLMAC] SET INSERTSIZE n
```

where:

`SQLMAC`

Indicates the Adapter for Microsoft Access. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

CHAPTER 22

Using the Adapter for Microsoft Analysis Server

Topics:

The Adapter for Microsoft Analysis Server allows applications to access Microsoft Analysis Server data sources. The adapter converts application requests into native Microsoft Analysis Server statements and returns optimized answer sets to the requesting application.

Preparing the Microsoft Analysis Server Environment

The Adapter for Microsoft Analysis Server minimally requires the installation of the Microsoft OLE DB software which is part of the Microsoft Data Access Components (MDAC) installation.

Procedure: How to Set Up the Environment on Windows

On Windows, you only need to install MDAC 2.6 or higher.

The Adapter for Microsoft Analysis Server provides read-only access to analytical data stored in Analytical Engine cubes. The adapter is an OLE DB for OLAP consumer. It employs Multi-dimensional Data Retrieval (MDX) language to access aggregated or rolled-up data used for decision-support processing.

You can use any server front-end technology with the WebFOCUS reporting engine to access the OLAP data retrieved by the server for Windows. This makes your OLAP data available to your entire enterprise. Additionally, data from Analytical Engine cubes can be joined with data from any other supported data source, providing additional information to your analytical process.

Setting Microsoft Analysis Server Security

There is only one method by which you can be authenticated when connecting to a Microsoft Analysis Server:

Operating System (Trusted Mode). The user ID and password used to log on to the operating system are automatically used to connect to the Analytical Engine. The server data access agent impersonates an operating system user according to the server deployment mode. The agent process establishes a connection to a Analytical Engine based on the impersonated operating system user credentials.

The Adapter for Microsoft Analysis Server recognizes the security restrictions that the OLAP Database Administrator specifies and applies them to catalogs and cubes using Analytical Engine Services. This includes the role of security on the database and the application level. Although the Server for Windows establishes the connection to the Analytical Engine, an impersonation of the client is used to maintain access rights.

- **Security off.** If the server is running with security off, the user ID and password of the interactive user (the person logged onto a Windows or Workstation) is passed to the Analytical Engine in order to establish a connection and access rights. This is the logon ID specified for the Windows or Windows Workstation after it is first started.
- **Security on.** If the server is running with security on, the user ID and password of the client connecting to the server is passed to the Analytical Engine in order to establish a connection and access rights. This process is called impersonation.

Accessing a Microsoft Analysis Server

Using the standard rules for deploying OLE DB, the server supports connections to a:

- Local Microsoft Analysis Server.
- Remote Microsoft Analysis Server. To connect to a remote Analytical Engine, it is recommended that you have MDAC 2.6 or greater installed.

Configuring the Adapter for Microsoft Analysis Server

In this section:

Declaring Connection Attributes

Changing the Default Connection

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

The SET CONNECTION_ATTRIBUTES command allows you to declare a connection to one Microsoft Analysis Server.

You can declare connections to more than one Analytical Engine by issuing multiple SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query referencing that connection is issued (see *Changing the Default Connection* on page 22-5). You can include SET CONNECTION_ATTRIBUTES commands in an RPC or a server profile. The profile can be encrypted.

If you issue multiple SET CONNECTION_ATTRIBUTES commands, the first SET CONNECTION_ATTRIBUTES command sets the default Microsoft Analysis Server to be used.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, select *Data Adapters*.
2. Expand the *Add* folder, expand the *OLAP* group folder, then expand the *MS OLAP Services* folder and click on the version of the SQL server you wish to use. The Add Microsoft Analysis Server to Configuration pane opens.
3. Supply values for the following parameters:

Field	Description
Server	Name of the machine where Microsoft Analysis Services is running. The value for this field is used to describe connection attributes for the Microsoft Analysis Server.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

   Tip: Once the adapter is configured, the connection name appears in the navigation pane under *Configured*. If you left-click either the adapter name or the connection name, an appropriate set of options becomes available.

Syntax: How to Declare Connection Attributes Manually

For Trusted mode authentication

```
ENGINE [MSOLAP] SET CONNECTION_ATTRIBUTES datasource_name/
```

where:

`MSOLAP`

Indicates the Adapter for Microsoft Analysis Server. You can omit this value if you previously issued the SET SQL ENGINE command.

`datasource_name`

Is the server name used as a connect descriptor to an Analytical Engine across the network. If omitted, the local database server will be set as the default.

Changing the Default Connection

Once all Analytical Engine connections to be accessed have been declared using the SET CONNECTION_ATTRIBUTES command, there are two ways to select a specific Analytical Engine connection from the list of declared connections:

- You can select a default connection using the SET DEFAULT_CONNECTION command. If you do not issue this command, the connection_name value specified in the *first* SET CONNECTION_ATTRIBUTES command is used.
- You can include the CONNECTION= attribute in the Access File of the table specified in the current SQL query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION= attribute is automatically included in the Access File. This attribute supercedes the default connection.

Syntax: How to Change the Default Connection

```
ENGINE [MSOLAP] SET DEFAULT_CONNECTION [datasource_name]
```

where:

`MSOLAP`

Indicates the Adapter for Microsoft Analysis Server. You can omit this value if you previously issued the SET SQLENGINE command.

`datasource_name`

Is the data source name specified in a previously issued SET CONNECTION_ATTRIBUTES command. If omitted, then the local database server will be set as the default. If this connection name has not been previously declared, a FOC1671 message is issued.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection_name specified in the last command will be the active connection_name.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, a FOC1671 message is issued.

Example: Changing the Default Connection

The following SET DEFAULT_CONNECTION command selects the Microsoft Analysis Server named OLAPSERV as the default Analytical Engine:

```
ENGINE MSOLAP SET DEFAULT_CONNECTION OLAPSERV
```

Note: You must have previously issued a SET CONNECTION_ATTRIBUTES command for OLAPSERV.

Managing Microsoft Analysis Server Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft Analysis Server data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Analytical Engine cube that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

For details on how to create a synonym through the Web Console, see the *Server Configuration and Operation* manual for your platform.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Editing Metadata page opens.

   Note: To create a synonym, you must have configured the adapter. See *Configuring the Adapter for Microsoft Analysis Server* on page 22-3 for more information.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane (Step 1 of 2).

3. Click a connection for the configured adapter. The *Select Cubes* button is displayed in the right pane.

4. Click *Select Cubes* to display the Create Synonym pane (Step 2 of2).

5. Enter the following parameters, as appropriate:

Parameter	**Description**
Select Application	Select an application directory. baseapp is the default value.
Prefix/Suffix	If you have cubes with identical names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll cubes, assign the prefix HR to distinguish the synonyms for the human resources cubes. Note that the resulting synonym name cannot exceed 64 characters. If all cubes have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

6. Select the cubes for which you wish to create synonyms:

- To select all cubes in the list, select the check box to the left of the *Default Synonym Name* column heading.
- To select specific cubes, select the corresponding check boxes.

7. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

8. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

9. A status window displays the message:

```
All Synonyms Created Successfully
```

10. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR cube_name DBMS MSOLAP [AT datasource_name]
  DATABASE database_name
 END
```

where:

appname

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (it can be a maximum of 64 characters).

`cube_name`

Is the name of the OLAP cube. See the Microsoft Analysis Server documentation for specific naming conventions.

`MSOLAP`

Indicates the Adapter for Microsoft Analysis Server.

`AT datasource_name`

Is the *datasource_name* as previously specified in a SET CONNECTION_ATTRIBUTES command. When the server creates the synonym, this *connection_name* becomes the value for DATASOURCE= in the Access File.

`database_name`

Is the name of the data source (catalog) the OLAP cube is using.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.
- CREATE SYNONYM creates a Master File and Access File which represents the server metadata.

Example: Using CREATE SYNONYM

Use the following syntax to create a synonym for the Sales cube.

```
CREATE SYNONYM sales FOR sales DBMS MSOLAP AT OLAPSVR1
DATABASE 'FOODMART 2000'
END
```

Generated Master File sales.mas

```
FILENAME=SALES, SUFFIX=MSOLAP,$
SEGNAME=SALES, SEGTYPE=S0,$

$ DIMENSION: -MEASURES-
 FIELDNAME=PROFIT, ALIAS=Profit, USAGE=D17.9, ACTUAL=D8, MISSING=ON,
   TITLE='Profit', REFERENCE=(SUM.STORE_SALES - SUM.STORE_COST),
   PROPERTY=CALCULATED MEASURE,
 FIELDNAME=SALES_COUNT, ALIAS='Sales Count', USAGE=I11, ACTUAL=I4,
   MISSING=ON,
   TITLE='Sales Count', REFERENCE=CNT.SALES_COUNT, PROPERTY=MEASURE,  $

$  DIMENSION:  Customers
 FIELDNAME=COUNTRY, ALIAS=Country, USAGE=A6, ACTUAL=A6,
   TITLE='Country', WITHIN='*[Customers]', $
```

The Master File that is produced contains a segment for each dimension in the OLAP outline. For each generation within a dimension, a field description is produced. The WITHIN keyword is used to describe the hierarchy inherent in each dimension structure.

Note: Master Files should not be altered. Manipulating the Master File causes incorrect query results.

The Measures dimension can contain calculated and non-calculated members. Calculated members in the Measures dimension will contain a Reference defining the calculated member and a Property. Some properties are mandatory while others may be user-defined.

Generated Access File sales.acx

```
SEGNAME=SALES,
CUBENAME=Sales,
SCHEMA=,
CATALOG=FoodMart 2000,
DATASOURCE=OLAPSVR1, $
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`CUBENAME`	Name of the OLAP cube.
`SCHEMA`	Schema that describes the cube.
`DATASOURCE`	Microsoft Analysis Server name.
`CATALOG=`	Name of the data source (catalog) the OLAP cube is using.

Note: Access Files reflect the structure of the cube Master File that is produced and contains a segment for each dimension in the OLAP outline. For each generation within a dimension, a field description is produced. The WITHIN keyword is used to describe the hierarchy inherent in each dimension structure.

Customizing the Microsoft Analysis Server Environment

In this section:

The Adapter for Microsoft Analysis Server provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Activating NONBLOCK Mode and Issuing a TIMEOUT Limit

You can use the NONBLOCK command to prevent runaway queries. The Adapter for Microsoft Analysis Server uses non-blocking protocol for query execution.

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Microsoft Analysis Server.

Syntax: How to Activate NONBLOCK Mode and Issue a TIMEOUT Limit

```
ENGINE MSOLAP SET NONBLOCK n [TIMEOUT m]
```

where:

n

Is the polling period in seconds.

m

Is the total timeout after which the query is terminated if it does not get completed.

Accelerating Queries

Using the SLICEROPTIMIZATION command usually significantly accelerates queries where some dimensions are referenced for tests only.

Syntax: How to Use SLICEROPTIMIZATION

```
ENGINE MSOLAP SET SLICEROPTIMIZATION {ON|OFF}
```

where:

`ON`

Optimizes some queries by putting some member sets on the slicer axis and using the AGGREGATE function rather than retrieving them. ON is the default value.

`OFF`

Suppresses the optimization. This is provided for compatibility and performance comparison purposes.

Example: Using the SLICEROPTIMIZATION Command

In the following request, the PRODUCT dimension is referenced for testing only, therefore the AGGREGATE of all the products containing 'Best' can be put on the slicer. As a result, much less data will have to be read by the adapter.

```
TABLE FILE SALES
WRITE SALES
BY YEAR
WHERE PRODUCT_NAME LIKE 'Best%';
END
```

Using Verbs Different From Measure Aggregator

You can use the CHECKAGGREGATION SET command to allow the use of any verb against any measure regardless of the measure aggregator.

Syntax: How to Set CHECKAGGREGATION

```
ENGINE MSOLAP SET CHECKAGGREGATION {ON|OFF}
```

where:

`ON`

Only allows verbs to be used that match the measure's aggregator. This is the default value.

`OFF`

Allows any verb to be used with any measure regardless of the measure's aggregator.

Retrieving Columns With or Without Values

You can use the EMPTY SET command to display or suppress the column names in a report for which there are no values.

Syntax: How to Activate EMPTY Mode

```
ENGINE MSOLAP SET EMPTY {ON|OFF}
```

where:

`ON`

Does not suppress columns without values.

`OFF`

Suppresses columns with values. This is the default.

Example: Activating EMPTY Mode

In the following request, EMPTY is set to OFF. Therefore, columns with no data are suppressed.

```
TABLE FILE SALES
PRINT PROFIT
BY CITY
END
```

The output is:

```
City           Profit
----           ------
Albany         8,516.532900000
Altadena       3,345.876900000
Anacortes      956.535100000
```

With EMPTY set to ON, the column with no data is displayed.

```
City           Profit
----           ------
Acapulco                     .
Albany         8,516.532900000
Altadena       3,345.876900000
Anacortes        956.535100000
```

Adapter Functionality

The Adapter for Microsoft Analysis Server supports:

- **JOIN commands.** The Adapter for Microsoft Analysis Server does not support direct JOINs from or to a cube. Therefore, SQL JOINs using WHERE criteria are not supported. To achieve a joined request, use the FOCUS MATCH FILE command or create a HOLD file and include the HOLD file in the joined request.
- **SUM and PRINT commands with Measures dimensions.** When using these commands in a request, the verb used against the Adapter for Microsoft Analysis Server has to correspond to the aggregator. If the aggregator is SUM, the only verb that can be used in the request is SUM or WRITE.

Example: Using SUM and PRINT Commands Correctly in a Report Request

In the following request, the PRINT verb is used, and SUM is the aggregator for STORE_COSTS.

```
TABLE FILE SALES
PRINT STORE_COSTS
BY CITY
END
```

This request is incorrect, and the following message displays:

```
(FOC11207) ERROR IN THE QUERY:
(FOC11243) PRINT VERB SHOULD NOT BE USED ON MEASURE Store Cost WITH
AGGREGATOR SUM
```

The correct request is:

```
TABLE FILE SALES
SUM STORE_COSTS
BY CITY
END
```

The output is:

```
City          Store Cost
----          ----------
Albany        5,593.28
Altadena      2,239.71
Anacortes     641.93
Arcadia       2,045.73
Ballard       2,169.51
```

CHAPTER 23

Using the Adapter for Microsoft SQL Server

Topics:

The Adapter for Microsoft SQL Server allows applications to access Microsoft SQL Server data sources. The adapter converts data or application requests into native Microsoft SQL Server statements and returns optimized answer sets to the requesting program.

Preparing the Microsoft SQL Server Environment

You can set up the Microsoft SQL Server Environment on Windows and UNIX.

Procedure: How to Set Up the Environment on Windows

The Microsoft SQL Server environment is set up during the installation of the Microsoft SQL Server, Client, and MDAC software.

No additional setup steps are required.

Procedure: How to Set Up the Environment on UNIX

Identify the location of the JDBC Driver files using the environment variable $CLASSPATH. For example, to set the location of the JDBC Driver files to /usr/driver_files, specify:

```
CLASSPATH=/usr/driver_files
export CLASSPATH
```

Identify the installation directory of the Java Development Kit using the environment variable $JDK_HOME. For example if you want to set the location of the Java Development Kit to /usr/java, specify:

```
JDK_HOME=/usr/java
export JDK_HOME
```

Identify the installation directory of the Java Virtual Machine using the environment variable $LD_LIBRARY_PATH. For example, if you want to set the location of the java virtual machine to /usr/j2sdk1.4.2_01/jre/lib/i386/server, specify:

```
LD_LIBRARY_PATH=/usr/j2sdk1.4.2_01/jre/lib/i386/server
export LD_LIBRARY_PATH
```

Note: If the server is running with security on, the LD_LIBRARY_PATH variable is ignored. In this case, you must use IBI_LIBPATH.

Accessing Microsoft SQL Server Remotely

You can access Microsoft SQL Server on a remote node. To access Microsoft SQL Server remotely, you must:

- Locally install MDAC 2.6 or higher.
- Know the name of the remote Microsoft SQL Server instance.

All Microsoft SQL Servers installed are defined by using a unique NetBEUI name. The server can access any Microsoft SQL Server on the network, provided you define a valid user ID and password, as well as the name of the Microsoft SQL Server instance. You can define these parameters in either the server's global profile or a user profile.

XA Support

Read/write applications accessing Microsoft SQL Server data sources are able to execute transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see Appendix A, *XA Support*.

Configuring the Adapter for Microsoft SQL Server

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Microsoft SQL Server database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Microsoft SQL Server databases by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Microsoft SQL Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare SQL Server Connection Attributes From the Web Console on Windows

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	**Description**
Connection name	Logical name used to identify this particular set of connection attributes.
Server	Name of the machine where Microsoft SQL Server is running. If that machine has more than one instance of Microsoft SQL Server installed, provide the server name and the instance name as follows: server\instance.
Security	There are three methods by which a user can be authenticated when connecting to a Microsoft SQL Server: **Explicit.** The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication as a standard login. This option requires that SQL Server security be set to: SQL Server and Windows. **Password Passthru.** The user ID and password received from the client application are passed to Microsoft SQL Server, at connection time, for authentication as a standard login. This option requires that the server be started with security off and that SQL Server security be set to: SQL Server and Windows. **Trusted.** The adapter connects to Microsoft SQL Server as a Windows login using the credentials of the Windows user impersonated by the server data access agent. This option works with either of the SQL Server security settings.
User	Authorization ID by which the user is known at the instance of Microsoft SQL Server.

Attribute	Description
Password	Password associated with the authorization ID. The password is stored in encrypted form.
Default Database	Name of the default database for the connection. This value is used when a data object is not qualified with the database name. This parameter is optional. If not specified, it defaults to the database associated with the authorization ID.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually on Windows

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication.

```
ENGINE [SQLMSS] SET CONNECTION_ATTRIBUTES [connection]
server/userid,password [;dbname][:provider_string]
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication. This option requires that the server be started with security off.

```
ENGINE [SQLMSS] SET CONNECTION_ATTRIBUTES [connection]
server/[;dbname][:provider_string]
```

Trusted authentication. The adapter connects to Microsoft SQL Server as a Windows login using the credentials of the Windows user impersonated by the server data access agent.

```
ENGINE [SQLMSS] SET CONNECTION_ATTRIBUTES [connection]
  server/,[;dbname][:provider_string]
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is a logical name (or a data source name) used to identify this particular set of attributes.

If you plan to have only one connection to Microsoft SQL Server, this parameter is optional. If not specified, the local database server serves as the default connection.

`server`

Is the name of the machine where Microsoft SQL Server is running. If that machine has more than one instance of Microsoft SQL Server installed, provide the server name and instance as follows: server\instance.

When specifying the server name and the instance name, we recommend that you enclose the value in single quotation marks (‘).

`userid`

Is the primary authorization ID by which you are known to Microsoft SQL Server.

`password`

Is the password associated with the primary authorization ID.

`dbname`

Also referred to as schema, is the name of the Microsoft SQL Server database used for this connection. The database name, including path, must be enclosed in single quotation marks.

`provider_string`

Is the Microsoft SQL Server provider string used to specify additional connection options, such as the name of the network library. Note that this parameter must be preceded by a colon and enclosed in single quotation marks. This parameter is optional.

Note: Enclose values that contain special characters in single quotation marks. If a value contains a single quotation mark, this quotation mark must be preceded by another single quotation mark, resulting in two single quotation marks in succession. For example, to specify the user ID Mary O'Brien, which contains both a blank and a single quotation mark, enter: 'Mary O' 'Brian'.

Example: Declaring Connection Attributes on Windows

The following SET CONNECTION_ATTRIBUTES command allow the application to access the Microsoft SQL Server database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Microsoft SQL Server database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

The following SET CONNECTION_ATTRIBUTES command connects to a local Microsoft SQL Server database server using operating system authentication:

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES /,
```

Procedure: How to Declare Connection Attributes From the Web Console on UNIX, OS/390, and z/OS

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	Description
Connection name	Logical name used to identify this particular set of connection attributes.
URL	Enter the location URL for the Microsoft SQL Server data source.
Security	There are two methods by which a user can be authenticated when connecting to a Microsoft SQL Server: **Explicit.** The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication as a standard login. This option requires that SQL Server security be set to: SQL Server and UNIX. **Trusted.** The adapter connects to Microsoft SQL Server as a UNIX login using the credentials of the operating system user impersonated by the server data access agent. This option works with either of the SQL Server security settings.
User	Enter the primary authorization ID by which you are known to the target database.

Attribute	**Description**
Password	Enter the password associated with the primary authorization ID.
Driver name	Name for the Microsoft JDBC driver.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually on UNIX, OS/390, and z/OS

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES CON1 'URL'/userid,password
```

where:

`SQLMSS`

Indicates the Adapter for SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`CON1`

Is the connection name.

`URL`

Is the URL to the location of the data source.

`userid`

Is the primary authorization ID by which you are known to the target database.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes on UNIX, OS/390, and z/OS

The following SET CONNECTION_ATTRIBUTES command specifies using the Microsoft JDBC Driver.

```
ENGINE SQLMSS SET JDBCDRIVERNAME com.microsoft.jdbc.sqlserver.SQLServerDriver
```

The following SET CONNECTION_ATTRIBUTES command connects to a myServer via the Microsoft SQL Server JDBC Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES CON1
  'jdbc:microsoft:sqlserver://myServer:1433'MYUSER,PASS
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLMSS] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Microsoft SQL Server database server named SAMPLENAME as the default Microsoft SQL Server database server:

```
ENGINE SQLMSS SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLMSS] SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing Microsoft SQL Server Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft SQL Server data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Microsoft SQL Server table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value. See *Controlling the Mapping of Large Character Data Types* for more information.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.

Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLMSS [AT connection|AT '']
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLMSS

Indicates the Data Adapter for Microsoft SQL Server.

`AT connection`

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`AT ' '`

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local database server.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM baseapp/nf29004 FOR edaqa.nf29004 DBMS SQLMSS AT connmss
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLMSS ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4,     I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=edaqa.nf29004,
CONNECTION=connmss, KEYS=1, WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the Microsoft SQL Server table. The table name can be fully qualified as follows: `TABLENAME=[[`*`database.`*`]`*`owner.`*`]`*`table`*
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following table lists how the server maps Microsoft SQL Server data types. Note that you can:

- Control the mapping of large character data types.
- Control the mapping of variable-length data types.
- Enable National Language Support.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Microsoft SQL Server Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
CHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 8000
NCHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 4000
VARCHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 8000
NVARCHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 4000
TEXT	A32767	A32767	
NTEXT	TX50	TX	
BINARY(*n*)	A*m*	A*m*	*n* is an integer between 1 and 8000 *m* = 2 * *n*
VARBINARY(*n*)	A*m*	A*m*	*n* is an integer between 1 and 8000 *m* = 2 * *n*
SQL_VARIANT	A8000	A8000	
UNIQUEIDENTIFIER (GUID)	A38	A38	
TIMESTAMP	A16	A16	Supported as Read-only
DATETIME	HYYMDs	HYYMDs	range: 1/1/1753 to 12/31/9999
SMALLDATETIME	HYYMDI	HYYMDI	range: 1/1/1900 thru 6/6/2079
INT	I11	I4	range: -2^{31} to $2^{31} - 1$

Microsoft SQL Server Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
BIGINT	P20	P10	range: 2^{63} to 2^{63} - 1
SMALLINT	I6	I4	range: -2^{15} to 2^{15} - 1
TINYINT	I6	I4	range: 0 to 255
BIT	I11	I4	-1 for "True" and 0 for "False"
DECIMAL (p, s)	Pn.m	Pk	p is an integer between 1 and 38 s is an integer between 0 and p If s is 0 and p is between 1 and 31, n = p + 1 If s is 0 and p is between 32 and 38, n = 32 If s is greater than 0 and p is between 1 and 31, n = p + 2 and m = s If s is greater than 0 and p is between 32 and 38, n = 33 and m = 31 If p is between 1 and 31, k = (p / 2) + 1 If p is between 32 and 38, k = 16 **Note:** If the column is nullable, p is greater than or equal to 8.

Microsoft SQL Server Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
NUMERIC (p, s)	Pn.m	Pk	p is an integer between 1 and 38 s is an integer between 0 and p If s is 0 and p is between 1 and 31, n = p + 1 If s is 0 and p is between 32 and 38, n = 32 If s is greater than 0 and p is between 1 and 31, n = p + 2 and m = s If s is greater than 0 and p is between 32 and 38, n = 33 and m= 31 If p is between 1 and 31, k = (p / 2) + 1 If p is between 32 and 38, k = 16 **Note:** If the column is nullable, p is greater than or equal to 8.
MONEY	P21.4	P10	range: -2^{63} to $2^{63} - 1$
SMALLMONEY	P12.4	P8	range: -214,748.3648 to 214,748.3647
FLOAT	D20.2	D8	range: -1.79E+308 to 1.79E+308
REAL	D20.2	D8	range: -3.40E+38 to 3.40E+38
IMAGE	BLOB	BLOB	length: $2^{31} - 1$ Supported through iWay API

Controlling the Mapping of Large Character Data Types

How to:

Control the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Microsoft SQL Server data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Microsoft SQL Server Data Type	**Remarks**	**LONGCHAR ALPHA or BLOB**		**LONGCHAR TEXT**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
CHAR (*n*)	*n* is an integer between 1 and 256	A*n*	A*n*	A*n*	A*n*
	n is an integer between 257 and 8000	A*n*	A*n*	TX50	TX
NCHAR (*n*)	*n* is an integer between 1 and 128	A*n*	A*n*	A*n*	A*n*
	n is an integer between 129 and 4000	A*n*	A*n*	TX50	TX
VARCHAR (*n*)	*n* is an integer between 1 and 256	A*n*	A*n*	A*n*	A*n*
	n is an integer between 257 and 8000	A*n*	A*n*	TX50	TX
NVARCHAR (*n*)	*n* is an integer between 1 and 128	A*n*	A*n*	A*n*	A*n*
	n is an integer between 129 and 4000	A*n*	A*n*	TX50	TX
TEXT		A32767	A32767	TX50	TX

Microsoft SQL Server Data Type	Remarks	LONGCHAR ALPHA or BLOB		LONGCHAR TEXT	
		USAGE	ACTUAL	USAGE	ACTUAL
BINARY (*n*)	*n* is an integer between 1 and 8000 *m* = 2 * *n*	A*m*	A*m*	TX50	TX
VARBINARY (*n*)	*n* is an integer between 1 and 8000 *m* = 2 * *n*	A*m*	A*m*	TX50	TX
SQL_VARIANT		A8000	A8000	TX50	TX

Syntax: How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLMSS] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`ALPHA`

Maps the Microsoft SQL Server data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). ALPHA is the default value.

`TEXT`

Maps the Microsoft SQL Server data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

`BLOB`

For Windows is identical to ALPHA. That is, it maps the Microsoft SQL Server data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

Controlling the Mapping of Variable-Length Data Types

How to:

Control the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Microsoft SQL Server data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Microsoft SQL Server Data Type	**Remarks**	**VARCHAR ON**		**VARCHAR OFF**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
VARCHAR (*n*)	*n* is an integer between 1 and 8000	A*n*V	A*n*V	A*n*	A*n*
NVARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*V	A*n*V	A*n*	A*n*
VARBINARY (*n*)	*n* is an integer between 1 and 8000 *m* = 2 * *n*	A*m*V	A*m*V	A*m*	A*m*

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLMSS] SET VARCHAR {ON|OFF}
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Microsoft SQL Server data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (A*n*V).

`OFF`

Maps the Microsoft SQL Server data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). OFF is the default value.

Enabling National Language Support

How to:

Enable National Language Support

The SET parameter NCHAR indicates whether the character set is single-byte, double-byte, or triple-byte. The NCHAR setting affects the mapping of NCHAR and NVARCHAR data types.

The following chart lists data type mappings based on the value of NCHAR.

Microsoft SQL Server Data Type	**Remarks**	**NCHAR SBCS**		**NCHAR DBCS**		**NCHAR TBCS**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**	**USAGE**	**ACTUAL**
NCHAR (*n*)	*n* is an integer between 1 and 4000 *d* = 2 * *n* *t* = 3 * *n*	A*n*	A*n*	A*d*	A*d*	A*t*	A*t*
NVARCHAR (*n*)	*n* is an integer between 1 and 4000 *d* = 2 * *n* *t* = 3 * *n*	A*n*	A*n*	A*d*	A*d*	A*t*	A*t*

Syntax: How to Enable National Language Support

```
ENGINE [SQLMSS] SET NCHAR {SBCS|DBCS|TBCS}
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`SBCS`

Indicates a single-byte character set. SBCS is the default value.

`DBCS`

Indicates a double-byte character set.

`TBCS`

Indicates a triple-byte character set.

Changing the Precision and Scale of Numeric Columns

How to:

Override Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLMSS] SET CONVERSION RESET
ENGINE [SQLMSS] SET CONVERSION format RESET
ENGINE [SQLMSS] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLMSS] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`REAL` which indicates that the command applies only to single precision floating point columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT and REAL data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLMSS SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLMSS SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLMSS SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLMSS SET CONVERSION RESET
```

Support of Read-Only Fields

CREATE SYNONYM creates a field description with FIELDTYPE=R for Microsoft SQL Server columns created as TIMESTAMP or columns with the IDENTITY attribute. These fields are read-only. When executing a MAINTAIN or MODIFY procedure, the adapter suppresses all write operations against columns marked in the Master File with FIELDTYPE=R.

Example: Supporting a Read-Only Field

This example creates a table in which the first column has the IDENTITY property and the second column is a timestamp column:

```
CREATE TABLE TAB1
 (idproptab int IDENTITY (1,1), timstmp timestamp)
```

CREATE SYNONYM generates the following Master File for this table:

```
FILE=TAB1, SUFFIX=SQLMSS ,$
SEGNAME=TAB1, SEGTYPE=S0 ,$
FIELD=IDPROPTAB, idproptab, I11, I4,  MISSING=OFF, FIELDTYPE=R ,$
FIELD=TIMSTMP,   timstmp,   A16, A16, MISSING=ON,  FIELDTYPE=R ,$
```

Customizing the Microsoft SQL Server Environment

In this section:

The Adapter for Microsoft SQL Server provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Microsoft SQL Server.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE [SQLMSS] SET COMMANDTIMEOUT {nn|0}
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`nn`

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for response.

Note: If you do not specify a COMMANDTIMEOUT value, you default to the current Miscrosoft SQL Server default setting.

Specifying the Cursor Type

You can use the SET CURSORS command to specify the type of cursors for retrieval.

Syntax: How to Specify the Cursor Type

```
ENGINE [SQLMSS] SET CURSORS [CLIENT|SERVER]
```

where:

`CLIENT`

Uses Microsoft SQL Server client-side cursors for retrieving data. Client-side cursors normally demonstrate the best performance for data retrieval and benefit the Microsoft SQL Server process. However, except in TRANSACTIONS AUTOCOMMITTED mode, using client-side cursors prevents a server agent from simultaneously reading more than one answer set from the same instance of Microsoft SQL Server.

`SERVER`

Uses Microsoft SQL Server server-side cursors for retrieving data. Server-side cursors demonstrate lower performance than client cursors. However, setting a high FETCHSIZE factor (100 is the adapter default) improves their performance dramatically making them almost as fast as client-side cursors. Client-side cursors are recommended wherever possible in order to take the load off the Microsoft SQL Server process.

`blank`

Uses client-side cursors in TRANSACTIONS AUTOCOMMITTED mode and server-side cursors otherwise. This value is the default.

Specifying the Login Wait Time

You can use the LOGINTIMEOUT command to specify the number of seconds the adapter will wait for a response from Microsoft SQL Server at connect time. **Note:** For compatibility with previous releases of the adapter, TIMEOUT is available as a synonym for LOGINTIMEOUT.

Syntax: How to Specify the Login Wait Time

```
ENGINE [SQLMSS] SET LOGINTIMEOUT|TIMEOUT {nn|0}
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`nn`

Is the number of seconds before a timeout occurs. The default value is approximately 15 seconds.

`0`

Represents an infinite period to wait for login response.

Activating NONBLOCK Mode

The Adapter for Microsoft SQL Server has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE [SQLMSS] SET NONBLOCK {0|n}
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLMSS] SET PASSRECS {ON|OFF}
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Controlling Transactions

You can use the SET TRANSACTIONS command to control how the adapter handles transactions.

Syntax: How to Control Transactions

```
ENGINE [SQLMSS] SET TRANSACTIONS {LOCAL|DISTRIBUTED|AUTOCOMMITED}
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`LOCAL`

Indicates that the adapter implicitly starts a local transaction on each of the connections where any work is performed. At the time of COMMIT or ROLLBACK, or at the end of the server session, the adapter commits or aborts the work on each connection consecutively. LOCAL is the default value.

`DISTRIBUTED`

Indicates that the adapter implicitly invokes Microsoft Distributed Transactions Coordinator (DTC) to create a single distributed transaction within which to perform all work on all the connections. At the time of COMMIT or ROLLBACK, or at the end of the server session, the adapter invokes DTC to execute the two-phase commit or rollback protocol. For this purpose, the DTC service must be started on the machine where the server is running and also on all the machines where involved instances of Microsoft SQL Server reside.

This mode is recommended for read-write applications that perform updates on multiple connections simultaneously.

`AUTOCOMITTED`

Indicates that each individual operation with Microsoft SQL Server is immediately committed (if successful) or rolled back (in case of errors) by the SQL Server. This is recommended for read-only applications for performance considerations. It is not recommended for read-write applications because in this mode it is impossible to roll back a logical unit of work that consists of several operations.

Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLMSS] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLMSS`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

OFF instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

ON instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

FOCUS passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

SQL passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLMSS set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLMSS set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLMSS] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLMSS`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLMSS SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLMSS SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLMSS SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Specifying Block Size for Retrieval Processing

The Adapter for Microsoft SQL Server supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Specify Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE [SQLMSS] SET FETCHSIZE n
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Syntax: How to Specify Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE [SQLMSS] SET INSERTSIZE n
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Syntax: How to Suppress the Bulk Insert API

```
ENGINE [SQLMSS] SET FASTLOAD [ON|OFF]
```

where:

`SQLMSS`

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Uses the Bulk Insert API. ON is the default.

`OFF`

Suppresses the use of the Bulk Insert API.

Reference: Bulk Insert API Behavior

You can use DataMigrator with the Bulk Insert API for Microsoft SQL Server.

For the Adapter for Microsoft SQL Server, the Bulk API is used automatically in LOADONLY mode. Measurements show that intermediate flushes do not affect performance; therefore, the behavior does not depend on the INSERTSIZE.

Errors that occur during the load (such as duplication) can cause the batch of rows to be rejected as a whole.

Calling a Microsoft SQL Server Stored Procedure Using SQL Passthru

Microsoft SQL Server stored procedures are supported using SQL Passthru. These procedures need to be developed within Microsoft SQL Server using the CREATE PROCEDURE command.

The adapter supports stored procedures with input, output, and in-out parameters.

The output parameter values that are returned by stored procedures are available as result sets. These values form a single-row result set that is transferred to the client after all other result sets returned by the invoked stored procedure. The names of the output parameters (if available) become the column titles of that result set.

Note that only the output parameters (and the returned value) referenced in the invocation string are returned to the client. As a result, users have full control over which output parameters have their values displayed.

The server supports invocation of stored procedures written according to rules that are specific to each adapter.

Syntax: How to Invoke a Stored Procedure

```
SQL [SQLMSS]EX procname [parameter_specification1]
[,parameter_specification2]...
END
```

where:

`SQLMSS`

Is the ENGINE suffix.

`procname`

Is the name of the stored procedure. It is the fully or partially qualified name of the stored procedure in the native RDBMS syntax.

`parameter_specification`

Input, output, and in-out parameters are supported. Use the variation required by the stored procedure:

`input`

Is a literal (for example, 125, 3.14, 'abcde'). You can use reserved words as input. Unlike character literals, reserved words are not enclosed in quotation marks (for example, NULL). Input is required.

`output`

Is represented as a question mark (?). You can control whether output is passed to an application by including or omitting this parameter. If omitted, this entry will be an empty string (containing 0 characters).

`in-out`

Consists of a question mark (?) for output and a literal for input, separated by a slash: /. (For example: ?/125, ?/3.14, ?/'abcde'.) The out value can be an empty string (containing 0 characters).

Example: Invoking a Stored Procedure

In this example, a user invokes a stored procedure, edaqa.test_proc01, supplies input values for parameters 1, 3, 5 and 7, and requests the returned value of the stored procedure, as well as output values for parameters 2 and 3.

Note that parameters 4 and 6 are omitted; the stored procedure will use their default values, as specified at the time of its creation.

```
SQL SQLMSS EX edaqa.test_proc01 125,?,?/3.14,,'abc',,'xyz'
END
```

Example: Sample Stored Procedure

This stored procedure uses out and inout parameters:

```
CREATE PROCEDURE EDAQA.PROCP3 (   OUT chSQLSTATE_OUT   CHAR(5),
                                  OUT intSQLCODE_OUT   INT,
                                  INOUT l_name char(20),
                                  INOUT f_name char(20))
    RESULT SETS 1
    LANGUAGE SQL
------------------------------------------------------------------------
-- SQL Stored Procedure
------------------------------------------------------------------------
P1: BEGIN

    -- Declare variable
    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
    DECLARE SQLCODE INT DEFAULT 0;

    -- Declare cursor
    DECLARE cursor1 CURSOR WITH RETURN FOR

        SELECT
           EDAQA.NF29005.SSN5 AS SSN5,
           EDAQA.NF29005.LAST_NAME5 AS LAST_NAME5,
           EDAQA.NF29005.FIRST_NAME5 AS FIRST_NAME5,
           EDAQA.NF29005.BIRTHDATE5 AS BIRTHDATE5,
           EDAQA.NF29005.SEX5 AS SEX5
        FROM
           EDAQA.NF29005
        WHERE
           (
             ( EDAQA.NF29005.LAST_NAME5 =  l_name )
        AND
             ( EDAQA.NF29005.FIRST_NAME5 = f_name )
           );

     -- Cursor left open for client application
    OPEN cursor1;

    SET chSQLSTATE_OUT = SQLSTATE;
    SET intSQLCODE_OUT = SQLCODE;
    SET l_name = 'this is first name';
    SET f_name = 'this is last name';

END P1   @
```

Reference: Capturing Application Errors in Stored Procedures

You can capture application errors using the RAISERROR method. Any application error that is issued by the stored procedure is available in the server variable &MSSMSGTXT.

Microsoft SQL Server Compatibility With ODBC

In this section:

Release 5.2 of the Adapter for Microsoft SQL Server is based on OLE DB, the Microsoft-recommended API for developing high-performance components. The adapter uses the OLE DB Provider for SQL Server (SQLOLEDB), a native, high-performance provider that directly accesses the SQL Server TDS protocol.

Earlier releases of the adapter were based on the ODBC API. Release 5.2 of the adapter supports all the functionality of the earlier, ODBC-based versions. In addition, it introduces the following enhancements:

- Support for distributed transactions.
- Array Retrieval.
- Command execution timeout.
- Fast load. This feature, enabled automatically, improves the performance of load-only DataMigrator applications and MODIFY procedures.
- XA transaction support. For details, see Appendix A, *XA Support*.

Differences between adapter functionality under OLE DB and ODBC exist in the following areas:

- Connection attributes.
- Cursors.
- Mapping of UNIQUEIDENTIFIER and BIT data types.

If you are using ODBC, be sure to review these topics.

Microsoft SQL Server Connection Attributes With ODBC

An ODBC data source declaration contains the following information: the instance of Microsoft SQL Server, the default database, the network libraries, and so on. In earlier, ODBC-based releases of the adapter, the SET CONNECTION_ATTRIBUTES command required only the name of an ODBC data source (User or System DNS) and authentication parameters.

Unlike ODBC data sources, OLE DB data sources directly reference instances of Microsoft SQL Server. To connect to an OLE DB data source, the adapter has to specify all the pertinent parameters at connection time. For this reason, the syntax of the SET CONNECTION_ATTRIBUTES command has been expanded to account for these parameters and also to provide the ability to declare more than one connection to the same instance of Microsoft SQL Server.

Microsoft SQL Server Cursors With ODBC

In earlier, ODBC-based releases of the adapter, the SET parameter CONCUR is used to control the ODBC cursor type. The CONCUR parameter has two settings: RONLY results in client-side cursors, and ROWVER forces cursors to be server-side. Release 5.2 of the adapter takes advantage of OLE DB's ability to directly control the cursor type using a CURSOR parameter. The SET parameter CURSOR has two settings: CLIENT and SERVER. By default, the adapter assigns cursor properties that provide maximum performance for the given environment.

Mapping of UNIQUEIDENTIFIER and BIT Data Types

OLE DB maps the following data types differently than ODBC:

- UNIQUEIDENTIFER is mapped to 38 characters. ODBC maps this data type to 36 characters. This difference results from the fact that OLE DB encloses the value in braces ({ }).
- BIT is mapped to -1 for TRUE and 0 for FALSE. ODBC maps TRUE to 1.

CHAPTER 24

Using the Adapter for Millennium

Topics:

The Adapter for Millennium allows applications to access Millennium data sources. The adapter converts application requests into native Millennium statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the Server Environment for Millennium

The Adapter for Millennium supports compressed Millennium VSAM files. The Millennium decompression routines have been added through the existing Zcomp1 exit point. To enable support for compressed files, include the following set command is in your server profile, EDASPROF.prf:

```
ENGINE CPMILL SET ZCOMP FOCMILZ
```

The adapter requires two control files to be built and allocated to DDNAME CPMILL and CPMILLI via JCL or DYNAM allocation. The Millennium control file is used as input to these files and is dynamically allocated to the server only for the creation of CPMILL and CPMILLI.

Procedure: How to Prepare the Server Environment for Millennium

Allocate two MVS PDSs, CPMILL and CPMILLI, with no DCB information—DCB information is determined by the server at runtime. The space requirements are:

- First extent cylinders: 5
- Secondary cylinders: 2

Sample JCL for CPMILL allocation follows:

```
//ALOCLIB EXEC PGM=IEFBR14
//APPL   DD DSN=HIGHLQ.CPMILL,DISP=(NEW,CATLG),
//       VOL=SER=USERMC,UNIT=3390,SPACE=(CYL,(5,2))
```

Sample JCL for CPMILLI allocation follows:

```
//ALOCLIB EXEC PGM=IEFBR14
//APPL   DD DSN=HIGHLQ.CPMILLI,DISP=(NEW,CATLG),
//       VOL=SER=USERMC,UNIT=3390,SPACE=(CYL,(5,2))
```

Configuring the Adapter for Millennium

You can use the Web Console to configure the adapter. You must then edit and execute several files to complete the configuration.

Procedure: How to Configure the Adapter for Millennium From the Web Console

To configure the adapter for Millennium from the Web Console:

1. Start the Web Console and select *Data Adapters* from the left pane.
2. Expand the *Add* folder, expand the *Other DBMS* group folder, then expand the *Millennium* folder and click a connection. The Add Millennium to Configuration pane opens.
3. No input parameters are required. Simply, click *Configure*. The adapter is added to the Configured adapters folder.

Procedure: How to Edit Configuration Files for the Adapter for Millennium

After configuring the adapter from the Web Console, perform the following additional steps:

1. Edit the server profile, EDASPROF, to:
   - Allocate CPMILL and CPMILLI to the server.
   - Add your GEAC VSAM files, as in the following sample.

```
-***************************************
  ENGINE CPMILL SET ZCOMP FOCMILZ
-************* GEAC Allocations*********
   DYNAM ALLOC FILE CPMILL  MOD -
    DATASET HIGHLQ.CPMILL
   DYNAM ALLOC FILE CPMILLI MOD -
    DATASET HIGHLQ.CPMILLI
   DYNAM ALLOC FILE HRMH0B  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33EM1
   DYNAM ALLOC FILE H33EM2  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33EM2
   DYNAM ALLOC FILE H33EM3  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33EM3
   DYNAM ALLOC FILE HRMH7O  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33PER
   DYNAM ALLOC FILE HRMH7S   SHR REU -
    DATASET MKTPJC.GEAC.HR.H33PER
   DYNAM ALLOC FILE H33QEH1  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33QEH1
   DYNAM ALLOC FILE H33QEH2  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33QEH2
   DYNAM ALLOC FILE H33QEH3  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33QEH3
   DYNAM ALLOC FILE H33TAX   SHR REU -
    DATASET MKTPJC.GEAC.HR.H33TAX
   DYNAM ALLOC FILE H33UTL   SHR REU -
```

2. Start the server.

3. Using RDAAPP on MVS or the iWay Test Tool on NT, execute the CPMILLI RPC, located in the IBIFEX DDNAME allocation of the server. This procedure uses the Millennium control file as input to populate the CPMILLI control file.

   To execute the RPC, you need the dataset name of the Millennium control file (for example, M3000).

```
CPMILLI DSN=GEAC.MILL.M30000
```

After execution, CPMILLI will contain the following records:

```
CDBAMDM3LL______
CDBAMIM3LL______
CDBAMNM3LL______
CDBH0BM3LL______
CDBH7OM3LL______
CDBH7SM3LL______
```

4. Edit CPMILLI to associate the Millennium DB_DBIDs, DB_DBSUBIDs, and DB_TRANIDs with the names of the corresponding Master Files, which are delivered with the Millennium product.

CPMILLI Record	Master File Name	Edited Result
CDBAMDM3LL	APMAMD	CDBAMDM3LLAPMAMD
CDBAMIM3LL	APMAMI	CDBAMIM3LLAPMAMI
CDBAMNM3LL	APMAMN	CDBAMNM3LLAPMAMN
CDBH0BM3LL	HRMH0B	CDBH0BM3LLHRMH0B
CDBH7OM3LL	HRMH0B	CDBH7OM3LLHRMH7O
CDBH7SM3LL	HRMH0B	CDBH7SM3LLHRMH7S

5. Using RDAAP on MVS or the iWay Test Tool on NT, execute the CPMILL RPC, located in the IBIFEX DDNAME allocation of the server. This procedure associates the Millennium control file information with the adapter routines.

   CPMILL takes two input parameters: &dsn for the CPMILLI dataset, and &dsn1 for the Millennium control file.

```
CPMILL DSN=HIGHLQ.CPMILLI,DSN1=GEAC.MILL.M30000
```

6. Edit the server profile, EDASPROF, to change the file allocation for CPMILL and CPMILLI from MOD to SHR REU, as shown below:

```
-**************************************
  ENGINE CPMILL SET ZCOMP FOCMILZ
-************* GEAC Allocations*********
   DYNAM ALLOC FILE CPMILL  SHR REU -
    DATASET HIGHLQ.CPMILL
   DYNAM ALLOC FILE CPMILLI SHR REU -
    DATASET HIGHLQ.CPMILLI
```

Preparing the Millennium Environment

In this section:

Adapter Tracing

How to:

Edit the Server's IRUNJCL

Prior to using the server and configuring the Adapter for Millennium using the Web Console, you need to edit the server's IRUNJCL procedure.

Procedure: How to Edit the Server's IRUNJCL

Procedure overview:

1. Concatenate the Millennium API modules for the adapter to DDNAME STEPLIB.
2. Allocate the DDNAMEs M30000 and M30002 to your site's Millennium Control File data sets, if you are running Millennium Release 3.
3. Determine the DDNAMEs for the Millennium databases based on your site's Millennium installation options. (See your Millennium documentation or contact your Millennium administrator for this information.)
4. Allocate DDNAME FOCPRV to the data set that contains Access Files. You can omit the allocation for DDNAME FOCPRV if all database IDs are assigned using the Master File member name (for more information, see *Managing Millennium Metadata* on page 24-7), or access your site's default lead transaction ID.

   Your system support staff can supply proper data set names for your site. The Millennium API modules must be MVS-loadable.

The server must allocate the adapter's specific files in the IRUNJCL, as well as in a global profile (EDASPROF.CFG).

Edit the IRUNJCL and EDASPROF.CFG as follows:

IRUNJCL. Millennium load library and Control files and databases

```
//M30000 DD DISP=SHR,DSN=millen.ctrlfile.M30000
//M30002 DD DISP=SHR,DSN=millen.ctrlfile.M30002
//database DD DISP=SHR,DSN=millen.database
```

The IRUNJCL must have the following data sets allocated to the following DDNAMEs.

EDASPROF.CFG. Millennium Access File data set allocation

```
//* Millennium ALLOCATIONS
DYNAM ALLOC F FOCPRV DA qualif.FOCPRV.DATA SHR REUSE
```

where:

millen

Is the high-level qualifier for your site's Millennium production data sets.

ctrlfile

Is the Millennium control file.

database

Is the DDNAME required by Millennium for a Millennium database. If you access more than one Millennium database, you must allocate more than one DDNAME, each determined by the options chosen when your site installed Millennium.

qualif

Is the high-level qualifier for your site's server production data sets.

Adapter Tracing

To activate the Adapter for Millennium Tracing Facility, you must allocate DDNAME GPTRACE in the server JCL:

```
//GPTRACE DD SYSOUT=*
```

Note: Use the adapter Tracing Facility carefully. The DDNAME is allocated to the server's JCL; therefore, it is applied globally. This causes the adapter’s activity to be traced for all server tasks. This facility should be used under tightly controlled testing circumstances.

Managing Millennium Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Millennium data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Millennium file or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File that represents the server's metadata.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The Select Synonym Candidates pane (Step 1 of 2) opens.
4. Under File System Selection, choose one of the following options from the drop-down list:
   - *Fully qualified PDS name* to indicate a partitioned dataset on MVS.

     In the input boxes provided, type a PDS name preceded by // and a Member name containing the location of the COBOL FD source. If you wish, you can filter the member name using a wildcard character (%).

     or
   - *Absolute HFS directory pathname* to indicate a hierarchical file structure on USS.

     In the input boxes provided, type a Directory name to specify the HFS location that contains the COBOL FD and a File name and File extension. If you wish, you can filter the file and extension using a wildcard character (%).
5. Click *Submit*. The Create Synonym pane (Step 2 of 2) opens.

   If selected, the filtered COBOL definition is displayed at the top of the pane.
6. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

   When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.
7. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.
8. Optionally, select *Customize options* to customize how the COBOL FD is translated. For details about these options, see *Customization Options for COBOL File Descriptions* on page 24-11. If you do not select the check box, default translation settings are applied.
9. Enter the following additional parameters as appropriate:

Parameter	**Description**
Select Application	Select an application directory. baseapp is the default value.

Parameter	**Description**
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

10. Complete your selection:

- To select all synonyms in the list, select the check box to the left of the *Default Synonym Name* column heading.
- Enter a cluster name to associate it with a particular metadata description. The cluster name is embedded in the Master File. If you do not enter the cluster name during the synonym creation process, you will have to add it dynamically at run time.
- To select specific synonyms, select the corresponding check boxes.

11. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

12. Click *Create Synonym*.

Synonyms are created and added under the specified application directory.

A status window displays the message:

```
All Synonyms Created Successfully
```

13. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: Customization Options for COBOL File Descriptions

You can customize how a COBOL FD is translated by selecting *Customize options* in the Synonym creation pane. The following options are added to the right pane:

Parameter	**Definition**
On Error	Choose *Continue* to continue generating the Master File when an error occurs. Choose *Abort* to stop generating the Master File when an error occurs. Choose *Comment* to produce a commented Master File when an error occurs. Continue is the default value.
Hyphens as	Choose *No* to remove all hyphens in the COBOL name from the Master File field names. Choose *Yes* to replace all hyphens in the COBOL name with the underscore character. *Yes* is the default value.
Redefines	You may treat COBOL REDEFINE fields in one of three ways: Choose *Segments* to describe REDEFINE fields as segments in the Master File. Segments is the default value. Choose *Comments* to describe REDEFINE fields as comments in the Master File. Choose *None* to exclude REDEFINE fields altogether.
Occurs as	Choose *Segments* to describe OCCURS structures as segments, or *Field* otherwise. Segments is the default value.
Alignment	Choose *Yes* to insert slack bytes into a record to ensure alignment of numeric fields. Choose *No* to generate Master Files without alignment of slack bytes. No is the default value.

Parameter	**Definition**
Number of Hyphens to skip	FD Translator removes characters from the left, up to and including the Nth hyphen (depending on the value of N chosen in the menu). A value of *0* will retain the entire COBOL name. *All* means all prefixes will be removed. *0* is the default value.
Order fields	Choose *Yes* to generate Order fields in a Master File. Choose *No* to generate a Master File without Order fields. *No* is the default value.
Level 88 as	Choose *Comment* to include COBOL Level 88 fields as comments in the Master Files. Choose *Skip* to exclude level 88 fields. *Skip* is the default value.
Zoned Numeric Fields	Sets how zoned numeric values will be stored.
Numeric Fields Edit Options	
Zeroes	Choose *Suppress* to suppress printing of the digit zero for a field whose value is zero. Choose *Display* to display leading zeroes, for example, 00124. Choose *None* for no formatting.
Negative value	Choose *Bracket* to bracket negative values, for example, (1234). Choose *Credit* to credit negative values, for example, 1234 CR. Choose *None* for no formatting.
Dollar Sign	Choose *Floating* to display a floating dollar sign and commas, for example, $1,1234. Choose *Fixed* to display a fixed dollar sign and commas, for example, $ 1,1234. Choose *None* for no formatting.

Parameter	**Definition**
Separate Thousands	Choose *Comma* to include commas where appropriate. Choose *None* for no formatting.

For additional information about customization options, see Appendix D, *Translating COBOL File Descriptions*.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '| ', '&', ' (', ') ', ' <', '> ', '' ', ' =', ''''

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM baseapp/synonym FOR cobol_fd DBMS CPMILL
DATABASE 'cpmill_dataset_name'
[CHECKNAMES][UNIQUENAMES]
END
```

where:

synonym

Is the name to be assigned the resulting synonym.

cobol_fd

Is the HFS location or MVS PDS, including the member name containing the COBOL FD source.

vsam_dataset_name l

Is the MVS name of the CPMILL cluster.

CHECKNAMES

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

UNIQUENAMES

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

Example: Sample COBOL FD

```
 01 COUNTRY-REC.
    02 G1.
       03  COUNTRY_COD1   PIC X(5).
       03  FILLER         PIC X(3).
     02  COUNTRY_NAM1   PIC X(15).
     02  FILLER         PIC X(1).
```

Example: Using CREATE SYNONYM for a Millennium Data Source

```
CREATE SYNONYM baseapp/CPMILL2901
FOR /pgm/edaport/R729999B/tscq/nf2901.cbl
DBMS CPMILL
DATABASE 'EDAQA.SQLTRANS.NF29001.CPMILL'
END
```

Synonym:

```
FILENAME=CPMILL2901, SUFFIX=CPMILL    ,
 DATASET=EDAQA.SQLTRANS.NF29001.CPMILL, $
  SEGMENT=SEG1, SEGTYPE=S0, $
$  GROUP=COUNTRYREC, USAGE=A24, ACTUAL=A24, $
   GROUP=G1, ALIAS=KEY, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=COUNTRY_COD1, USAGE=A5, ACTUAL=A5, $
    FIELDNAME=FILLER, USAGE=A3, ACTUAL=A3, $
    FIELDNAME=COUNTRY_NAM1, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=FILLER, USAGE=A1, ACTUAL=A1, $
```

Standard Master File Attributes for a Millennium Data Source

For each Master File, the server must access the appropriate Millennium database ID (DBID). There are two methods for assigning the DBID to a Master File (see *Specifying the Millennium DBID and TRANID* on page 24-19). Depending on your assignment method and on your Millennium lead transaction ID, you may need an Access File in addition to the Master File:

- Master Files describe the fields in Millennium files (see *Master Files* on page 24-16). Master Files for Millennium file are the same as Master Files for VSAM files, except that the SUFFIX value for a Millennium file must be CPMILL for Millennium Release 3.

  The Master File must conform to the COBOL copybook for the file. If your site has the COBOL FD Translator feature installed, you can use it to help create a Master File. You can then edit this Master File (to change the SUFFIX value) with any text editor.

- Access Files enable you to use a lead transaction ID (TRANID) other than the default that your site established during installation (see *Specifying the Millennium DBID and TRANID* on page 24-19). An Access File can also assign the DBID for its corresponding Master File.

When you issue a report request, the server processes the request with the following steps:

1. It locates the Master File specified by the request.
2. It examines the SUFFIX attribute in the Master File. Each Master File that describes a Millennium database must include the attribute SUFFIX=CPMILL for Millennium Release 3. When the server detects this SUFFIX, it passes control to the adapter.
3. It examines the Master File member name. If an Access File PDS has been allocated, and if it has a member with the same name as the Master File, the adapter locates that Access File.

   The adapter uses the information contained in both the Master File and the Access File (if there is one) to generate the Millennium calls required by the report request. It passes these calls to Millennium.
4. The adapter retrieves the data generated by the Millennium DBMS and returns control to the server. For some requests, the server may perform additional processing on the returned data.

Master Files

The following partial Master File illustrates how to describe a Millennium file. The numbers to the left refer to the explanatory notes that follow the sample:

```
1. FILENAME=ACCTMAST, SUFFIX=CPMILL [,$]
2. SEGNAME=ROOT,SEGTYPE=S0,$
3. FIELDNAME=DELETE_FLAG    ,ALIAS=     ,USAGE=A1  ,ACTUAL=A1  ,$
4. GROUP=CONTRL_KEY         ,ALIAS=KEY  ,USAGE=A23 ,ACTUAL=A23 ,$
      FIELDNAME=CORP        ,ALIAS=     ,USAGE=A3  ,ACTUAL=A3  ,$
      FIELDNAME=ACCOUNT     ,ALIAS=     ,USAGE=A10 ,ACTUAL=A10 ,$
      FIELDNAME=COST_CENTR  ,ALIAS=     ,USAGE=A10 ,ACTUAL=A10 ,$
```

Note:

1. Each Master File begins with a file declaration that names the file and describes the type of data source-a Millennium file in this case. The file declaration has two attributes, FILENAME and SUFFIX.

   The FILENAME can be any one- to eight-character name that complies with server naming conventions. For documentation purposes, you can give it the same name as the Master File member name.

   SUFFIX=CPMILL indicates that Millennium Release 3 is required for data retrieval.

2. Each type of Millennium record described in a Master File requires a segment declaration consisting of at least two attributes, SEGNAME and SEGTYPE. The SEGNAME value is ROOT, and the SEGTYPE value is S0 (S zero).
3. To describe a Millennium field in the Master File, you must include a FIELD record (or, for a key field, a GROUP record) that specifies the attributes FIELDNAME, ALIAS, USAGE, and ACTUAL:
   - The FIELDNAME value is the server name for the field. Since field names appear as default column titles on reports, select names that are representative of the data. You can specify field names, aliases, or a unique truncation of either in requests.
   - The ALIAS value in the Master File is an optional alternate name for the field, except if the field is a key field.
   - The USAGE format describes how the server will display the value in reports.
   - The ACTUAL format describes how the data field exists in storage. It must conform to the COBOL FD statement for the field. See ACTUAL Format Conversion Chart for how to determine ACTUAL formats.
4. You must describe the primary key field with the GROUP attribute instead of the FIELDNAME attribute, and you must assign its alias as ALIAS=KEY.

   If the key field is composed of multiple elementary fields, describe the elementary fields directly below the GROUP record. The USAGE format of the GROUP must be alphanumeric; its length is the sum of the server internal storage lengths for the individual fields. The ACTUAL format of the GROUP must also be alphanumeric; its length is the sum of the subordinate field lengths.

ACTUAL Format Conversion Chart

The ACTUAL attribute indicates the server representation of Millennium field formats. Use the following chart as a guide for describing ACTUAL field formats:

COBOL FORMAT	**COBOL PICTURE**	**FOCUS INTERNAL STORAGE**	**ACTUAL FORMAT**	**USAGE FORMAT**
DISPLAY	X(4)	4	A4	A4
DISPLAY	S99	2	Z2	P3
DISPLAY	9(5)V9	6	Z6.1	P8.1
DISPLAY	99	2	A2	A2
COMP	S9	4	I2	I2
COMP	S9(4)	4	I2	I4
COMP	S9(5)	4	I4	I5
COMP	S9(9)	4	I4	I9
COMP-1	-	4	F4	F6
COMP-2	-	8	D8	D15
COMP-3	9	8	P1	P1
COMP-3	S9V99	8	P2	P5.2
COMP-3	9(4)V9(3)	8	P4	P8.3

Note: The USAGE lengths shown are minimum values. You can make them larger and add edit options. You must allow space for all possible digits, a minus sign for negative numbers, and a decimal point in numbers with decimal digits.

Specifying the Millennium DBID and TRANID

You must assign a Millennium database ID (DBID) to each Master File. You can use either of the following techniques to assign the DBID:

- Include the DBID as the last three characters of the Master File member name. For example, if you store the Master File in member ANYGLM, its assigned DBID is GLM (General Ledger).
- Assign the DBID in an Access File. The Access File can also assign the lead transaction ID (TRANID) for its associated Master File.

Create an Access File if:

- The Master File member name does not include a DBID assignment; that is, the last three characters of the Master File member name do not identify a valid DBID.
- The Master File member name assigns a DBID, but the DBID it assigns is not the one you need to access.
- You need a lead transaction ID other than the default (your site establishes the default lead transaction ID during installation).

The Master File member name must either include the appropriate DBID assignment, or it must identify an Access File, or both.

Syntax: How to Specify the Master File Member Name

```
{xxxxxdbd|anyname}
```

where:

xxxxx

Is a name, which can be up to five characters long, that conforms to file-naming conventions.

dbd

Is a three-character DBID. The adapter accesses this DBID unless an Access File member named xxxxxdbd exists and assigns a different DBID. When both the Master File member name and the Access File assign the DBID, the DBID from the Access File takes precedence.

anyname

Is the name of a member in the Access File data set that contains the DBID and/or lead transaction ID to use.

As an example, consider the Master File in *Master Files* on page 24-16. It is member ACCTMAST in the Master File PDS, indicating that member ACCTMAST in the Access File data set contains the required DBID and/or TRANID.

Access Files

If an Access File is required, the server JCL must allocate the PDS containing Access File members to DDNAME FOCPRV.

```
[DBID=dbd], [TRANID={tran|M3LL}]  ,$
```

where:

dbd

Is the three-character Millennium DBID associated with this file. You can omit this parameter if you included the appropriate DBID as the last three characters of the Master File member name.

tran

Is the optional Multi:Mill lead transaction ID. M3LL is the default lead transaction ID for Millennium Release 3 unless your site chooses a different default value during installation. For more information, check with your systems support staff.

Note: Every Master File named in a single request (such as a join of two databases) must use the same TRANID.

The following is a sample Access File:

```
DBID=GLM,$
```

This Access File points to:

- A General Ledger file (DBID=GLM).
- The default lead transaction ID, M3LL for Millennium Release 3 (unless the site chose a different default during installation).

CHAPTER 25

Using the Adapter for MODEL 204

Topics:

The Adapter for MODEL 204 allows applications to access MODEL 204 data sources. The adapter converts application requests into native MODEL 204 statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the MODEL 204 Environment

Before you install the adapter, review the following list of software requirements:

- MODEL 204 must be installed and working. If it is not, contact your MODEL 204 database administrator. The adapter may be used with MODEL 204 Version 2.2.0 and higher.
- The server must be installed on your system. If it is not, contact your server administrator or consult the appropriate server installation guide.

Configuring the Adapter for MODEL 204

In this section:

How to:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. The Adapter for MODEL 204 is configured from the Web Console.

Procedure: How to Configure the Adapter

To configure the Adapter for MODEL 204:

1. Edit the ISTART JCL:
   - Allocate the MODEL 204 load library to ddname STEPLIB.
   - Any private Master File libraries and *prefix*.EDAMFD.DATA should be allocated to ddname MASTER in the JCL.
   - Any private Access File libraries and *prefix*.EDAAFD.DATA should be allocated to ddname ACCESS in the JCL.
2. From the Web Console Adapter Add screen, click *Configure*.

Specifying Account and File Passwords

Your MODEL 204 database administrator must provide account and file security information in order for the adapter to access password protected MODEL 204 files or groups.

An account consists of the MODEL 204 user ID and password. You can specify these security values by either including the ACCOUNT and ACCOUNTPASS attributes in the first declaration of the Access File, or by issuing the adapter M204IN SET M204ACCNT and M204PASS commands. Your database administrator can also provide MODEL 204 logon and account information with a security exit subroutine.

You must also know the file name and password (if there is one) for each MODEL 204 file or group you access. Specify these values with the FILE and PASS attributes in each SEGNAME declaration of the Access File.

For more information, see *Customizing the MODEL 204 Environment* on page 25-31.

Testing the Adapter Installation

To verify the adapter installation, run a simple query using one of the pairs of Master and Access Files you have defined for a MODEL 204 file.

MODEL 204 Overview and Mapping Considerations

In this section:

A MODEL 204 file can represent one file or a collection of files called a group. A single MODEL 204 file can contain records that vary in length and format; a record can consist of fields that also vary in length. Fields are the smallest elements or units of data. They can occur more than once per record. A unit of records with identical sets of fields is called a logical record type. A MODEL 204 file is defined in a MODEL 204 file description.

In the sections that follow, you will notice that:

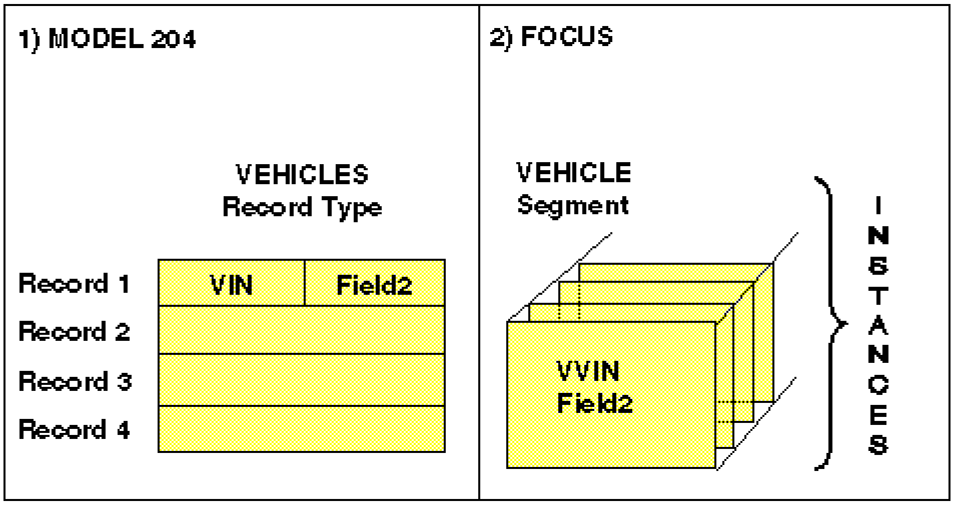
- A MODEL 204 field is equivalent to a server Master File field.
- A logical record type corresponds to a server Master File segment. The individual record corresponds to a segment instance.
- MODEL 204 fields that appear more than once in a record map to a server Master File OCCURS segment.
- One or more record types can be described in any order in a pair of Master and Access Files.
- One or more MODEL 204 files can be described in a pair of Master and Access Files.

Files With One Logical Record Type

A MODEL 204 file that contains records with identical sets of fields (the same logical record type) can be represented as a single Master File segment. Each MODEL 204 field that occurs only once becomes a Master File field. The ALIAS attribute in the Master File identifies the MODEL 204 field name.

In the Master File, describe the logical record type as a segment; the segment description can include some or all of the MODEL 204 fields in any order. The corresponding Access File identifies the record type's MODEL 204 file or group name and its password; the Access File can also include FIELD declarations that specify a TYPE attribute to indicate the appropriate suffix operators for MODEL 204 key fields. Master File field suffix operators (for example, KEY) identify different types of MODEL 204 key fields.

Note: A MODEL 204 field that can occur more than once should be represented as a separate OCCURS segment.



1. Is one MODEL 204 file with one logical record type.
2. Is the equivalent server Master File segment. The Master File field VVIN represents the MODEL 204 field VIN.

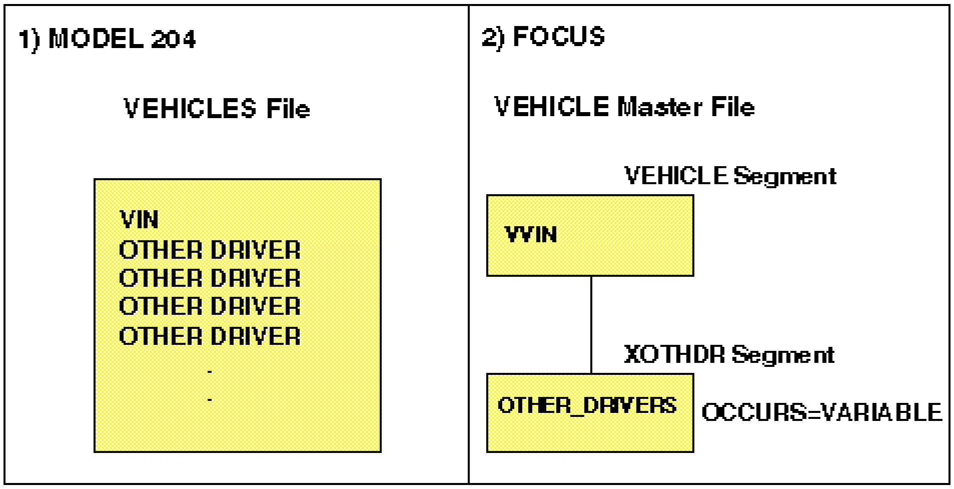
Files With Multiply Occurring Fields

In MODEL 204 files, some fields may appear more than once per record. Represent each multiply occurring field separately from the non-repeating fields as a server Master File OCCURS segment. In this case, the Master File contains a parent segment to describe all the non-repeating fields, and a separate dependent OCCURS segment for each multiply occurring field.

When you create a Master File, describe the OCCURS segment only if you need the multiply occurring field for your reports; you are not required to describe every multiply occurring field. Do not describe the OCCURS segment in the corresponding Access File unless its size exceeds the buffer capacity, a situation that is explained in *Customizing the MODEL 204 Environment* on page 25-31.

Describing Files to the Server

The following diagram illustrates one multiply occurring field and its equivalent OCCURS segment:



1. Is the MODEL 204 VEHICLES file with one logical record type and one multiply occurring field called OTHER DRIVER that identifies drivers besides the principal driver.

2. Is the equivalent multi-segment server structure. The parent (or root, in this case) is the VEHICLE segment; it contains all of the non-repeating fields. The dependent segment is the OCCURS segment which contains iterations of the MODEL 204 OTHER DRIVER field. In the Master File, the segment declaration for the dependent segment must include the OCCURS attribute.

If a MODEL 204 file contains multiple logical record types, each logical record type corresponds to one segment. You can represent a MODEL 204 file with several logical record types either:

- As a multi-segment server Master File structure with a hierarchical retrieval path. The segments are cross-referenced by Embedded Joins defined on the parent/child relationships.
- As several individual segments, each described in a separate pair of Master and Access Files.

In a multi-segment Master File, you can include segment descriptions for all of the logical record types or only those you need for your reports. To identify the parent of a dependent segment, specify the PARENT attribute in the segment declaration of the dependent segment.

In the corresponding Access File, for each logical record type you must specify the:

- MODEL 204 file and password.
- MODEL 204 record type field and the unique value that identifies it.
- Shared field that implements the Join relationship.
- MODEL 204 key fields (TYPE attributes in FIELD declarations identify the appropriate suffix operators).

Note: Record type fields identify individual record types that reside in the same MODEL 204 file. MODEL 204 files that consist of one logical record type do not require record type fields. In the Access File, the RECTYPE and RECTVAL attributes specify record type fields and the values.

Mapping MODEL 204 and Server Relationships

In MODEL 204, interfile relationships are not coded in the MODEL 204 file description. This provides a certain degree of flexibility because the MODEL 204 files can be cross-referenced dynamically. On the other hand, the burden of connecting the MODEL 204 files falls on the user.

MODEL 204 relationships between logical record types are cross-referenced using shared key fields. The adapter supports this implementation. You can logically join MODEL 204 files using the adapter with either of the following techniques:

- Issue JOIN commands to dynamically join the MODEL 204 files.
- Create a multi-segment server Master File structure and describe the Join relationships in a pair of Master and Access Files (this multi-segment structure is also called an Embedded Join or Server View).

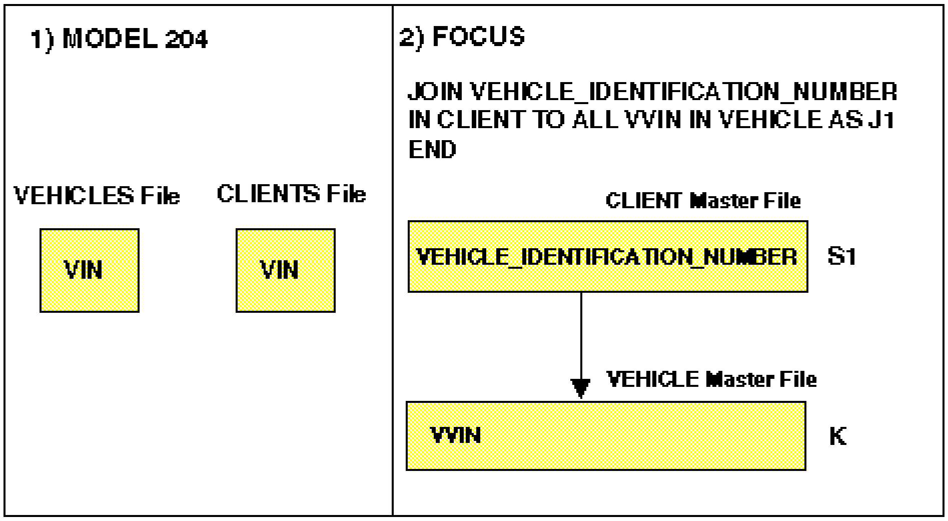
Both techniques are described in the following sections.

Dynamic Joins

If there is a shared or common field, you can use the server JOIN command to dynamically Join:

- Separate physical MODEL 204 files that each contain one logical record type. These MODEL 204 files are usually described as single segment Master Files.
- Logical record types that are described to the server as single segment Master Files, but that originate from the same MODEL 204 file.
- Multi-segment server structures that represent several MODEL 204 files and/or different record types from the same MODEL 204 file. Multi-segment Master Files are explained in *Embedded Joins* on page 25-9.

The following diagram illustrates a MODEL 204 Join and its equivalent Server Join:


1) MODEL 204
VEHICLES File
CLIENTS File
VIN
VIN
2) FOCUS
JOIN VEHICLE_IDENTIFICATION_NUMBER
IN CLIENT TO ALL VVIN IN VEHICLE AS J1
END
CLIENT Master File
VEHICLE_IDENTIFICATION_NUMBER
S1
VEHICLE Master File
VVIN
K

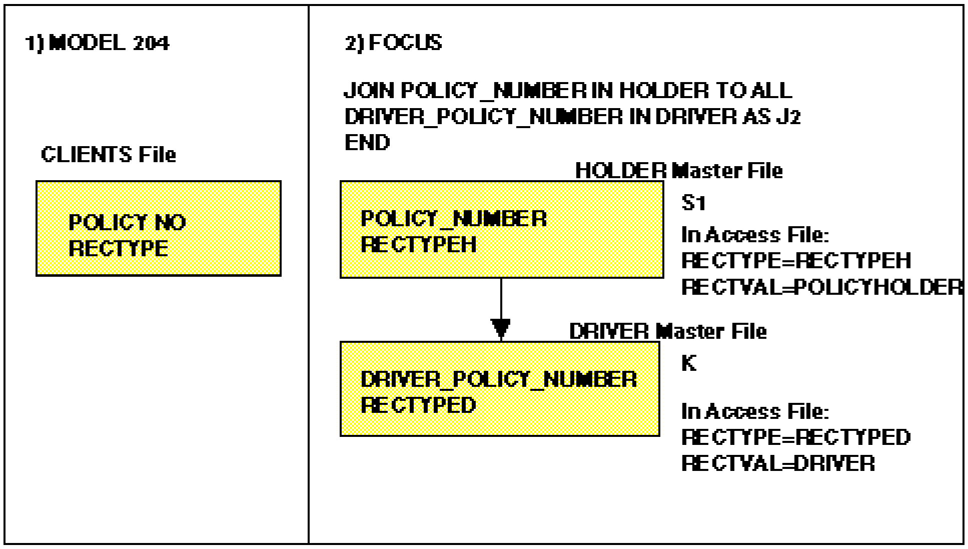
1. Are two MODEL 204 files, VEHICLES and CLIENTS. Each file contains one logical record type for this example. The common or key field is VIN.

2. Are two equivalent server segments. Each segment is described in a separate pair of Master and Access Files. The field names for the common field are VEHICLE_IDENTIFICATION_NUMBER and VVIN. The MODEL 204 key fields are specified with the TYPE=KEY attribute in the Access File.

The JOIN command includes the names of the Master Files and the common field names. To retrieve data from the joined structure, issue a report request that specifies the host (or parent) Master File from the JOIN command.

Note: Since in this example each MODEL 204 file consists of one logical record type, record type fields are not specified in the Access Files.

The following diagram illustrates a MODEL 204 file with two record types and the Server Join:



1. Is the MODEL 204 CLIENTS file containing two logical record types. The common or key field is POLICY NO. The RECTYPE field contains the value POLICYHOLDER for the HOLDER record type and the value DRIVER for the DRIVER record type.

2. Are two equivalent Master File segments. Each segment is described in a separate pair of Master and Access Files. The field names for the common field are POLICY_NUMBER and DRIVER_POLICY_NUMBER. The MODEL 204 key fields are specified with the TYPE=KEY attribute in the Access File. Since both record types reside in the same MODEL 204 file, each Access File segment declaration also includes RECTYPE (record type) and RECTVAL (record type value) attributes.

The JOIN command includes the names of the Master Files and the common field names. To retrieve data from the joined structure, issue a report request that specifies the host (or parent) Master File from the JOIN command.

Embedded Joins

You can also implement Join relationships by describing a multi-segment FOCUS structure in a single pair of Master and Access Files. If there is a shared or common field, you can create an Embedded Join for:

- Separate physical MODEL 204 files that each contain one logical record type. The MODEL 204 files are described as segments in the multi-segment structure.
- Logical record types that originate from the same MODEL 204 file. These record types are described as segments in the multi-segment structure. In the Access File, the RECTYPE and RECTVAL attributes in each segment record identify the record type field and its corresponding value.
- Other multi-segment FOCUS structures. A Master File might relate two hierarchical structures as one. For example, the AUTO Master File might incorporate the VEHICLE and CLIENT Master Files.

Embedded Joins offer two advantages:

- They create a logical FOCUS view of the record types. The view may also provide a measure of security by limiting access to segments or fields.
- You do not have to issue an explicit JOIN command.

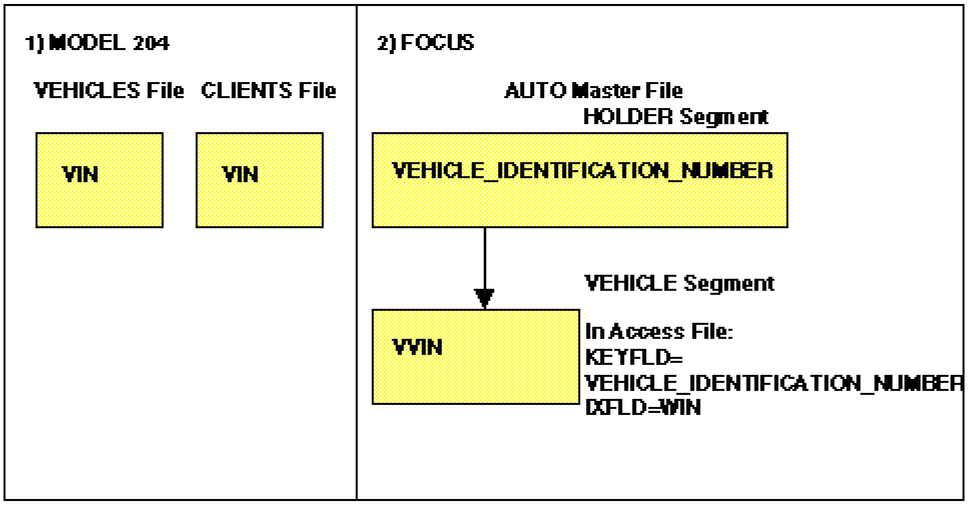
When you create a Master File, describe the MODEL 204 logical record types or files as a hierarchy. Start first by choosing a record type or file to be the root segment. The root segment has dependent segments which, in turn, may have dependent segments. The dependent segments identify the superiors (or parents) with the PARENT attribute. Any record type can act as a dependent provided that its shared field is a key field.

In the associated Access File, for each pair of related segments, specify the shared field from the parent and dependent segments with the KEYFLD and IXFLD attributes. The Interface implements the relationship by matching values at run time.

The Embedded Join is executed automatically when you issue a report request that references fields from two related segments. You do not specify the shared field in your report request; the selection of the shared field is transparent.

Note: It is possible to join unrelated MODEL 204 files and record types that do not have common fields by describing a DUMMY segment.

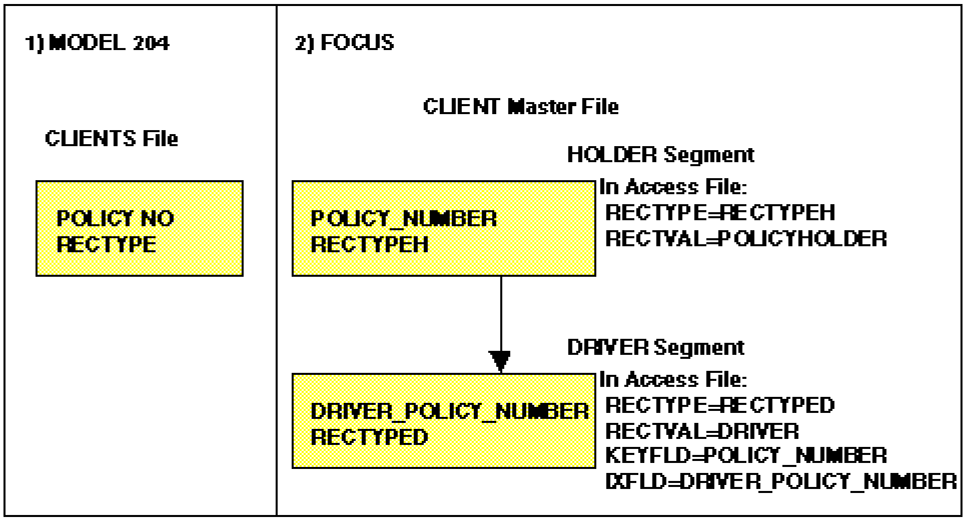
The following diagram illustrates an Embedded Join for two MODEL 204 files:



1. Are two MODEL 204 files, VEHICLES and CLIENTS. Each file contains one logical record type for this example. Since they exist as physically separate files, they do not contain a MODEL 204 record type field. The common or key field is VIN.
2. Are two equivalent FOCUS segments described in one pair of Master and Access Files. The field names for the common field are VEHICLE_IDENTIFICATION_NUMBER and VVIN. The MODEL 204 key fields are specified with the TYPE=KEY attribute in the Access File.

   To create a parent/child relationship, describe the HOLDER segment as the parent (or root) and the VEHICLE segment as the dependent in the Master File. Include the PARENT=HOLDER attribute in the segment declaration for VEHICLE. In the Access File, specify the KEYFLD and IXFLD attributes to create the Embedded Join.

The following diagram illustrates an Embedded Join for a MODEL 204 file with two record types:



1. Is the MODEL 204 CLIENTS file containing two logical record types. The common or key field is POLICY NO. The RECTYPE field contains the value POLICYHOLDER for the HOLDER record type and the value DRIVER for the DRIVER record type.

2. Are two equivalent FOCUS segments described in one pair of Master and Access Files. The field names for the common field are POLICY_NUMBER and DRIVER_POLICY_NUMBER. The MODEL 204 key fields are specified with the TYPE=KEY attribute in the Access File. Since both record types reside in the same MODEL 204 file, the Access File segment declarations also include the RECTYPE (record type) and RECTVAL (record type value) attributes.

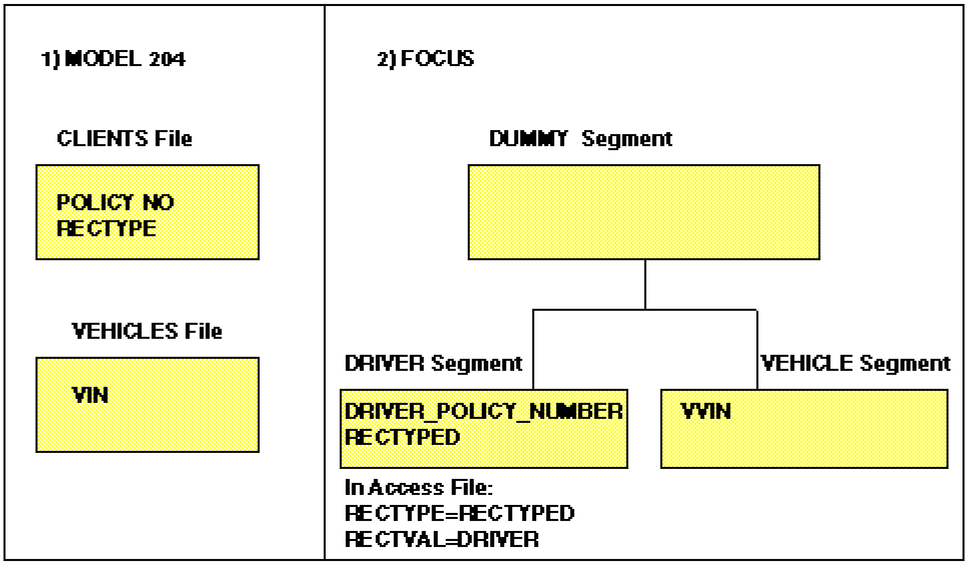
   To create a parent/child relationship, describe the HOLDER segment as the parent (or root) and the DRIVER segment as the dependent in the Master File. Include the PARENT=HOLDER attribute in the segment declaration for DRIVER. In the Access File, specify the KEYFLD and IXFLD attributes in the DRIVER segment declaration to create the Embedded Join.

Joining Unrelated Files

You can describe unrelated MODEL 204 files or logical record types as Embedded Joins even if a shared field does not exist. To do so, describe a special dummy segment (SEGNAME=DUMMY) as the root in the Master File. Then, specify the unrelated segments as parallel dependents of the dummy root. Since the dummy segment is a virtual segment, do not describe fields for it in the Master File, and do not include a corresponding segment declaration for it in the Access File.

If your report request includes fields from two or more segments, FOCUS makes only one retrieval pass over the answer set returned by the MODEL 204 DBMS. Fields specified for sort phrases (BY and ACROSS) and screening tests (IF and WHERE) must lie on the same retrieval path as the other requested fields. You cannot use a field from one unrelated segment to sort the data in other segments.

The following diagram illustrates two MODEL 204 files, the CLIENTS file and the VEHICLES file. These two files are joined using a FOCUS dummy segment:



1. Are two MODEL 204 files, CLIENTS and VEHICLES. The record type field in the DRIVER segment identifies all client records that have the value 'DRIVER'. A dummy segment is needed because there is no shared field to cross-reference or join these files.
2. Is an equivalent multi-segment FOCUS structure. The root is a DUMMY segment. The two dependents are related to the root; they are not related to each other because a common field does not exist.

To create a Master File for this structure, specify the segment declaration for the dummy segment (SEGNAME=DUMMY) first, so it functions as the root. Do not describe fields for it. Then, describe the dependent segments, and include the PARENT=DUMMY attribute to identify the dummy segment as the parent. In the associated Access File, do not specify a segment declaration for the dummy segment; include segment declarations only for the dependent segments. The record type field is required in the DRIVER segment since the DRIVER segment comes from a MODEL 204 file, CLIENTS, that contains two logical record types. Do not specify KEYFLD/IXFLD values.

Summary of Mapping Rules

MODEL 204 and FOCUS use similar elements:

MODEL 204 Entities	FOCUS Entities
MODEL 204 field	FOCUS field
Logical record type	FOCUS segment
Individual record	Segment instance
MODEL 204 file with one record type	FOCUS segment
Multiply-occurring MODEL 204 fields in a record	FOCUS OCCURS segment

Some general rules for describing MODEL 204 files are:

- Each logical record type becomes a FOCUS segment. A FOCUS segment may represent an entire MODEL 204 file (with one logical record type) or one logical record type from a MODEL 204 file with several record types.
- A MODEL 204 file containing two or more logical record types must have a record type field to identify each record type.
- A MODEL 204 file containing one logical record type does not require a record type field; the MODEL 204 file may contain an optional one.
- Relationships can be implemented with the JOIN command or by describing Embedded Joins in the FOCUS file descriptions. The maximum number of MODEL 204 files or logical record types that can be related is 64 for Embedded Joins and 16 for Dynamic Joins.

Managing MODEL 204 Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a Master File and Access File that describe the structure of the data source and the server mapping of the MODEL 204 data types. The logical description of a MODEL 204 file is stored in a Master File, which describes the field layout. The physical attributes of the MODEL 204 file, such as the database number, file number, descriptors, and security are stored in the Access File.

Master Files

A MODEL 204 file consists of records and fields. It can represent one file or a collection of files called a group. If records have identical sets of fields, the unit of records is called a logical record type. A logical record type is described as a single segment in the Master File.

A Master File consists of file, segment, and field declarations. Rules for declarations are:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,$). Text appearing after the comma and dollar sign is treated as a comment.
- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas separate attribute=value pairs.
- Certain attributes are required; the rest are optional.

The following sections summarize the syntax for each type of declaration and then present detailed explanations for each attribute.

Syntax: How to Declare File Attributes

Each Master File begins with a file declaration that names the file and describes the type of data source--a MODEL 204 file or group in this case. The file declaration has two attributes, FILENAME and SUFFIX.

```
FILE[NAME]=name, SUFFIX=M204IN [,$]
```

where:

name

Is a 1- to 8-character name that identifies the Master File. The name must comply with server file naming conventions.

M204IN

Is the SUFFIX value that specifies the Adapter for MODEL 204.

Syntax: How to Declare Segment Attributes

Each logical record type described in a Master File requires a segment declaration that consists of at least two attributes, SEGNAME and SEGTYPE.

```
SEGNAME={name|DUMMY}, SEGTYPE={S|U} [, PARENT=parent] ,$
```

where:

name

Is a 1- to 8-character name that identifies the segment. The value DUMMY creates a virtual segment used for joining unrelated segments.

S|U

Indicates to the adapter that the MODEL 204 DBMS handles the storage order of the data. S is the default setting.

parent

For multi-segment structures (Embedded Joins) only, is the SEGNAME value for the parent record type. This attribute is required in segment declarations that describe dependent segments.

SEGNAME

The SEGNAME (or SEGMENT) attribute identifies each MODEL 204 logical record type. The SEGNAME value can be a maximum of 8 alphanumeric characters. It may be the name of the record type or an arbitrary name and must be unique within a Master File.

The SEGNAME value from the Master File is also specified in the Access File where the segment declaration identifies the name of the MODEL 204 file. In this manner, the SEGNAME value serves as a link to the actual MODEL 204 file name.

SEGTYPE

In a single segment Master File, the SEGTYPE value is S. The MODEL 204 DBMS handles the storage order of the data.

In a multi-segment Master File, SEGTYPE values for dependent segment declarations can be S or U (for unique).

- SEGTYPE=S indicates that the dependent record type has a one-to-many or many-to-many relationship with the record type named as its parent. For every record of the parent record type, there may be more than one record in the dependent record type.
- SEGTYPE=U indicates that the dependent record type has a one-to-one or a many-to-one relationship with the record type named as its parent. For every record of the parent record type, there may be (at most) one record in the dependent record type. For a one-to-one relationship to exist, both record types must share a common key. For a many-to-one relationship to exist, the common key of the dependent record type must be a subset of the common key of the parent record type.

If the SEGTYPE attribute is omitted from any segment record in the Master File, its value defaults to S.

PARENT

The PARENT attribute is required in dependent segment declarations. Its value is the SEGNAME value of the dependent record type's parent (the logical record type to which it will be related at run time).

Syntax: How to Declare Field Attributes

Each record in a logical record type consists of one or more fields. These MODEL 204 fields are described in the Master File with the primary attributes FIELDNAME, ALIAS, USAGE, and ACTUAL. You are only required to describe MODEL 204 fields you use in reports, and you can specify them in any order within each segment.

```
FIELD[NAME]=field, [ALIAS=] [m204field], [USAGE=]display, [ACTUAL=]An ,$
```

where:

field

Is a 1- to 66-character field name. In requests, the field name can be qualified with the Master File and/or segment name. Although the qualifiers and qualification characters do not appear in the Master File, they count toward the 66 character maximum.

m204field

Is the MODEL 204 field name, which can be up to 66 characters (or, optionally left blank if the field name exceeds 66 characters and is described in the Access File).

display

Is the server display format for the field.

A*n*

Is the server definition of the MODEL 204 field format and length (n).

You can omit the ALIAS, USAGE, and ACTUAL keywords from the field declaration if the values are specified in the standard order (FIELD, ALIAS, USAGE, ACTUAL). For example, the following declarations are equivalent:

```
FIELD = YEAR, ALIAS=YEAR, USAGE=A2, ACTUAL=A2,$
FIELD = YEAR, YEAR, A2, A2,$
```

FIELDNAME

Field names must be unique within the Master File and may consist of up to 66 alphanumeric characters. MODEL 204 field names are acceptable values if they meet the following naming conventions:

- Names can consist of letters, digits, and underscore characters. Special characters and embedded blanks are not advised.
- The name must contain at least one letter.

Since the field name appears as the default column title for reports, select a name that is representative of the data. You can specify field names, aliases, or a unique truncation of either in requests.

Duplicate field names (the same field names and aliases) within a segment are not permitted. Requests can qualify duplicate field names from different segments with the name of the Master File and/or segment.

Note: Field names specified in OCCURS segments may not be qualified.

ALIAS

The ALIAS value for each field must be the full MODEL 204 field name; the adapter uses it to generate HLI calls. The ALIAS name must be unique within the segment. If the name is longer than the 66 character maximum, leave the ALIAS attribute blank in the Master File and specify the alias in the Access File instead. The ALIAS name must comply with the field naming conventions listed previously. If the MODEL 204 field name contains an embedded blank, enclose the name in single quotation marks.

Aliases may be duplicated within the Master File if they are defined for different MODEL 204 logical record types. However, the corresponding field names must be unique. For example, the ALIAS values for two MODEL 204 record type fields may be RECTYPE, but the field names must be unique.

Note: Do not use RECTYPE as a field name.

USAGE

The USAGE attribute indicates the display format of the field. An acceptable value must include the field type and length and may contain edit options. The server uses the USAGE format for data display on reports. All standard server USAGE formats (A, D, F, I, P) are available.

ACTUAL

The ACTUAL attribute indicates the server representation of MODEL 204 field formats as returned by the MODEL 204 DBMS in the data buffer. Specify an alphanumeric format (A) with a length (n) equal to the maximum length for the MODEL 204 field. If the ACTUAL length is less than the maximum length for the MODEL 204 field, field values will be truncated.

Note: Since MODEL 204 file descriptions do not define field lengths, ask your MODEL 204 database administrator or consult other sources, such as data dictionaries, for field information.

Syntax: How to Declare OCCURS Attributes

Describe each multiply occurring field that exists in a MODEL 204 record as an OCCURS segment. The segment declaration for an OCCURS segment consists of the SEGNAME, PARENT, and OCCURS attributes. It contains one field declaration to represent the multiply occurring field, and, optionally, a field declaration for an internal counter, the ORDER field. Each OCCURS segment is described as a dependent segment. Its parent segment describes all non-repeating fields from the logical record type. You do not have to describe an OCCURS segment if you do not plan to use the multiply occurring field in report requests.

```
SEGNAME=segname, PARENT=parentname, OCCURS=VARIABLE,$
```

where:

segname

Is the 1- to 8-character name of the OCCURS segment.

parentname

Is the SEGNAME value of the parent segment that contains the non-repeating fields from the logical record type.

VARIABLE

Indicates that the number of occurrences varies.

Note:

- The SEGTYPE attribute is always S for OCCURS segments and, therefore, can be omitted.
- OCCURS segments generally do not require corresponding segment declarations in the Access File. An exception is for processing OCCURS segments that exceed the buffer capacity.
- Field names in OCCURS segments may not be qualified with Master File or segment names, but the TYPE attribute can define suffix operators in the Access File.

Syntax: How to Track a Sequence Within Multiply Occurring Fields

The ORDER field is an optional counter that you can define to identify the sequence number of each multiply occurring field when the order of data is significant. You can use its value in subsequent report requests. The ORDER field does not represent an existing MODEL 204 field; it is used only for internal processing. The ORDER field must be the last field described in the OCCURS segment.

```
FIELD=name, ALIAS=ORDER, USAGE=In, ACTUAL=I4,$
```

where:

name

Is any meaningful name.

ALIAS

Must be ORDER.

USAGE

Is an integer (I*n*) format.

ACTUAL

Must be I4.

The following annotated example illustrates an OCCURS segment in the VEHICLE Master File.

```
1. SEGNAME=VEHICLE,$
      FIELD=VVIN ,ALIAS=VIN  ,A10 ,A10 ,$
      FIELD=YEAR ,ALIAS=YEAR ,A2  ,A2  ,$
      FIELD=MAKE ,ALIAS=MAKE ,A16 ,A16 ,$
2. SEGNAME=XOTHDR, PARENT=VEHICLE, OCCURS=VARIABLE ,$
3. FIELD=OTHER_DRIVERS, ALIAS='OTHER DRIVER' ,A6 ,A6 ,$
4. FIELD=DRVCNT, ALIAS=ORDER ,I4 ,I4 ,$
```

Notice that:

1. Is the segment declaration for the parent segment, VEHICLE.
2. Is the OCCURS segment declaration.
3. Is the field declaration for the multiply occurring field. Field qualifiers (Master File and segment names) are not permitted, but the TYPE attribute can define a suffix operator in the Access File.
4. Is the field declaration for the ORDER field. As an internal counter field, it does not correspond to a MODEL 204 field.

Access Files

Each Master File for a MODEL 204 file must have a corresponding Access File. The name of the Access File must be the same as that of the Master File. In OS/390, the Access File PDS is allocated to ddname ACCESS in the server JCL. The Access File associates a segment in the Master File with the MODEL 204 logical record type it describes.

The Access File consists of account, segment, and field declarations. The Access File minimally identifies account and segment information and field suffix operators. It can also contain field declarations for aliases that exceed 66 characters.

In multi-segment structures, the Access File must contain a segment declaration for each logical record type described in the Master File. Segment information may include Embedded Joins and MODEL 204 record type fields. The order of segment declarations in the Access File must correspond to the order in the Master File.

The Access File consists of 80 character declarations in comma-delimited format. Rules for declarations are:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,$).

- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas separate attribute=value pairs.
- Blank lines and leading or trailing blanks around attribute=value pairs are ignored.
- Declarations starting with an asterisk (*) in Column 1 are treated as comments.
- Values that contain commas, equal signs, dollar signs, or embedded blanks must be enclosed in single quotation marks.
- If you wish, you can number the declarations in columns 73 through 80.

For example, the VEHICLE Access File contains one segment declaration because the VEHICLE Master File contains only one logical record type:

```
FILE=VEHICLE1, SUFFIX=M204IN

SEGNAME=VEHICLE,$
FIELD=VVIN                 ,ALIAS=VIN                  ,A10   ,A10   ,$
FIELD=YEAR                 ,ALIAS=YEAR                 ,A2    ,A2    ,$
FIELD=MAKE                 ,ALIAS=MAKE                 ,A16   ,A16   ,$
FIELD=MODEL                ,ALIAS=MODEL                ,A12   ,A12   ,$
FIELD=BODY                 ,ALIAS=BODY                 ,A4    ,A4    ,$
FIELD=COLOR                ,ALIAS=COLOR                ,A10   ,A10   ,$
FIELD=VEHICLE_PREMIUM      ,ALIAS='VEHICLE PREMIUM'    ,A8    ,A8    ,$
FIELD=COLLISION_PREMIUM    ,ALIAS='COLLISION PREMIUM'  ,A6    ,A6    ,$
FIELD=LIABILITY_PREMIUM    ,ALIAS='LIABILITY PREMIUM'  ,A12   ,A12   ,$
FIELD=LIABILITY_LIMIT      ,ALIAS='LIABILITY LIMIT'    ,A3    ,A3    ,$
FIELD=DEDUCTIBLE           ,ALIAS=DEDUCTIBLE           ,A3    ,A3    ,$
FIELD=GARAGE_LOCATION      ,ALIAS='GARAGING LOCATION'  ,A12   ,A12   ,$
FIELD=TRANS                ,ALIAS=TRANS                ,A2    ,A2    ,$
FIELD=USAGE                ,ALIAS=USAGE                ,A12   ,A12   ,$
FIELD=VEHICLE_USAGE_CLASS  ,ALIAS='VEHICLE USE CLASS'  ,A2    ,A2    ,$
FIELD=VEHICLE_RATING       ,ALIAS='VEHICLE RATING'     ,A1    ,A1    ,$
FIELD=OWNER_POLICY         ,ALIAS='OWNER POLICY'       ,A6    ,A6    ,$
FIELD=PRINCIPLE_DRIVER     ,ALIAS='PRINCIPLE DRIVER'   ,A12   ,A12   ,$

SEGNAME=XOTHDR,  PARENT=VEHICLE, OCCURS=VARIABLE,$
FIELD=OTHER_DRIVERS        ,ALIAS='OTHER DRIVER'       ,A6    ,A6    ,$
FIELD=DRVCNT               ,ALIAS=ORDER                ,I4    ,I4    ,$
```

The following sections summarize the syntax for each type of declaration and provide detailed explanations for each attribute.

Syntax: How to Declare Account Declaration

The account declaration indicates authorization to access MODEL 204 files. At most, one account declaration is required; if included, it must be the first declaration in the Access File.

```
ACCOUNT=m204id, ACCOUNTPASS=password, IFAMCHNL=channel ,$
```

where:

m204id

Is the 1- to 16-character name for the MODEL 204 account (user ID).

password

Is the 1- to 16-character password for the account.

channel

Is the 1- to 8-character site specific CRAM channel name. The default is IFAMPROD for OS/390.

Syntax: How to Declare ACCOUNT and ACCOUNTPASS

The ACCOUNT and ACCOUNTPASS attributes describe MODEL 204 accounts (user IDs) and passwords; they protect MODEL 204 files from unauthorized access. The account must have authorization to read the MODEL 204 file or group. Acceptable values are 1- to 16- character values.

You can omit the ACCOUNT and ACCOUNTPASS attributes from the declaration if you specify the values with the adapter M204IN SET M204ACCNT and M204PASS commands. These SET commands also enable you to change ACCOUNT and ACCOUNTPASS values during the session.

Adapter SET commands are explained in *Customizing the MODEL 204 Environment* on page 25-31.

Syntax: How to Declare IFAMCHNL

The IFAMCHNL attribute describes the CRAM channel name for OS/390. You must specify the CRAM channel if more than one ONLINE or IFAM2 job is running or if the CRAM channel name is not IFAMPROD. For the channel name used at your site, check with your MODEL 204 database administrator or the IFAM2 systems support staff.

The account declaration in the Access File can consist of only the IFAMCHNL attribute=value pair if the account and password values are set with the adapter M204IN SET M204ACCNT and M204PASS commands.

Syntax: How to Declare Segment Declarations

The segment declaration in the Access File establishes the link between the server Master File and the MODEL 204 file or group. Attributes that constitute the segment declaration are: SEGNAME, FILE, PASS, KEYFLD, IXFLD, RECTYPE, RECTVAL, and ACCESS. Values for SEGNAME, FILE, and PASS are required; the rest apply to multi-segment structures. The ACCESS attribute applies to OCCURS segments.

```
SEGNAME=name, FILE=m204file, PASS=password
  [,KEYFLD=pfield] [,IXFLD=cfield] [,RECTYPE=rtfield] [,RECTVAL=rtvalue]
  ,ACCESS={ALL|nnnn[/xxx]|AUTO[/xxx]}
```

where:

name

Is the SEGNAME value from the Master File.

m204file

Is the 1- to 8-character name for the MODEL 204 file.

password

Is the 1- to 8-character read password for the MODEL 204 file. The value can be blank if a password does not exist.

pfield

Is the field name of the common field from the parent segment, and is used to implement the Join relationship. It is required in dependent segment declarations for multi-segment structures.

cfield

Is the field name of the common field from the dependent segment, and is used to implement the Join relationship. It is required in dependent segment declarations for multi-segment structures.

rtfield

Is the field name for the MODEL 204 record type field that exists when a MODEL 204 file has several logical record types.

rtvalue

Is the value that identifies the MODEL 204 record type or a field name.

ALL
nnnn/*xxx*
AUTO/*xxx*

Indicates how OCCURS segments should be processed when all occurrences cannot fit into the buffer at once and a (FOC4883) IFMORE FAILED message results. ALL is the default value. These options are described in *Access Files* on page 25-21.

Note:

- For multi-segment structures, the order of the Access File segment declarations must correspond to the order in the Master File.
- Dummy segments used to link unrelated segments do not have corresponding segment declarations in the Access File.

Syntax: How to Declare SEGNAME

The SEGNAME value must be the first attribute in each Access File segment declaration. It must be identical to the SEGNAME value in the Master File.

Syntax: How to Declare FILE and PASS

The FILE and PASS attributes are required in each Access File segment declaration. They describe MODEL 204 file security information.

The value for the FILE attribute is the name of the MODEL 204 file or group that contains the logical record type. The value for the PASS attribute is either the read password associated with the MODEL 204 file or blank if the password does not exist. For both attributes, acceptable values are 1- to 8-character values.

Access Files can be encrypted to prevent access to this security information.

Syntax: How to Declare KEYFLD and IXFLD

The KEYFLD and IXFLD attributes identify the common fields for parent/child relationships in a multi-segment Master File. These relationships are referred to as Embedded Joins, server views, or cross-references.

For each dependent segment, the KEYFLD and IXFLD attributes identify the field names of the common or shared field that implements the Embedded Join. The parent field supplies the value for cross-referencing; the dependent field contains the corresponding value. The adapter implements the relationship by matching values at run time.

The value for the KEYFLD attribute is a 1- to 66- character field or alias name from the parent segment. The value for the IXFLD attribute is a 1- to 66-character field or alias name from the dependent segment.

Note:

- Include the pair of attributes in Access File segment declarations for dependent segments. Do not specify them in the segment declaration for the root segment.
- Do not specify KEYFLD and IXFLD attributes for dependent segments of a virtual parent segment (PARENT=DUMMY).

A JOIN can be based on more than one field in the host and cross-referenced logical record types. If the MODEL 204 file uses multiple fields to establish a relationship or link between logical record types, you can specify concatenated fields in an Embedded Join (you can also specify multiple fields with the Dynamic JOIN command).

In a multi-field Embedded Join, the KEYFLD and IXFLD values consist of a list of the component fields separated by slashes (/). Additional Access File attributes are not required.

```
KEYFLD = field1/field2/....
IXFLD = cfield1/cfield2/....
```

where:

field1/...

Is a composite of up to 16 key fields from the parent segment. Slashes are required.

cfield1/...

Is a composite of up to 16 key fields in the dependent segment.

The adapter compares each field pair for comparable data formats prior to format conversion. It evaluates each field pair with the following rules:

- Any of the MODEL 204 key fields listed in Field Declarations can be used to create an Embedded Join.
- Parent and dependent fields must be real fields.
- All participating fields for either the parent or dependent file must reside in the same segment.
- KEYFLD and IXFLD attributes must specify the same number of fields. If the format of the parent field is longer than the format of the dependent field, its values are truncated. If the format of the parent field is shorter, its values are padded with zeros or blanks.
- The KEYFLD and IXFLD attributes can consist of a maximum of 16 concatenated fields in high to low significance order. You must specify the field order; the order determines how the values will be matched.
- Each pair of fields in the KEYFLD/IXFLD attribute must have comparable data formats. The formats of the parent fields must be compatible with the formats of the dependent fields as specified in the MODEL 204 file description.

To implement Joins, the MODEL 204 DBMS converts the alphanumeric server field formats to equivalent MODEL 204 field formats in order to perform the necessary search and match operations. When the MODEL 204 DBMS returns the answer set of matched values, it also converts the values back to alphanumeric formats.

Internally, each KEYFLD value is passed to the MODEL 204 DBMS in the IFFIND call to retrieve all matching records (segment instances) for the dependent segment.

Syntax: How to Declare RECTYPE and RECTVAL

The RECTYPE and RECTVAL attributes identify the MODEL 204 record type field and its corresponding value. A MODEL 204 record type field exists in a MODEL 204 file description when the file has more than one logical record type. If the MODEL 204 file consists of one logical record type, it may contain an optional record type field.

The value for the RECTYPE attribute is the 1- to 66-character field or alias name of the MODEL 204 record type field. The value for the RECTVAL attribute is the identifying value for the record type field; it can be up to 255 characters in length or a field name. The RECTVAL attribute is required whenever the RECTYPE attribute is used.

The RECTYPE and RECTVAL attributes are required in each Access File segment declaration (including the one for the root) if the MODEL 204 file contains record types.

Note: Do not use the name RECTYPE as a field name in the Master File.

ACCESS

OCCURS segments are generally described only in the Master File and do not require corresponding segment declarations in the Access File. However, sometimes the number of occurrences retrieved for an OCCURS segment exceeds the MODEL 204 buffer capacity, causing an error condition. You can alter the processing of OCCURS segments by including a segment declaration in the Access File.

The MODEL 204 buffer is used to return data to IFAM2 applications. Buffer capacity is sometimes exceeded when both of the following conditions exist:

- A multiply occurring field repeats a large number of times (for example, 2000 occurrences).
- The total number of occurrences multiplied by the size of the OCCURS segment is greater than the size of the IFAMBS buffer (the IFAMBS size is specified as a parameter in the MODEL 204 startup procedure).

The size of the IFAMBS buffer is determined by a predefined formula:

```
IFAMBS := (LIBUFF * 7) + 284 s 32688
```

When the buffer capacity is exceeded, the MODEL 204 DBMS sends the adapter the following message:

```
IFMORE error M204.0903 - Results too long
```

This causes the server message:

```
(FOC4883) IFMORE FAILED
```

If you encounter this condition, you can include a segment declaration in the Access File to alter the number of occurrences retrieved. The adapter incorporates your specification in its HLI call to the DBMS.

An OCCURS segment declaration in an Access File consists of two attributes, SEGNAME and ACCESS. The ACCESS attribute indicates the type of processing to be performed.

```
SEGNAME=name, ACCESS={ALL|nnnn[/xxx]|AUTO[/xxx]} ,$
```

where:

name

Is the SEGNAME value for the OCCURS segment from the Master File.

ALL

Indicates that the adapter will attempt to retrieve all occurrences of the OCCURS segment. ALL is the default value.

Note: If the ACCESS attribute is omitted, the ALL setting is assumed.

nnnn

Specify the largest number of occurrences that exists for the multiply occurring field. Based on the nnnn value, the adapter may manipulate the length of the QTBL buffer for the user session and then reset it to its original length.

AUTO

Indicates that the adapter will retrieve up to 5000 occurrences. If there are more than 5000 occurrences, use nnnn instead of AUTO.

/xxx

Instructs the adapter to issue multiple IFMORE calls, with each call retrieving xxx occurrences. This parameter is optional.

Append to the nnnn or AUTO parameters to indicate that you want to control the number of records to be retrieved by each IFMORE call. If xxx is larger than the maximum allowed (based on the size of the buffers used by the Adapter for MODEL 204), you will receive the message:

```
(FOC4873) SPECIFIED NUMBER OF INSTANCES CANNOT FIT INTO BUFFER
```

To correct the problem, decrease the xxx value and rerun the request.

Note:

- You can turn this feature on or off by defining the OCCURS segment in the Access File. OCCURS segments not described in the Access File undergo normal processing, which means that the ALL setting is assumed and all occurrences are retrieved at one time.
- If the SEGNAME attribute is specified in the Access File without the ACCESS attribute, ACCESS=ALL is assumed by default.

- Use this feature when the following message occurs:

```
IFMORE error M204.0903 Results too long
```

Syntax: How to Declare Field Declarations

Field declarations in the Access File provide an alternate method for describing certain MODEL 204 field characteristics. Include a field declaration if the MODEL 204 field has a MODEL 204 key designation or if the MODEL 204 field name (the ALIAS attribute) exceeds 66 characters.

Example: Supporting Types of Key Fields

The MODEL 204 DBMS supports a variety of key field types. The adapter supports the MODEL 204 key designations with comparable abbreviations called suffix operators. You describe these operators with the TYPE attribute in the Access File. The actual MODEL 204 key designation determines the proper suffix operator to apply.

The following chart lists MODEL 204 key designations and the corresponding suffix operators.

MODEL 204 Key Designation	Master File Suffix Operator
Key	KEY
Key, invisible	IVK
Ordered Character	ORA
Ordered Numeric	ORN
Numeric Range	RNG
Numeric Range, invisible	IVR

Note: Master Files created in earlier server releases may contain field declarations with suffix operators for MODEL 204 key fields. While this earlier method is supported, the recommended method is to specify suffix operators in the Access File.

Syntax: How to Declare Field Attributes in an Access File

```
FIELD=field [,ALIAS=m204field] [,TYPE=suffix] [,ATTRIBUTE=FRV],
  [ACCDATA= {STR|NUM}] ,$
```

where:

m204field

Is a MODEL 204 field name that exceeds 66 characters. 256 characters is the maximum amount.

Note: Omit the ALIAS attribute if the ALIAS attribute in the Master File already specifies the MODEL 204 field name.

suffix

Is one of the suffix operators KEY, IVK, ORA, ORN, RNG, or IVR.

FRV

This attribute value incorporates MODEL 204 FOR EACH VALUE processing on a BY field in a server COUNT request.

STR

This attribute value will generate character string HLI selection tests for MODEL 204 fields that do not have a MODEL 204 key specification (Non-Ordered, Non-KEY, Non-Numeric Range).

NUM

This attribute value will generate numeric HLI selection tests for MODEL 204 fields that do not have a MODEL 204 key specification (Non-Ordered, Non-KEY, Non-Numeric Range).

Two additional field attributes, ATTRIBUTE and ACCDATA are available:

- ATTRIBUTE will incorporate MODEL 204 FOR EACH VALUE processing on a BY field in a server COUNT request. The field defined with ATTRIBUTE=FRV must be a MODEL 204 Ordered field or a MODEL 204 KEY/FRV field.
- ACCDATA will instruct the adapter to generate either character string or numeric HLI selection tests for MODEL 204 fields that do not have a MODEL 204 key specification (Non-Ordered, Non-KEY, Non-Numeric Range).

Customizing the MODEL 204 Environment

Each Master File for a MODEL 204 file must have a corresponding Access File. The name of the Access File must be the same as that of the Master File. In OS/390, the Access File PDS is allocated to ddname ACCESS in the server JCL. The Access File associates a segment in the Master File with the MODEL 204 logical record type it describes.

The Adapter for MODEL 204 provides several parameters for customizing the environment and optimizing performance.

Adapter SET commands enable you to change parameters that govern its behavior. These parameters control or identify MODEL 204 account security, the size of all adapter buffers, the size of the MODEL 204 FTBL buffer, null values in reports, locked records, and thread management. To display current adapter parameter settings, issue the adapter M204IN SET ? query command. You can issue these commands from an RPC. They remain in effect for the duration of the server client task or until you change them.

Syntax: How to Set Adapter Parameters

The syntax for adapter SET commands includes the ENGINE environment qualifier and the M204IN keyword.

```
ENGINE M204IN SET command value
```

where:

command

Is an Adapter for MODEL 204 command.

value

Is an acceptable value for the adapter command.

Specifying a User Account and Password

You can supply MODEL 204 accounts and passwords with the adapter M204IN SET M204ACCNT and M204PASS commands. Authorized accounts (user IDs) and passwords protect read access to MODEL 204 files or groups of files.

The M204IN SET M204ACCNT and M204PASS commands provide an alternative to supplying security information in encrypted Access Files. You can also use the M204IN SET M204ACCNT and M204PASS commands to override existing account and password values specified with the ACCOUNT and ACCOUNTPASS attributes in the Access File.

Syntax: How to Specify a User Account and Password

To specify an account, issue the following from the command level

```
ENGINE M204IN SET M204ACCNT account
```

where:

account

Is a 1- to 16-character name for an authorized MODEL 204 account (user ID).

To specify a password for an account, issue the following from the command level

```
ENGINE M204IN SET M204PASS password
```

where:

password

Is a 1- to 16-character password for the account.

Note: If you do not issue the SET commands or you leave the values blank, they default to the values of the ACCOUNT and ACCOUNTPASS attributes specified in the Access File.

Specifying the Capacity of Adapter Buffers

You can set or change the maximum size of all adapter buffers with the M204IN SET MAXMBUFF command.

Before you issue the M204IN SET MAXMBUFF command, consider that the MAXMBUFF value:

- Should be greater than the larger of two MODEL 204 buffer settings, LIBUFF and LOBUFF. To identify which buffer is larger, check the parameters in the MODEL 204 start-up procedure or use the adapter Trace Facility (SET TRACEON=M204IN) to display the settings.
- Should be equal to the size of the largest segment in the Master File if the segment size is greater than the MODEL 204 LIBUFF and LOBUFF buffer settings. Having the maximum buffer size equal to that of the largest segment prevents S0C4 ABEND conditions.

Syntax: How to Specify the Capacity of Adapter Buffers

```
ENGINE M204IN SET MAXMBUFF nnnn
```

where:

nnnn

Is the maximum size in bytes. For example, specify 4096 for a MAXMBUFF value of 4K. Unless you explicitly issue this command, it is not used and there is no default value.

Controlling the Size of the FBTL Buffer

The M204IN SET FTBL command enables you to control the size of the MODEL 204 FTBL buffer. The LFTBL parameter, specified in the MODEL 204 start-up procedure, governs the size of the FTBL buffer. You can change it for the current client session only. Issue this command when a MODEL 204 message indicates that the LFTBL value is too small.

Syntax: How to Control the Size of the FTBL Buffer

```
ENGINE M204IN SET FTBL nnnn
```

where:

nnnn

Is the FTBL size in bytes. For example, specify 4096 for an FTBL value of 4K.

You should set FTBL to a value larger than the previously defined MODEL 204 LFTBL value.

Note: The FTBL value you set applies for the current user session only; it is reset when the session is ended.

Indicating Missing Data on Reports

With the adapter M204IN SET MISSING command, you can control the display of MODEL 204 null data on reports. Null data is translated into the server missing data display value. The default NODATA display value is the period (.). When the adapter MISSING attribute is set to ON, the edit specification for all IFGET calls includes an (L) format code. If MODEL 204 returns a zero in the first byte, the adapter considers that field to be missing.

Note: IFFIND calls contain IS PRESENT or IS NOT PRESENT criteria only when the request includes a server IF or WHERE MISSING selection test.

Syntax: How to Indicate Missing Data on Reports

```
ENGINE M204IN SET MISSING {ON|OFF}
```

where:

`ON`

Uses the NODATA value to display MODEL 204 null data.

`OFF`

Null values are not represented on reports. OFF is the default value.

Note:

- The default setting can affect the results of server DML SUM or COUNT aggregate operations. Null values may be counted or averaged in as existing values.
- You must issue the M204IN SET MISSING ON command in order to specify screening conditions (IF or WHERE MISSING tests) for null values.

Locking Records to Ensure Accurate Data Reports

When retrieving records, the MODEL 204 DBMS usually holds locks on the appropriate records to ensure accurate data for reports. The locks prevent other users from updating the target records while MODEL 204 constructs answer sets. The adapter generates standard IFFIND calls that lock the found set in SHR mode.

For certain circumstances, you can issue the adapter SET READWOL (read without locks) command to instruct the adapter to issue an IFFWOL (find without locks) HLI call. Use the following syntax to turn the setting ON:

```
ENGINE M204IN SET READWOL ON
```

For information about the design and performance considerations involved in deciding when to use the MODEL 204 IFFWOL command instead of the IFFIND command, consult the *MODEL 204 Host Language Adapter Programming Guide*.

Controlling Thread Management

The adapter M204IN SET SINGLETHREAD command enables you to control MODEL 204 thread management during record retrieval. A thread is a communications path or link between the adapter and the MODEL 204 DBMS. Adapter requests (generated HLI calls) are transmitted using threads. Threads are deallocated after each adapter request is processed.

There are two types (or modes) of thread management:

- **Multiple.** The adapter creates as many threads as it needs whenever they are needed to access files.
- **Single.** The adapter creates one thread for all file access; this restricts processing to one request.

Syntax: How to Change the Mode of Thread Management

To change the mode, issue the following from the command level

```
ENGINE M204IN SET SINGLETHREAD {ON|OFF}
```

where:

`ON`

Forces the adapter to use a single thread for all file access.

`OFF`

Allows the adapter to use multiple threads. OFF is the default value.

For example, if a request joins 10 files and the SINGLETHREAD setting is ON, only one thread is used to access all of the files. If the setting is OFF, up to 10 threads may be used. For performance reasons, the OFF setting is recommended.

Displaying Adapter Defaults and Current Settings

The adapter M204IN SET ? query command displays adapter defaults and current settings. Issue the following:

```
ENGINE M204IN SET ?
```

Adapter Tracing for MODEL 204

The Adapter for MODEL 204 Tracing Facility is activated from the Web Console main page.

Procedure: How to Activate Adapter Tracing

Select *Diagnostics*, then *Traces*, and click *Enable Traces*.

The Default traces include the Adapter for MODEL 204 tracing information.

CHAPTER 26

Using the Adapter for MQSeries

Topics:

- Configuring the Adapter for MQSeries
- Managing MQSeries Metadata

The Adapter for MQSeries allows applications to access MQSeries data sources. The adapter converts application requests into native MQSeries statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Configuring the Adapter for MQSeries

You can configure the Adapter for MQSeries from the Web Console.

Procedure: How to Configure the Adapter for MQSeries From the Web Console

To configure the adapter for MQSeries from the Web Console:

1. Start the Web Console and select *Data Adapters* from the left pane.
2. Expand the *Add* folder, expand the *Other DBMS* group folder, then expand the *MQSeries* folder and click a connection. The Add MQSeries to Configuration pane opens.
3. No input parameters are required. Simply, click *Configure*.

   The adapter is added to the Configured adapters folder and an entry is added to the edaserv.cfg file.

If you wish to create a synonym for the configured adapter from the Web Console, see *How to Create a Synonym From the Web Console* on page 26-3.

Issuing a FILEDEF Command

If you plan to create a synonym manually, you must first issue a FILEDEF command.

Because a queue in MQSeries can be viewed as a sequential file, the server uses the logical layer of a flat file interface. In order to distinguish between sequential file storage types and MQSeries storage types, you must issue the FILEDEF command prior to any read/write request against MQSeries. FILEDEF is used to provide definitions for the queue manager, the queue name, and the optional message ID to the MQI layer.

Syntax: How to Issue the FILEDEF Command

```
FILEDEF master QMAN queuemgr QUEUE queuename [MSGID msg_id]
```

where:

master

Is the Master File name used to provide the dynamic definitions of the queue manager, the queue, and the optional message ID.

queuemgr

Is the name of the queue manager.

queuename

Is the name of the queue.

msg_id

Is the optional parameter used when producing messages in a queue that are defined outside of the program assigned message ID.

If you wish to create a synonym manually for the configured adapter, see *How to Create a Synonym Manually for Server Family Messages* on page 26-6.

Managing MQSeries Metadata

This topic describes how to use CREATE SYNONYM for MQSeries data sources. It also describes MQSeries data type support.

Creating Synonyms

Synonyms define unique names (or aliases) for each MQSeries data structure that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The Queue Information for MQSeries (Step 1 of 2) opens.
4. In the QMGR input field, enter the name of the queue manager on the machine where the server resides.
5. In the QUEUE input field, enter the the message queue.
6. Click *Select Operations*.
7. Enter the following additional parameters as appropriate:

Parameter	**Description**
Select Application	Select an application directory. baseapp is the default value.
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

8. Complete your selection:
   - To select all messages in the queue, click the check box to the left of the *Default Synonym Name* column heading.
   - To select specific messages in the queue, click the corresponding check boxes.
9. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

10. Click *Create Synonym*.

Synonyms are created and added under the specified application directory.

A status window displays the message:

```
All Synonyms Created Successfully
```

11. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually for Server Family Messages

```
CREATE SYNONYM appname/synonym FOR qmanager.queue.msgid DBMS MQS
END
```

where:

appname

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters for UNIX and Windows Server platforms).

qmanager.queue.msgid

Is the fully qualified name of the queue manager, the message queue, and the message ID if used.

`MQS`

Is the MQS value for the DBMS parameter that CREATE SYNONYM requires.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line, however a single element cannot span more than one line.
- CREATE SYNONYM creates a Master File. An Access File is not created when creating a synonym from an MQSeries queue.

Example: Using CREATE SYNONYM

Use the following syntax to create a synonym for the variable mqpro.

```
CREATE SYNONYM mqpro for edarisc64.MQHOLD1.41423 DBMS MQS
```

Generated Master File

```
FILE     =MQPRO,
SUFFIX   =MQS,
DATASET  ="edarisc64"."MQHOLD1".
 414230000000000000000000000000000000000000000000, $
SEGNAME  =MQPRO   ,
SEGTYPE  =S0
FIELDNAME=DIVISION4      ,E01         ,I11      ,I04     ,$
FIELDNAME=DIVISION_NA4   ,E02         ,A25      ,A28     ,
MISSING  =ON, $
FIELDNAME=DIVISION_HE4   ,E03         ,I11      ,I04     ,
MISSING  =ON, $
$FIELD   =MSGID,
$ ALIAS  =414230000000000000000000000000000000000000000000,
USAGE    =A48,A48,$
$FIELD   =CORID,
$ ALIAS  =4942495F434F525F494432000000000000000000000000000,
USAGE    =A48,A48,$
$FIELD   =TIMEOUT,
ALIAS    =0,
USAGE    =I4,I4,$
```

Retrieving Foreign Messages From the Queue

Example:

Retrieving Foreign Messages From the Queue

You may receive the following types of messages:

- **Server Family Messages.** Messages created by the Adapter for MQSeries. These messages are stored as logical sets, metadata, and data. This implementation allows the server to store results of queries on the sending end, recreate metadata on the receiving end, and retrieve the data as it would from any relational data source. Access to any DBMS requires an active connection, which is not always available. One example of an implementation would be an asynchronous DBMS access.
- **Foreign Messages.** Messages produced by non-server software. The adapter can retrieve any message, but the user must first create metadata.

Example: Retrieving Foreign Messages From the Queue

The Master File must be created manually. For example:

```
FILE=MQS1, SUFFIX=MQS,$
DATASET=edarisc3.MQHOLD1.41423, $
SEGNAME=ROOT, SEGTYPE=S0,$
FIELD=MESSAGE,      ALIAS=AA, USAGE=A720, A720 ,$
  GROUP=SPECIALS,   ALIAS=,   USAGE=A100, A100 ,$
    FIELD=MSGID     ALIAS=,   USAGE=A48,  A48  ,$
    FIELD=CORID,    ALIAS=,   USAGE=A48,  A48  ,$
    FIELD=TIMEOUT, ALIAS=0,   USAGE=I4,   I4   ,$
```

Data Type Support

Data types in foreign messages are generated by application programs writing to the queues. If a COBOL copy book describing data layout exists, the metadata type could be generated by COBOL FD Translator.

The following chart provides information about the default mapping of MQSeries data types to server data types:

COBOL Format	**Data Type**	
	ACTUAL	**USAGE**
PICTURE X(n)	An	An
PICTURE 9(n)	Zn (packed option)	Pn (packed option)
PICTURE 9(n)	An (packed option)	Pn (packed option)
PICTURE S9(n)	Zn	Pn+1
PICTURE 9(n)V9(m)	Zn+m	Pn+m+1.m
PICTURE S9(n)V9(m)	Zn+m	Pn+m+2.m
PICTURE 9(n) COMP (1 < = n < = 4)	I2	I9
PICTURE 9(n) COMP (5 < = n < = 9)	I4	I9
PICTURE 9(n) COMP (n > 9)	A8	A8
COMP-1	F4	F8
COMP-2	D8	D15
PICTURE 9(n) COMP-3	P(n+2/2)	Pn

COBOL Format	Data Type	
	ACTUAL	USAGE
PICTURE S9(n) COMP-3	P(n+2/2)	Pn+1
PICTURE 9(n) V9(m) COMP-3	P(n+m+2/2)	Pn+m+1.m
PICTURE S9(n) V9(m) COMP-3	P(n+m+2/2)	Pn+m+2.m

The format conversions are subject to the following limitations:

- Alphanumeric fields are limited to 256 characters. Elementary fields that exceed 256 characters are truncated to 256 characters and have filler fields inserted to provide for the total length. Group fields that exceed 256 characters are commented.
- Unsigned zoned fields that exclude a decimal point are described with a USAGE of packed (P) or alphanumeric (A), depending on the value entered for the Zoned Numeric Field Usage option on the Translator menu.
- COMP-4 binary fields are described the same as COMP fields.
- Binary fields that exceed 9 digits in the picture clause are described with ACTUAL format A8 and USAGE format A8.

The maximum length of the packed USAGE format depends on the software release that uses the generated Master File. For releases that support long packed fields (16 or more digits), the maximum ACTUAL length is 16 bytes and the maximum USAGE length is 31 digits (or 32 characters including a decimal point and sign). For earlier releases, the maximum ACTUAL length is 8 bytes and the maximum USAGE length is 15 total digits, including decimal and sign. The number of decimal places is limited to one fewer than the total USAGE length. In general, the USAGE length of packed fields is calculated as the sum of the digits to the left of the decimal (n), the number of decimal positions (m), one for the leading minus sign if present (S), and one for the decimal (V) if present.

The ACTUAL and USAGE formats for GROUP fields are always alphanumeric (A). The ACTUAL length is the sum of the ACTUAL lengths of its components. The USAGE length is the sum of the following:

- The USAGE lengths of all the alphanumeric (A) components.
- 16 for each long packed (P) component.
- 8 for each short packed (P) and double precision (D) component.
- 4 for each integer (I) and floating point (F) component.

CHAPTER 27

Using the Adapter for Digital Standard Mumps

Topics:

The Adapter for Mumps allows applications to access Mumps data sources. The adapter converts application requests into native Mumps statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the Mumps Environment

To prepare the Mumps environment on OpenVMS, the system administrator must set the following logicals:

```
DSM$DEFAULT_DSMAXP028 =
/SOURCE_BUFFER_SIZE   = 65535/SYMBOL_TABLE_SIZE=300000
DSM$ENVIRONMENT       = DSMAXP028
DSM$ID_DSMAXP028      = 00001
```

Configuring the Adapter for Digital Standard Mumps

In this section:

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

Example:

Declaring Connection Attributes

Configuring the adapter consists of specifying connection information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to Mumps, the adapter requires connection information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *Other DBMS* group folder, then expand the MUMPS adapter folder and click a connection. The Add MUMPS to Configuration pane opens.
3. Supply values for the following parameters:

Field	Description
Volume Name	The volume set name where the global data structure exists.
Recovery Mode	Indicates whether recovery mode is turned ON or OFF for the specified volume. **Note:** If you set recovery mode to ON, you must enter the volume name in uppercase character.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

   Tip: Once the adapter is configured, the connection name appears in the navigation pane under *Configured*. If you left-click either the adapter name or the connection name, an appropriate set of options becomes available.

Syntax: How to Declare Connection Attributes Manually

```
ENGINE DSM SET CONNECTION_ATTRIBUTES volume recovery_mode
```

where:

`DSM`

Indicates the Adapter for Mumps.

`volume`

Is the volume set name where the global data structure exists.

recovery_mode

Indicates whether recovery mode is turned ON or OFF for the specified volume set name.

Note: If you set recovery mode to ON, you must enter the volume name in uppercase character.

Example: Declaring Connection Attributes

The following illustrations configure the adapter using different volume and recovery specifications:

```
ENGINE DSM SET CONNECTION_ATTRIBUTES ;RAQ:RECOVERY=ON
```

```
ENGINE DSM SET CONNECTION_ATTRIBUTES ;qaq:RECOVERY=OFF
```

Managing Mumps Metadata

In this section:

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Mumps data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Mumps table or view that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The Select UCI for MUMPS (Step 1 of 3) pane opens. The volume name and recovery mode entered during configuration are displayed at the top of the screen.

4. To select all UCIs (User Class Identifiers), click *Select UCI*.

   If you wish to filter UCIs, click the Filter check box and enter a UCI name, then click *Select UCI*.

   The Select UCI Name for MUMPS (Step 2 of 3) pane opens. The list of UCIs displayed depends on the volume you have specified.

5. Choose the UCI that contains the tables (globals) for which you wish to create synonyms, then click *Select Tables.* The Select Synonym Candidates (Step 3 of 3) pane opens.

6. Enter the following parameters, as required:

Parameter	Description
Select Application	Select an application directory in which the synonym will be stored. The default value is baseapp.
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank. **Tip:** While you can change individual names, entering a prefix or suffix enables you to make global changes that are particularly useful when many synonyms are being created.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

7. Complete your selection:

   - To select all globals in the list, select the check box to the left of the *Default Synonym Name* column heading.
   - To select specific globals, select the corresponding check boxes.

8. The Mark for Global Scan check box is activated for all selected globals.

   - Click the check box if you wish to create a synonym that is ready for reporting, based on default subscript names.
   - Do not click the check if you wish to create a synonym template that you can modify manually by entering actual field names before you use it for reporting. The template contains 25 segments.

9. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

10. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

11. In the message window, you can click the *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR global_name DBMS DSM DATABASE vol AT uci
[PARMS "FULL"]
END
```

where:

appname

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

global_name

Is the name of the Mumps global data structure.

vol

Is the volume set name where the global data structure exists.

uci

Is the user class identifier (uci) name where the global data structure exists.

PARMS FULL

Include this parameter to create a synonym that is ready for reporting, based on default subscript names.

Omit this parameter to create a synonym template that you can modify manually by entering actual field names before you use it for reporting. The template contains 25 segments. A synonym template is the default.

END

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.
- CREATE SYNONYM creates a Master File and Access File which represents the server metadata.

Example: Using CREATE SYNONYM

Use the following syntax to create a synonym for the global Clinic.

```
CREATE SYNONYM Clinic FOR Clinic DBMS DSM DATABASE RAB AT MUA
Global Variable CLINIC:
^CLINIC("BROOKDALE CLINIC","231-28-8833","031291")
DR01,DG05,COR3XD063,45.00
^CLINIC("COMMUNITY CLINIC","665-36-5843","040191")
DR22,DG13,PSU1XW004,30.00
^CLINIC("COMMUNITY CLINIC","834-48-8672","041891")
DR67,DG12,CCL2XD021,25.00
^CLINIC("COMMUNITY CLINIC","935-55-4254","043091")
DR85,DG11,PEN3XW015,15.00
^CLINIC("GEORGETOWN CLINIC","241-89-0001","041891")
DR64,DG03,VAL2XD100,100.00
^CLINIC("GEORGETOWN CLINIC","251-64-0011","041791")
DR43,DG10,AMX3XD042,42.00
^CLINIC("GEORGETOWN CLINIC","523-85-0007","031791")
DR64,DG10,AMX2XD020,20.00
^CLINIC("GEORGETOWN CLINIC","651-77-0009","041791")
DR64,DG07,COD3XD063,63.00
^CLINIC("GEORGETOWN CLINIC","656-64-0012","041791")
DR43,DG07,TYL4XD050,25.00
^CLINIC("HILLSIDE CLINIC","637-36-3615","032291")
DR31,DG07,COD1XD021,60.00
^CLINIC("MAIN STREET CLINIC","241-89-0001","041891")
DR58,DG03,VAL2XD070,70.00
^CLINIC("MAIN STREET CLINIC","251-64-0014","041991")
DR73,DG11,PEN3XD042,84.00
```

Generated Clinic Master File

```
FILE=CLINIC, SUFFIX=DSM, $
SEGMENT=GLOBAL, SEGTYPE=U, $
  FIELD=DUMMY, ALIAS=SUBSCRIPT, USAGE=A1,   ACTUAL=A1,   $
  FIELD=GLOBAL_VALUE, ALIAS=GLOBAL_VALUE, USAGE=A512, ACTUAL=A512, $
SEGMENT=SUB1, SEGTYPE=S0, $
  FIELD=SUB1, ALIAS=SUBSCRIPT,  USAGE=A245, ACTUAL=A245, $
SEGMENT=SUB2, SEGTYPE=S0, $
  FIELD=SUB2, ALIAS=SUBSCRIPT,  USAGE=A245, ACTUAL=A245, $
SEGMENT=SUB3, SEGTYPE=S0, $
  FIELD=SUB3, ALIAS=SUBSCRIPT,  USAGE=A245, ACTUAL=A245, $
  FIELD=VAL3, ALIAS=VAL3,       USAGE=A512, ACTUAL=A512, $
```

Generated Clinic Access File

```
UCI=MUA, VOLUMESET=RAB, GLOBAL=CLINIC, CARDINALITY=6, WRITE=NO, $
SEGNAM=GLOBAL, DELIMITER=,$
SEGNAM=SUB3, DELIMITER=,$
```

Unlike other databases, Mumps does not provide column names. Therefore, the API does not return column names. All columns are assigned names prefixed with SUB and numbered sequentially (SUB1, SUB2, ..., SUBn). You can change the field names generated by CREATE SYNONYM to make them more descriptive. Data value fields are placed in the segment associated with the subscript. Data values might contain a sequence of variable length fields delimited by special characters. These delimiting characters are described in the Access File and are associated with subscript segments. Redefinition of the data can be done to further facilitate reporting. This is also a manual update of the synonym. In this example, the previous Master File synonym Clinic, field VAL3 is redefined:

```
FILE=CLINIC, SUFFIX=DSM, $
SEGMENT=GLOBAL, SEGTYPE=U, $
  FIELD=DUMMY, ALIAS=SUBSCRIPT,USAGE=A1,    ACTUAL=A1,    $
  FIELD=GLOBAL_VALUE, ALIAS=GLOBAL_VALUE, USAGE=A512, ACTUAL=A512, $
SEGMENT=SUB1, SEGTYPE=S0, $
  FIELD=SUB1, ALIAS=SUBSCRIPT, USAGE=A245, ACTUAL=A245,   $
SEGMENT=SUB2, SEGTYPE=S0, $
  FIELD=SUB2, ALIAS=SUBSCRIPT, USAGE=A245, ACTUAL=A245,   $
SEGMENT=SUB3, SEGTYPE=S0, $
  FIELD=SUB3, ALIAS=SUBSCRIPT, USAGE=A245, ACTUAL=A245,   $
  FIELD=VAL3, ALIAS=VAL3,      USAGE=A512, ACTUAL=A512,   $
SEGMENT=SUB3RED, SEGTYPE=S0, PARENT=SUB3, POSITION=VAL3, $
  FIELD=VAL3A, ALIAS=VAL3A,    USAGE=A4,   ACTUAL=A4,     $
  FIELD=VAL3B, ALIAS=VAL3B,    USAGE=A4,   ACTUAL=A4,     $
  FIELD=VAL3C, ALIAS=VAL3C,    USAGE=A9,   ACTUAL=A9,     $
  FIELD=VAL3D, ALIAS=VAL3B,    USAGE=P6.2, ACUTAL=A6,     $
```

Reference: Access File Keywords

Keyword	Description
UCI	User Class Identifier for the location of the global data structure.
SEGNAM	Corresponds to the number of segments with data values in a synonym created with PARM "FULL" the Master File. If a synonym template is create the number of segments is 25.
VOLUME SET	Volume set name where the global data structures reside.
GLOBAL	Is a table name.
WRITE	Specifies whether write operations are allowed against the table.

Data Type Support

All data in DSM Mumps is represented as alphanumeric. The format of the data can be changed manually in the USAGE attribute of the Master File. For example, see the VAL3D field in the previous example.

CHAPTER 28

Using the Adapter for MySQL

Topics:

The Adapter for MySQL allows applications to access MySQL data sources. The adapter converts application requests into native MySQL statements and returns optimized answer sets to the requesting application.

Preparing the MySQL Environment

The Adapter for MySQL minimally requires the installation of the MyODBC Driver. MyODBC Driver allows you to connect to a local or remote MySQL database server.

Procedure: How to Prepare the MySQL Environment on Windows

On Windows, the MySQL environment is set up during the installation of MySQL.

Procedure: How to Prepare the MySQL Environment on UNIX

In order to connect to a MySQL data source, you need to point to the odbci.ini file, which contains the definition of that data source, using the ODBCINI environment variable:

```
ODBCINI=/users/myhome/odbc.ini
export ODBCINI
```

Configuring the Adapter for MySQL

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

The SET CONNECTION_ATTRIBUTES command allows you to declare a connection to one MySQL database server and to supply authentication attributes necessary to connect to the server.

You can declare connections to more than one MySQL database server by issuing multiple SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query referencing that connection is issued (see *Overriding the Default Connection* on page 28-6). You can include SET CONNECTION_ATTRIBUTES commands in an RPC or a server profile. The profile can be encrypted.

If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The *first* SET CONNECTION_ATTRIBUTES command sets the default MySQL database server to be used.

- If more than one SET CONNECTION_ATTRIBUTES command declares the same MySQL database server, the authentication information is taken from the *last* SET CONNECTION_ATTRIBUTES command.

Connecting to an MySQL Database Server

Using the standard rules for deploying MyODBC, the server supports connections to:

- Local MySQL database servers.
- Remote MySQL database servers. To connect to a remote MySQL database server, the odbc.ini file on the source machine must contain an entry for the MySQL data source name of the target machine and the listening process must be running on the target machine.

Authenticating a User

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

Example:

Declaring Connection Attributes

There are two methods by which a user can be authenticated when connecting to a MySQL database server:

- **Explicit.** User IDs and passwords are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.
- **Database or Password Passthru.** User ID and password received from the client application are passed to the MySQL database server for authentication.

When a client connects to the server, the user ID and password are passed to MySQL for authentication and are not authenticated by the server. To implement this type of authentication, start the server with security turned off. The server allows the client connection, and then stores an encrypted form of the client's connection message to be used for connection to a MySQL database server at anytime during the lifetime of the server agent.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Field	Description
Datasource	Valid MySQL ODBC data source name.
Security	There are two methods by which a user can be authenticated when connecting to a MySQL database server: **Explicit.** The user ID and password are explicitly specified for each connection and passed to MySQL, at connection time, for authentication. **Password Passthru.** The user ID and password received from the client application are passed to MySQL, at connection time, for authentication. This option requires that the server be started with security off.
User	Username for the MySQL authenticated login.
Password	Password that identifies the entered username.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

The values for these fields will be used to describe connection attributes for the MySQL server.

Syntax: How to Declare Connection Attributes Manually

For explicit authentication:

```
ENGINE [MYSQL] SET CONNECTION_ATTRIBUTES [connection]/userid,password
```

For Password Passthru authentication:

```
ENGINE [MYSQL] SET CONNECTION_ATTRIBUTES connection/
```

where:

`MYSQL`

Indicates the Adapter for MySQL. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

MySQL data source name as defined in the odbc.ini file

`userid`

Is the primary authorization ID by which you are known to MySQL.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the MySQL database server named TEST with an explicit user ID and password:

```
ENGINE MYSQL SET CONNECTION_ATTRIBUTES TEST/USERA,PWDA
```

The following SET CONNECTION_ATTRIBUTES command connects to the MySQL database server named TEST using Password Passthru authentication:

```
ENGINE MYSQL SET CONNECTION_ATTRIBUTES TEST/
```

Overriding the Default Connection

Once all MySQL connections to be accessed have been declared using the SET CONNECTION_ATTRIBUTES command, there are two ways to select a specific MySQL connection from the list of declared connections:

- You can select a default connection using the SET DEFAULT_CONNECTION command. If you do not issue this command, the connection name value specified in the *first* SET CONNECTION_ATTRIBUTES command is used.
- You can include the CONNECTION= attribute in the Access File of the table specified in the current SQL query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION= attribute is automatically included in the Access File. This attribute supersedes the default connection.

Syntax: How to Select a Connection to Access

```
ENGINE [MYSQL] SET DEFAULT_CONNECTION [connection]
```

where:

`MYSQL`

Indicates the Adapter for MySQL. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name specified in a previously issued SET CONNECTION_ATTRIBUTES command. If omitted, then the local database server will be set as the default. If this connection name has not been previously declared, a FOC1671 message is issued.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the last command will be the active connection name.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, a FOC1671 message is issued.

Example: Selecting a Connection to Access

The following SET DEFAULT_CONNECTION command selects the MySQL database server named TNSNAMEB as the default MySQL database server:

```
ENGINE MYSQL SET DEFAULT_CONNECTION datasource_name
```

Note: You must have previously issued a SET CONNECTION_ATTRIBUTES command for the datasource_name

Controlling Connection Scope

This topic explains how to set the scope of logical units of work using adapters. This is accomplished by the SET AUTODISCONNECT command.

A connection occurs at the first interaction with the declared database server.

Syntax: How to Control the Connection Scope

```
ENGINE [MYSQL] SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

`MYSQL`

Indicates the Adapter for MySQL. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing MySQL Metadata

This topic describes how to use CREATE SYNONYM for MySQL data sources. It also describes MySQL data type support.

Creating Synonyms

Synonyms define unique names (or aliases) for each MySQL table or view that is accessible from the server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.
3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

   Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

   - **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
   - **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.
4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.

Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS MYSQL [AT connection|AT '']
[NOCOLS]
END
```

where:

`app`

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

`synonym`

Is an alias for the data source (maximum 64 characters).

`table_view`

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

`MYSQL`

Indicates the Data Adapter for MySQL.

`AT connection`

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`AT ' '`

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local database server.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS MYSQL AT TEST NOCOLS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=MYSQL ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4,     I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=TEST,KEYS=1, WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the MySQL tablename. The tablename may include owner (schema) name. For example, `TABLENAME=[`*`owner.`*`]`*`tablename`*
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following chart provides information about the default mapping of MySQL data types to server data types:

MySQL Data Type	Data Type		Remarks
	USAGE	**ACTUAL**	
TINYINT[(m)] m in signed range (-128...127), unsigned range (0...255)	I6	I4	BIT and BOOL are synonyms for TINYINT(1)
SMALLINT[(m)] m in signed range (-32768...32767) unsigned range (0...65535)	I6	I4	
MEDIUMINT[(m)] m in signed range (-8388608...8388607) unsigned range (0...16777215)	I11	I4	
INT[(m)] m in signed range (-2147483648...2147483647) unsigned range (0...4294967295)	I11	I4	
BIGINT[(m)] signed range is (-9223372036854775808... 9223372036854775807) unsigned range is (0...18446744073709551615)	P20	P10	
FLOAT(m,[d]) m in (1...24) or m in (25...53) will automatically create a double	D20.2	D8	REAL & DOUBLE PRECISION are synonyms of DOUBLE
DECIMAL [(m[d])] m (25...53) DEC (10,0)- default value	P11	P8	NUMERIC is a synonym for DECIMAL
DATE	YYMD	DATE	
DATETIME	HYYMDS	HYYMDS	

MySQL Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
TIMESTAMP	HYYMDS	HYYMDS	
TIME	HHIS	HHIS	
YEAR	I6	I4	
CHAR(*m*) NATIONAL CHAR (*m*)	A*n*	A*n*	*m* is an integer between 0 and 255 *n* is an integer between 1 and 255
VARCHAR (*m*) NATIONAL VARCHAR (*m*)	A*n*	A*n*	*m* is an integer between 1 and 255 *n* is an integer between 1 and 255
TINYBLOB TINYTEXT	A32767	A32767	
BLOB TEXT	A32767	A32767	
MEDIUMBLOB MEDIUMTEXT	A32767	A32767	
LONGBLOB LONGTEXT	A32767	A32767	
ENUM('*value1*', '*value2*',.......) max of 65535 values	Not Supported	Not Supported	
SET ('*value1*', '*value2*',......) max of 64 members	Not Supported	Not Supported	

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [MYSQL] SET CONVERSION RESET
ENGINE [MYSQL] SET CONVERSION format RESET
ENGINE [MYSQL] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [MYSQL] SET CONVERSION format [PRECISION MAX]
```

where:

`MYSQL`

Indicates the Adapter for MySQL. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER and SMALLINT columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

precision

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

scale

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2, except for DECIMAL, where both the precision and scale are derived from the decimal column precision and scale.

If the scale is not required, you must set scale to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE MYSQL SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE MYSQL SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE MYSQL SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE MYSQL SET CONVERSION RESET
```

CLOB Activation

The default mapping for MySQL data types CHAR, VARCHAR, VARCHAR2, and RAW is the data type ALPHA. The ALPHA data type maximum supported length is 4096 characters for TABLE/MODIFY and 32768 characters for API applications. It is possible to perform SELECT, INSERT, and UPDATE on columns longer than 256 with these data types without using the server data type CLOB.

Syntax: How to Set Server CLOB Activation

To activate the support for the server data type CLOB, you must issue the following command in one of the supported server profiles:

```
ENGINE [MYSQL] SET CONVERSION LONGCHAR TEXT
```

where:

`MYSQL`

Indicates the Adapter for MySQL. You can omit this value if you previously issued the SET SQLENGINE command.

`TEXT`

Activates support of long character columns as Character Large Objects. ALPHA is the default value.

Customizing the Adapter for MySQL Environment

This topic describes operational and performance adapter settings including optimization and bulk operations.

PASSRECS

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Set PASSRECS

```
ENGINE [MYSQL] SET PASSRECS {ON|OFF}
```

where:

`MYSQL`

Indicates the Adapter for MySQL. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

In this section:

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

How to:

Optimize Requests

Example:

SQL Requests Passed to the RDBMS With Optimization OFF

SQL Requests Passed to the RDBMS With Optimization ON

Reference:

SQL Generation in Optimization Examples

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [MYSQL] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`MYSQL`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

`OFF` instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

`ON` instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

`FOCUS` passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

`SQL` passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql MYSQL set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql MYSQL set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [MYSQL] SET OPTIFTHENELSE {ON|OFF}
```

where:

`MYSQL`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL MYSQL SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL MYSQL SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL MYSQL SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

CHAPTER 29

Using the Adapter for Nucleus

Topics:

The Adapter for Nucleus allows applications to access Nucleus data sources. The adapter converts application requests into native Nucleus statements and returns optimized answer sets to the requesting application.

Preparing the Nucleus Environment

In order to use the Adapter for Nucleus, the Nucleus database and ODBC must be installed and configured and the path to the Nucleus ODBC libraries directory must be added to SYSTEM LIBRARY PATH. See the SandTechnology Nucleus® documentation for more information about requirements for ODBC installation and configuration.

Procedure: How to Set Up the Environment on Windows

On Windows, the Nucleus environment is set up during the installation of Nucleus.

Procedure: How to Set Up the Environment on UNIX

You can access Nucleus using the NUCINI environment variable and one of the following:

- $HOME/.odbc.ini file
- ODBCINI environment variable.

Point the NUCINI variable to the directory where SandTechnology ODBC® is installed. For example:

```
NUCINI=/usr/nucleus/odbc
export NUCINI
```

The ODBCINI variable holds the absolute path to the *.ini file, which describes the Nucleus data sources. For example:

```
ODBCINI=/usr/ibiuser/odbc/nucodbc.ini
export ODBCINI
```

Configuring the Adapter for Nucleus

In this section:

Declaring Connection Attributes

Overriding the Default Connection

Controlling the Connection Scope

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

Example:

Declaring Connection Attributes

In order to connect to a Nucleus database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Nucleus database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Nucleus Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	Description
Data source	Valid Nucleus Data source Name (DSN). There is no default DSN; a value must be entered.
User	Valid Nucleus user name.
Password	Password that identifies the entered user name.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

```
ENGINE [SQLNUC] SET CONNECTION_ATTRIBUTES [connection]/userid,password
```

where:

`SQLNUC`

Indicates the Adapter for Nucleus. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the name of the Nucleus Data Source Name (DSN) you wish to access. It must match an entry in the .odbc.ini.

`userid`

Is the primary authorization ID by which you are known to Nucleus.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Nucleus database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLNUC SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLNUC] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLNUC`

Indicates the Adapter for Nucleus. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Nucleus database server named SAMPLENAME as the default Nucleus database server:

```
ENGINE SQLNUC SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLNUC] SET AUTODISCONNECT ON {FIN|COMMIT|COMMAND}
```

where:

`SQLNUC`

Indicates the Adapter for Nucleus. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing Nucleus Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Nucleus data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Nucleus table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.
10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value.

11. Complete your table or view selection:

To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

Synonyms are created and added under the specified application directory.

A status window displays the message:

```
All Synonyms Created Successfully
```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)

Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLNUC [AT connection]
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLNUC

Indicates the Data Adapter for Nucleus.

`AT connection`

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLNUC AT DSN_A
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLNUC ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the Nucleus table. The value assigned to this attributes can include the name of the owner (also known as schema) as follows: `TABLENAME=[owner.] table`
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=connection`
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following chart provides information about the default mapping of Nucleus data types to server data types:

Nucleus Data Type	Data Type		Remarks
	USAGE	**ACTUAL**	
CHAR (n)	An	An	n ranges in 1...2040
VARCHAR (n)	An	An	n ranges in 1...2040
SMALLINT (-32768... 32767)	I6	I4	
INTEGER (-2^{31}... 2^{31} - 1)	I11	I4	2^{31} = 2,147,483,648.
NUMERIC, DECIMAL (p,s)	D20.2	D8	See *Customizing the Nucleus Environment* on page 29-15.
FLOAT	D20.2	D8	
REAL	D20.2	D8	
DOUBLE PRECISION	D20.2	D8	

Changing the Precision and Scale of Numeric Columns

How to:

Override Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLNUC] SET CONVERSION RESET
ENGINE [SQLNUC] SET CONVERSION format RESET
ENGINE [SQLNUC] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLNUC] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLNUC`

Indicates the Adapter for Nucleus. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

precision

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

scale

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLNUC SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLNUC SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLNUC SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLNUC SET CONVERSION RESET
```

Customizing the Nucleus Environment

The Adapter for Nucleus provides several parameters for customizing the environment and optimizing performance.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLNUC] SET PASSRECS {ON|OFF}
```

where:

`SQLNUC`

Indicates the Adapter for Nucleus. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLNUC] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLNUC`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

OFF instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

ON instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

FOCUS passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

SQL passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLNUC set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLNUC set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLNUC] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLNUC`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLNUC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLNUC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLNUC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```
- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

CHAPTER 30

Using the Adapter for ODBC

Topics:

The Adapter for ODBC allows applications to access ODBC data sources. The adapter converts application requests into native ODBC statements and returns optimized answer sets to the requesting application.

Preparing the ODBC Environment

In order to use the Adapter for ODBC, you must install the Microsoft 32-bit ODBC Driver Manager on the Windows system. The Adapter for ODBC accesses all ODBC drivers installed and defined in the 32-bit ODBC Driver Manager.

Configuring the Adapter for ODBC

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to the ODBC database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one ODBC database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the ODBC Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	Description
Data source	Data source name configured using the ODBC Driver Manager.
Security	There are two methods by which a use can be authenticated when connecting to an ODBC data source: **Explicit.** The user ID and password are explicitly specified for each connection pass to the ODBC data source, at connection time, for authentication. **Trusted.** The adapter connects to the ODBC data source as a Windows login using the credentials of the operating system user impersonated by the server data access agent.
User	Primary authorization ID by which you are known to an ODBC data source.
Password	Password associated with the primary authorization ID.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to ODBC, at connection time, for authentication.

```
ENGINE [SQLODBC] SET CONNECTION_ATTRIBUTES [datasource]/userid,password
```

Trusted authentication. The adapter connects to ODBC as a Windows login using the credentials of the Windows user impersonated by the server data access agent.

```
ENGINE [SQLODBC] SET CONNECTION_ATTRIBUTES [datasource]/,
```

where:

`SQLODBC`

Indicates the Adapter for ODBC. You can omit this value if you previously issued the SET SQLENGINE command.

`datasource`

Is the name of the ODBC data source you wish to access.

`userid`

Is the primary authorization ID by which you are known to the data source.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the ODBC database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLODBC SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLODBC] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLODBC`

Indicates the Adapter for ODBC. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the ODBC database server named SAMPLENAME as the default ODBC database server:

```
ENGINE SQLODBC SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLODBC] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLODBC`

Indicates the Adapter for ODBC. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing ODBC Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the ODBC data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLODBC to identify the Adapter for ODBC.

Syntax: How to Identify the Adapter for ODBC

```
FILE[NAME]=file, SUFFIX=SQLODBC [,$]
```

where:

`file`

Is the file name for the Master File. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

`SQLODBC`

Is the value for the Adapter for ODBC.

Accessing Database Tables

If you choose to access a remote third-party table using ODBC, you must locally install the RDBMS' ODBC Driver.

The iWay Server can access third-party database tables across the network ODBC. You must provide a system data source name and possibly a user ID and/or password for the database tables you are accessing. You can define these parameters in either the server's global profile or in a user profile.

Creating Synonyms

How to:

Create a Synonym Using the Web Console

Create a Synonym Manually

Example:

Using CREATE SYNONYM

Reference:

Managing Synonyms

Access File Keywords

Synonyms define unique names (or aliases) for each ODBC table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.

Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLODBC [AT connection]
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

`table_view`

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

`SQLODBC`

Indicates the Data Adapter for ODBC.

`AT connection`

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

The following example creates a synonym, named nf29004, for the ODBC table named NF29004.

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS ODB AT DB1
```

The synonym creates the following Master File and Access File.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=ODB ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DB1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

Keyword	**Description**
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the ODBC table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: `TABLENAME=[`*`owner.`*`]`*`table`*`[`*`@databaselink`*`]`
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local ODBC database server. Absence of the CONNECTION attribute indicates access to the default database server.

`DBSPACE`	Optional keyword that indicates the storage area for the table. For example:
	`datasource.tablespace`
	`DATABASE` *`datasource`*

`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.

Keyword	Description
WRITE	Specifies whether write operations are allowed against the table.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

How to:

Override the Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override the Default Precision and Scale

```
ENGINE [SQLODBC] SET CONVERSION RESET
ENGINE [SQLODBC] SET CONVERSION format RESET
ENGINE [SQLODBC] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLODBC] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLODBC`

Indicates the Adapter for ODBC. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INTEGER columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

FLOAT which indicates that the command applies only to double precision floating point columns.

precision

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

scale

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLODBC SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLODBC SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLODBC SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLODBC SET CONVERSION RESET
```

Customizing the ODBC Environment

In this section:

Specifying a Timeout Limit

Obtaining the Number of Rows Updated or Deleted

How to:

Issue the TIMEOUT Command

Obtain the Number of Rows Updated or Deleted

The Adapter for ODBC provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to ODBC.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE [SQLODBC] SET TIMEOUT {nn|0}
```

where:

`SQLODBC`

Indicates the Adapter for ODBC. You can omit this value if you previously issued the SET SQLENGINE command.

`nn`

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLODBC] SET PASSRECS {ON|OFF}
```

where:

`SQLODBC`

Indicates the Adapter for ODBC. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

In this section:

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

How to:

Optimize Requests

Example:

SQL Requests Passed to the RDBMS With Optimization OFF

SQL Requests Passed to the RDBMS With Optimization ON

Reference:

SQL Generation in Optimization Examples

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLODBC] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLODBC`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

`setting`

Is the optimization setting. Valid values are as follows:

`OFF` instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

`ON` instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

`FOCUS` passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

`SQL` passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLODBC set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLODBC set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLODBC] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLODBC`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLODBC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLODBC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLODBC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

CHAPTER 31

Using the Adapter for Oracle

Topics:

The Adapter for Oracle allows applications to access Oracle data sources. The adapter converts data or application requests into native Oracle statements and returns optimized answer sets to the requesting program.

Preparing the Oracle Environment

The Adapter for Oracle minimally requires the installation of the Oracle Client. The Oracle Client allows you to connect to a local or remote Oracle database server.

Make sure that the client shared library, libclntch, was generated on UNIX, OS/390 and z/OS USS.

Procedure: How to Prepare the Oracle Environment on Windows

On Windows, the Oracle environment is set up during the installation of Oracle.

Procedure: How to Prepare the Oracle Environment on UNIX

1. Specify the Oracle Database Instance to access using the UNIX environment variable $ORACLE_SID. For example:

   ```
   ORACLE_SID=orac
   export ORACLE_SID
   ```

2. Specify the location of the Oracle database you wish to access using the UNIX environment variable $ORACLE_HOME. For example, to set the home directory for the Oracle software to /usr/oracle/9.0.1, specify:

   ```
   ORACLE_HOME=/usr/oracle/orac
   export ORACLE_HOME
   ```

3. Specify the path to the Oracle shared library using the UNIX environment variable $LD_LIBRARY_PATH. This must be done only for an unsecured server. For example:

   ```
   LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
   export LD_LIBRARY_PATH
   ```

Procedure: How to Prepare the Oracle Environment on OS/390 and z/OS

The configuration for Oracle on OS/390 and z/OS requires that EDASTART JCL is used to allocate Oracle specific variables. This can be done using the EDAENV ddname in the JCL:

```
ORACLE_SID=ORAT
ORACLE_HOME=/usr/lpp/orac
LIBPATH=$ORACLE_HOME/lib
```

Syntax: How to Prepare the Oracle Environment on OpenVMS

When a site creates an Oracle SID, the Oracle software automatically generates a DCL setup script so users and products such as iWay can properly invoke the environment.

The specification for the location of the Oracle setup file is:

```
$@disk:[oracle_root.DB_oraclesiddb]ORAUSER_oraclesiddb.COM
```

where:

disk

Is the disk on which Oracle is installed.

oracle_root

Is the root directory of the Oracle installation.

oraclesiddb

Is the name of the Oracle database. This name does not need to match the Oracle SID.

EDAENV.COM is an optional OpenVMS iWay file that is invoked at server start up to issue DCL commands such as calls to DBMS setup files. Due to Oracle's use of symbols and job logicals, the proper way to invoke Oracle's setup file is only using the EDAENV.COM file.

By default, EDAENV.COM does not exist and must be manually created in the [.BIN] directory of EDACONF. In this case, and in its simplest form, EDAENV.COM will contain a single line of syntax that specifies the call to the Oracle setup file.

Example: Specifying the $ORACLE_SID on OpenVMS

```
$@3$DKB0:[ORACLE.DB_ORAC]ORAUSER_ORAC.COM
```

Connecting to a Remote Oracle Database Server

Using the standard rules for deploying the Oracle Client, the server supports connections to:

- Local Oracle database servers.
- Remote Oracle database servers. To connect to a remote Oracle database server, the Oracle tnsnames.ora file on the source machine must contain an entry pointing to the target machine and the listening process must be running on the target machine.

Once you are connected to an Oracle database server, that server may define Oracle DATABASE LINKs that can be used to access Oracle tables on other Oracle database servers.

XA Support

Read/write applications accessing Oracle data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see Appendix A, *XA Support*.

Configuring the Adapter for Oracle

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

Example:

Declaring Connection Attributes

In order to connect to an Oracle database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Oracle database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Oracle Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.

2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.

3. Provide valid values for all input fields.

Attribute	**Description**
TNS name	Service (TNS) name used as a connect descriptor to an Oracle database server across the network. It must point to a valid entry in the tnsnames file. If you plan to have only one connection to Oracle, this parameter is optional. If not specified, the local database server serves as the default connection.
Security	There are three methods by which a user can be authenticated when connecting to an Oracle database server: **Explicit.** The user ID and password are explicitly specified for each connection and passed to Oracle, at connection time, for authentication. **Password Passthru.** The user ID and password received from the client application are passed to Oracle, at connection time, for authentication. This option requires that the server be started with security off. **Trusted.** The adapter connects to Oracle as an operating system login using the credentials of the operating system user impersonated by the server data access agent.
User	Primary authorization ID by which you are known to Oracle.
Password	Password associated with the primary authorization ID.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Oracle, at connection time, for authentication.

```
ENGINE [SQLORA] SET CONNECTION_ATTRIBUTES [connection]/userid,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Oracle, at connection time, for authentication. This option requires that the server be started with security off.

```
ENGINE [SQLORA] SET CONNECTION_ATTRIBUTES [connection]/
```

Trusted authentication. The adapter connects to Oracle as an operating system login using the credentials of the operating system user impersonated by the server data access agent.

```
ENGINE [SQLORA] SET CONNECTION_ATTRIBUTES [connection]/,
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is a logical name (or service (TNS) name) used to identify this particular set of attributes.

If you plan to have only a local connection to Oracle, this parameter is optional. If not specified, the local database server serves as the default connection.

`userid`

Is the primary authorization ID by which you are known to Oracle.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allow the application to access the Oracle database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Oracle database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

The following SET CONNECTION_ATTRIBUTES command connects to a local Oracle database server using operating system authentication:

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES /,
```

Connection Syntax in Earlier Server Releases

The following table lists the syntax for the SET CONNECTION_ATTRIBUTES command according to the different server releases:

Server Release	**Explicit Authentication**	**Password Passthru Authentication**	**Trusted Authentication**
4.2.1	SQL SQLORA SET USER *userid*/*password*[@*tns_name*]	No SET USER command in profile permitted.	SQL SQLORA SET USER/""
4.3.1	SQL SQLORA SET USER *userid*/*password*[@*tns_name*]	SQL SQLORA SET USER [@*tns_name*]	SQL SQLORA SET USER /[@*tns_name*]
5.1.0	ENGINE SQLORA SET CONNECTION_ATTRIBUTES [*tns_name*]/*userid*,*password*	ENGINE SQLORA SET CONNECTION_ATTRIBUTES [*tns_name*]/	ENGINE SQLORA SET CONNECTION_ATTRIBUTES [*tns_name*]/,

Overriding the Default Connection

How to:

Change the Default Connection

Example:

Selecting the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLORA] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Oracle database server named SAMPLENAME as the default Oracle database server:

```
ENGINE SQLORA SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLORA] SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing Oracle Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Oracle data types.

Creating Synonyms

How to:

Create a Synonym Using the Web Console

Create a Synonym Manually

Example:

Using CREATE SYNONYM

Reference:

Managing Synonyms

Access File Keywords

Synonyms define unique names (or aliases) for each Oracle table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.
10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value. See *Controlling the Mapping of Large Character Data Types* for more information.

11. Complete your table or view selection:

To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

Synonyms are created and added under the specified application directory.

A status window displays the message:

```
All Synonyms Created Successfully
```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)

Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view_syn DBMS SQLORA [AT connection]
                                                          [AT '']
DBSYNONYM [NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view_syn

Is the name for the table, view, or native Oracle synonym. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLORA

Indicates the Adapter for Oracle.

`AT` *`connection`*

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`AT ' '`

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local database server.

`DBSYNONYM`

Specifies that the data source is a native Oracle synonym. This entry is required when the data source type is synonym.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLORA AT ORA901 NOCOLS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLORA ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=ORA901,KEYS=1, WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the Oracle table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: `TABLENAME=[`*`owner.`*`]`*`table`*`[`*`@databaselink`*`]`
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local Oracle database server. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Accessing Multiple Database Servers in One SQL Request

To access a remote Oracle table using DATABASE LINKs, the following conditions must exist:

- The Oracle database server to which you are connected must have a valid DATABASE LINK defined.
- The TABLENAME attribute in the Access File for the Oracle table to be queried must have the following format:

  ```
  TABLENAME=[owner.]table@databaselink
  ```

  where:

  `owner`

  Is the user ID by default. It can consist of a maximum of 30 characters. Oracle prefers that the value be uppercase.

  `table`

  Is the name of the table or view. It can consist of a maximum of 30 characters.

`databaselink`

Is the valid DATABASE LINK name defined in the currently connected Oracle database server.

This format for TABLENAME can be placed in the Access File manually or using the CREATE SYNONYM command. For example:

```
CREATE SYNONYM filename FOR owner.table@databaselink DBMS SQLORA
```

Once you have met the above conditions, all requests for the table will be processed on the remote Oracle database server specified using the DATABASE LINK name. Using this method is another way to access multiple remote servers in one SQL request.

Data Type Support

The following table lists how the server maps Oracle data types. Note that you can:

- Control the mapping of large character data types.
- Change the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Oracle Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
CHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 2000
NCHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 2000
VARCHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 4000
VARCHAR2 (*n*)	A*n*	A*n*	*n* is an integer between 1 and 4000
NVARCHAR2 (*n*)	A*n*	A*n*	*n* is an integer between 1 and 4000
RAW (*n*)	A*m*	A*m*	*n* is an integer between 1 and 2000 m = 2 * n
LONG	TX50	TX	Supported through iWay API
LONG RAW	BLOB	BLOB	Supported through iWay API
BLOB			Not supported
CLOB	TX50	TX	Server supports CLOB through the OCI interface, with ora_oci=y set in edaserve.cfg
NCLOB			Not supported

Oracle Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
ROWID	A18	A18	
UROWID			Not supported
MLSLABEL			Not supported
BFILE			Not supported
DATE	HYYMD	HYYMD	
TIMESTAMP (*fractional_seconds_precision*) WITH LOCAL TIME ZONE	HYYMDS HYYMDs HYYMDm	HYYMDS HYYMDs HYYMDm	*fractional_seconds_precision* in (0,1,2) *fractional_seconds_precision* in (3,4,5) *fractional_seconds_precision* in (6,7,8,9) The Server supports TIMESTAMP without the TIME ZONE portion through the OCI interface: ora_oci=y in edaserve.cfg.
INTERVAL YEAR *(year_precision)* TO MONTH			Not supported
INTERVAL DAY *(day_precision)* TO SECOND *(fractional_seconds_precision)*			Not supported

Oracle Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
NUMBER (*p*, *s*)	P16	P*p,s*	*p* is an integer between 1 and 31. *s* is an integer between 0 and p.
	D8	D20.2	*p* is an integer between 32 and 37. *s* is an integer between 0 and p.
	I4	I11	*p* is 38. This value is the Oracle default. *s* is an integer between 0 and p.
INTEGER			Not an Oracle built-in data type. Stored in Oracle as NUMBER.
DECIMAL			Not an Oracle built-in data type. Stored in Oracle as NUMBER.
FLOAT DOUBLE PRECISION			Not an Oracle built-in data type. Stored in Oracle as NUMBER.

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Oracle data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Oracle Data Type	**Remarks**	**LONGCHAR ALPHA or BLOB**		**LONGCHAR TEXT**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
CHAR (*n*)	*n* is an integer between 1 and 2000	A*n*	A*n*	TX50	TX
VARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*	A*n*	TX50	TX
VARCHAR2 (*n*)	*n* is an integer between 1 and 4000	A*n*	A*n*	TX50	TX
RAW (*n*)	*n* is an integer between 1 and 2000 *m* = 2 * *n*	A*m*	A*m*	TX50	TX

Syntax: How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLORA] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`ALPHA`

Maps the Oracle data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). ALPHA is the default value.

`TEXT`

Maps the Oracle data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

`BLOB`

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Oracle data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

For OS/390 and z/OS, maps the Oracle data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Oracle data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Oracle Data Type	**Remarks**	**VARCHAR ON**		**VARCHAR OFF**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
VARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*V	A*n*V	A*n*	A*n*
VARCHAR2 (*n*)	*n* is an integer between 1 and 4000	A*n*V	A*n*V	A*n*	A*n*
NVARCHAR2 (*n*)	*n* is an integer between 1 and 4000	A*m*V	A*m*V	A*m*	A*m*

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLORA] SET VARCHAR {ON|OFF}
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Oracle data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (A*n*V).

`OFF`

Maps the Oracle data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). OFF is the default value.

Considerations for the CHAR and VARCHAR2 Data Types

Special attention must be paid to CHAR and VARCHAR2 data types. When you compare a CHAR data type column to a VARCHAR2 data type column, where the only difference is additional trailing spaces in the CHAR data type column, Oracle treats the column values as different.

The SET parameter ORACHAR lets you specify which of the two data types will be used for inserting, updating, and retrieving data.

If you create the tables outside of the server, we recommend that you use either CHAR or VARCHAR2 data types, but not both. If you create a table with both data types, you might not be able to retrieve the data you inserted due to Oracle's comparison mechanism. When inserting data into VARCHAR2 columns outside of the server, do not insert any trailing spaces.

If you use the server to generate Oracle tables and retrieve data, you will not encounter this problem, since the data type being used will be either CHAR or VARCHAR2, depending upon the ORACHAR setting.

Syntax: How to Set ORACHAR

```
ENGINE [SQLORA] SET ORACHAR {FIX|VAR}
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`FIX`

Uses the CHAR data type.

`VAR`

Uses the VARCHAR2 data type. VAR is the default value.

Changing the Precision and Scale of Numeric Columns

How to:

Override Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLORA] SET CONVERSION RESET
ENGINE [SQLORA] SET CONVERSION format RESET
ENGINE [SQLORA] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLORA] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER fields to 7:

```
ENGINE SQLORA SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLORA SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER fields to the default:

```
ENGINE SQLORA SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLORA SET CONVERSION RESET
```

Considerations for the NUMBER Data Type

When working with the NUMBER data type, where the precision is between 32 and 37, by default the NUMBER data type is mapped to the server data type double float (D), with a precision of 20 and a scale of 2.

When working with the NUMBER data type, where the precision is 38, by default the NUMBER data type is mapped to the server data type Integer (I), with a display length of 11.

To override the precision of the NUMBER data type, where the precision is between 32 and 38, use the ORANUMBER setting.

Syntax: How to Set ORANUMBER

```
ENGINE [SQLORA] SET ORANUMBER {COMPAT|DECIMAL}
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`COMPAT`

Indicates that the NUMBER data type will be mapped to the server data type double float (D), with a precision of 20 and a scale of 2. COMPAT is the default value.

`DECIMAL`

Indicates that the NUMBER data type will be mapped to the server data type decimal (P), with a precision of 33.

Customizing the Oracle Environment

In this section:

The Adapter for Oracle provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Designating a Default Tablespace

You can use the SET DBSPACE command to designate a default tablespace for tables you create. For the duration of the session, the adapter places these tables in the Oracle tablespace that you identify with the SET DBSPACE command. If the SET DBSPACE command is not used, Oracle uses the default tablespace for the connected user.

Syntax: How to Set DBSPACE

```
ENGINE [SQLORA] SET DBSPACE tablespace
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`tablespace`

Is a valid tablespace in the database.

Note: This command will affect only CREATE FILE and HOLD FORMAT SQLORA requests issued by Table Services. It does not affect Passthru CREATE TABLE commands.

Overriding Default Parameters for Index Space

You can use the SET IXSPACE command to override the default parameters for the index space implicitly created by the CREATE FILE and HOLD FORMAT commands.

Syntax: How to Set IXSPACE

```
ENGINE [SQLORA] SET IXSPACE [index-spec]
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`index-spec`

Is the portion of the Oracle CREATE INDEX statement that defines the parameters for the index. It can consist of up to 94 bytes of valid Oracle index space parameters. To reset the index space parameters to their default values, issue the SET IXSPACE command with no parameters.

Note: Refer to the Oracle documentation for more information on this command.

The long form of SQL Passthru syntax for commands exceeding one line is:

```
ENGINE SQLORA
SET IXSPACE index-spec
END
```

For example, to specify the NOSORT, NOLOGGING, and TABLESPACE portions of the Oracle CREATE INDEX statement, enter the following commands:

```
ENGINE SQLORA
SET IXSPACE NOSORT NOLOGGING
TABLESPACE TEMP
END
```

Note: This command will only affect CREATE INDEX requests issued by CREATE FILE and HOLD FORMAT SQLORA commands. It does not affect Passthru CREATE INDEX commands, for example:

```
ENGINE SQLORA SET IXSPACE TABLESPACE tablespace_name
TABLE FILE table_name
PRINT *
ON TABLE HOLD AS file_name FORMAT SQLORA
END
```

Activating NONBLOCK Mode

The Adapter for Oracle has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE [SQLORA] SET NONBLOCK {0|n}
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLORA] SET PASSRECS {ON|OFF}
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Specifying the Maximum Number of Parameters for Stored Procedures

You can use the SET SPMAXPRM command to specify the maximum number of input parameters that can be associated with any Oracle stored procedure.

Syntax: How to Set SPMAXPRM

```
ENGINE [SQLORA] SET SPMAXPRM nnn
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

nnn

Is the maximum number of parameters that can be passed to any stored procedure available to be run in this client session. 256 is the default value.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Oracle.

Syntax: How to Issue TIMEOUT Command

```
ENGINE [SQLORA] SET TIMEOUT {nn|0}
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Optimization Settings

In this section:

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

How to:

Optimize Requests

Example:

SQL Requests Passed to the RDBMS With Optimization OFF

SQL Requests Passed to the RDBMS With Optimization ON

Reference:

SQL Generation in Optimization Examples

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLORA] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLORA`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

`OFF` instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

`ON` instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

`FOCUS` passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

`SQL` passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLORA set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLORA set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLORA] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLORA`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLORA SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLORA SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLORA SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```
- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Specifying Block Size for Retrieval Processing

How to:

Specify the Block Size for Array Retrieval

Specify the Block Size for Insert Processing

The Adapter for Oracle supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Specify the Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE [SQLORA] SET FETCHSIZE n
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Syntax: How to Specify the Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE [SQLORA] SET INSERTSIZE n
```

where:

`SQLORA`

Indicates the Adapter for Oracle. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Improving Efficiency With Aggregate Awareness

Aggregate awareness substantially improves the efficiency of queries.

For details about this feature, see Appendix B, *Aggregate Awareness Support*.

Syntax: How to Set Aggregate Awareness

```
SET AGGREGATE_AWARENESS {FRESHONLY|OLD_OK|OFF}
```

where:

`FRESHONLY`
: Sets different values for the parameters associated with each RDBMS.

`OLD_OK`
: Sets different values for the parameters associated with each RDBMS.

`OFF`
: If no option is selected, the behavior of the target RDBMS is determined by the database configuration options.There is no default for this setting.

For details about adapter-specific settings, see *Usage Notes for Aggregate Awareness* on page B-3.

Calling an Oracle Stored Procedure Using SQL Passthru

Using SQL Passthru is supported for Oracle stored procedures. These procedures need to be developed within Oracle using the CREATE PROCEDURE command.

Example: Calling a Stored Procedure

The supported syntax to call a stored procedure is shown below. It is recommended that you use the syntax below instead of the previously supported CALLORA syntax.

```
ENGINE SQLORA
EX SAMPLE PARM1,PARM2,PARM3...;
TABLE FILE SQLOUT
END
```

The server supports invocation of stored procedures written according to the following rules:

- Only scalar input parameters (IN or IN OUT) are allowed, but not required.
- Output parameters are not allowed.
- A cursor must be defined with:
    - The TYPE statement in a PACKAGE or PROCEDURE.
    - An associated record layout of the answer set to be returned.
- The cursor must be opened in the procedure.
- No fetching is allowed in the stored procedure. The Adapter for Oracle fetches the answer set.
- Any messages must be issued using the RAISE APPLICATION ERROR method.

Note: Any application error that is issued by the stored procedure is available in the server variable &ORAMSGTXT.

Example: Oracle Stored Procedure

```
CREATE OR REPLACE PACKAGE pack1 AS
TYPE nfrectype IS RECORD (
employee NF29005.EMPLOYEE_ID5%TYPE,
ssn5     NF29005.SSN5%TYPE,
l_name   NF29005.LAST_NAME5%TYPE,
f_name   NF29005.FIRST_NAME5%TYPE,
birthday NF29005.BIRTHDATE5%TYPE,
salary   NF29005.SALARY5%TYPE,
joblevel NF29005.JOB_LEVEL5%TYPE);
TYPE nfcurtype IS REF CURSOR RETURN nfrectype ;
PROCEDURE proc1(c_saltable IN OUT nfcurtype);
END pack1 ;
/
CREATE OR REPLACE PACKAGE BODY pack1 AS
PROCEDURE proc1 (c_saltable IN OUT nfcurtype)
IS
BEGIN
OPEN c_saltable FOR SELECT
EMPLOYEE_ID5,SSN5,LAST_NAME5,FIRST_NAME5,BIRTHDAT
E5,SALARY5,JOB_LEVEL5 FROM NF29005;
END proc1 ;   -- end of procedure
END pack1; -- end of package body
/
```

CHAPTER 32

Using the Adapter for Oracle E-Business Suite

Topics:

The Adapter for Oracle E-Business Suite is a read-only adapter providing security integration for reporting against Oracle E-Business Suite databases. This adapter supports user authentication and data access rules that are defined at the metadata and data access layer. Specific integrated data security is available for:

- Table and View Restrictions by Responsibility ID
- View Security by Set of Books ID
- View Security by Operating Unit

Using this adapter, you can access data with the following security models:

- Oracle Applications Security
- Union of Responsibilities Security
- Total Access Security

Preparing the Oracle E-Business Suite Environment

Before adding and Oracle E-Business Suite, you must prepare the environment. The preparation steps are:

- configuring the Adapter for Oracle
- defining connections to the Oracle E-Business Suite
- customizing the General Ledger Security Package

Configuring the Adapter for Oracle

To configure the Adapter for Oracle, follow the instructions in *Oracle*. This adapter is used to support connectivity to the Oracle E-Business Suite by the application adapter.

Defining Connections to the Oracle E-Business Suite

To add a connection to the Oracle database, follow the instructions in *Oracle*. When defining a connection to the Oracle E-Business Suite database, you must use the delivered database APPS user ID.

The APPS user ID is the database ID, not an application user ID. Privileges assigned to the APPS user ID enable the enforcement of data security rules when accessing data. Functions performed by this ID include authentication, setting the application user system context, data selection, and accessing views with row-level data restrictions.

Customizing the General Ledger Security Package

In order to execute views that contain calls to the Oracle E-Business Suite General Ledger Security Package (APPS.GL_SECURITY_PKG), the package must be customized to enable third party access. This package is called from within many General Ledger Business Views and performs the following functions:

- Updates temporary flex field tables.
- Uses the temporary tables in DDL of the views.

Procedure: How to Update the General Ledger Security Package

To update the General Ledger Package:

1. Use the Oracle Enterprise Management Console (or equivalent) to locate the package body GL_SECURITY_PKG found in the APPS schema.
2. Select *Edit>View* to modify this package body.
3. Locate the following syntax in the *init* procedure:

```
IF      (( instrb(program_name,'dis',-1,1) =  0 )       AND
  ( nvl(l_module,'XXX')          <> 'Discoverer4' )    AND
  ( nvl(l_gl_bis_disco_flag,'N')    <> 'Y' )           THEN
    return;
END IF;
```

4. Edit the syntax to include a new test on the program name, as follows:

```
IF      (( instrb(program_name,'dis',-1,1) =  0 )       OR
   ( instrb(program_name,'tsc',-1,1) =  0 ))           AND
   ( nvl(l_module,'XXX')         <> 'Discoverer4' )    AND
   ( nvl(l_gl_bis_disco_flag,'N')   <> 'Y' )           THEN
     return;
END IF;
```

   This new syntax adds the tscom3.exe agent to the list of acceptable programs for the init procedure.

Data Access and Security

Oracle E-Business Suite relies on a combination of database functionality (views, stored procedures, and functions) along with interface rules to enforce business logic, such as data security rules. The adapter supports this design by enforcing security embedded rules within the database and by allowing queries to be written against both restricted and unrestricted types of data. Therefore, the adapter can be used to support data access with the following security models:

- **Oracle Applications Security.** Reports may be defined to require a Responsibility ID and Application ID as part of their processing. When this is done, requests against business views containing row-level security data restrictions will display the secured and filtered result data set only.
- **Union of Responsibilities.** When reporting against tables or views found in the applications, users will be able to access the full union of the data that all of their responsibilities allow. Only when Oracle Applications Security is defined will this data be further restricted to an individual responsibility ID access profile.
- **Total Enterprise Access.** Reports will always enforce table and view restrictions based on the application access granted to a given user. For all tables, and the subset of views that do not have row-level data security rules built-in, reports written against them will provide full data access.

Reference: Data Access and Security Limitations

- Synonyms created for use with this application adapter must be created with the same name as the underlying Oracle RDBMS Table or View.
- Oracle applications provide automated row-level security against many of the database views stored in its repository. These views are secured based on the initialization of environment settings during the user login process. The database tables do not support row-level security through this model automatically. To support row-level security against database tables, you must code filter criteria into individual reports.

Configuring the Adapter for Oracle E-Business Suite

How to:

Configure the Adapter from the Web Console

Create the Adapter Control Tables from the Web Console

Configure the Remote Services Profile

Example:

Sample edasprof.prf

The process of configuring the adapter includes configuring the adapter in the Web Console, creating Oracle E-Business Suite control tables, and configuring the Remote Services Profile to call the adapter.

Procedure: How to Configure the Adapter from the Web Console

1. Start the server with Security OFF or in Web Console protect mode.
2. Add OESSEC to the APP PATH in edasprof.prf using the following syntax:

   ```
   APP PATH OESSEC
   ```

3. Add the Adapter for Oracle E-Business Suite by selecting the Oracle Applications option on the Web Console Adapters page.
4. Accept the default configuration parameter:

Parameter	Value	Description
Responsibility ID	-1	Default value (-1) indicates that all individual Responsibility ID security rules will be combined. If another value were provided, only that specific Responsibility ID would be used for data access privileges.

5. Click *Configure*. Once the adapter is configured, the connection name appears in the navigation pane under Configured.
6. Execute the security maintenance routine as described in *Maintaining Security Rules* on page 32-7. This will pull in the required Oracle E-Business Suite data security rules used during run-time data access.

Example: Sample edasprof.prf

```
.
.
.
-*
APP PATH OESSEC IBISAMP
-*
ENGINE SQLORA SET CONNECTION_ATTRIBUTES vis.ibi.com/apps,9AA42BAE9EC283A8
SET USER=ORAAPPS
ENGINE ORAAPPS SET APPS RESP_ID -1
```

Procedure: How to Create the Adapter Control Tables from the Web Console

1. Open the Procedures window in the Web Console.
2. Locate OESFILES within the IBISAMP folder under APPROOT.
3. Click on the procedure name, then select *Run* from the pop-up menu to execute the procedure.

Procedure: How to Configure the Remote Services Profile

1. Edit or create the Remote Service profile for the client node to be used for Oracle E-Business Suite data access.

   For example, if you are using the default EDASERVE node, navigate to the iWay Client Configuration directory (EDACONF) and create or edit a text file with the name edaserve.prf.

2. Add the _site_profile call to the adapter procedure, OESLOGIN.

```
<VER1>

# Site Profile: Adapter Oracle E-Business Suite
# Description:  Enables remote user authentication against
#               the Oracle E-Business Suite and passes necessary
#               parameters to set the system context for the user

# Set default responsibility ID and application ID
#   Calling application must set and validate the resp ID
#   and appl ID before passing them to the procedure. If
#   these are not validated they will still be passed, but
#   invalid data may be returned for the user. If no values
#   are set, these defaults will be used in setting the system
#   context. In which case, certain database views will be
#   unusable.

# Bind MR User to WF User
<ifdef> IBIMR_user
IBIC_user=&IBIMR_user
```

```
<endif>
<ifdef> IBIMR_pass
IBIC_pass=&IBIMR_pass
<endif>

# Responsibility ID (IBIWF_respid)
<ifndef> IBIWF_respid
IBIWF_respid=0
<endif>

# Application ID (IBIWF_applid)
<ifndef> IBIWF_applid
IBIWF_applid=0
<endif>

# Execute parameterized call to OESLOGIN

_site_profile=&_site_profile\EX OESLOGIN OES_RESP_ID=&IBIWF_respid,
_site_profile=&_site_profile\n   OES_APPL_ID=&IBIWF_applid,
   PASSTHRU=ON
_site_profile=&_site_profile\n-RUN

# Enable these &variables to be passed to the server

<SET> IBIWF_respid(PASS)
<SET> IBIWF_applid(PASS)
```

Maintaining Security Rules

The OESSEC application namespace contains FOCUS tables loaded with security settings at a specific point in time. In order to maintain current data access security rules within these FOCUS tables, you must rerun the OESFILES procedure periodically. This procedure can be rerun manually, on an as-needed basis.

Alternatively, you can schedule the execution of OESFILES on a regular basis. Sites that use ReportCaster for scheduling can execute OESFILES as a Server Procedure Schedule Task. Sites that do not use ReportCaster can use any standard operating system scheduling tool.

Syntax: How to Maintain Security Rules

Use one of the following variations to execute the procedure with command line syntax that calls the server engine:

- From the command prompt, enter:

  ```
  /ibi/srv53/wfs/bin/edastart -x "EX IBISAMP/OESFILES"
  ```

- From the FOCUS prompt, enter:

  ```
  EX EX IBISAMP/OESFILES
  ```

CHAPTER 33

Using the Adapter for PeopleSoft

Topics:

The Adapter for PeopleSoft provides data access for reporting from the following PeopleSoft Record Types: SQL tables, SQL views, and query views. The adapter fully enforces PeopleSoft Query tree and row-level security.

The adapter provides three types of services:

- Authentication services.
- PeopleSoft data source connection management services.
- Metadata administration and data access security services.

Preparing the Environment

In this section:

Software Requirements

Setting Up the PeopleSoft Access ID

Prior to configuring the Adapter for PeopleSoft, you must ensure that minimal software requirements have been met. In addition, you must define a PeopleSoft Access ID to enforce PeopleSoft security. Administrators must also review available connection options before beginning the initial configuration.

Software Requirements

The following software is required on the server to support the Adapter for PeopleSoft:

- Any PeopleSoft PeopleTools 7.x- or 8.x-based application or PeopleTools Enterprise.

  All PeopleSoft applications are built on the PeopleTools metadata platform, which is the framework for adapter integration.

- An appropriate RDBMS connectivity product on the computer where you will install the Adapter for PeopleSoft server agent, for example Oracle SQL*Net for Oracle data source connectivity. This is to provide access to a PeopleSoft data source server for operational data access.

- To support integrated authentication to PeopleSoft 8.1 or higher, Java 2 SDK version 1.3.1 or higher must be installed. Additionally, Publish and Subscribe services must be active on the PeopleSoft Application server.

Note: For information about PeopleSoft with versions of software other than those listed in this topic, contact your iWay Software representative.

Setting Up the PeopleSoft Access ID

The Adapter for PeopleSoft initially authenticates with PeopleSoft using a PeopleSoft application user ID and password. If successful, the adapter attaches to the RDBMS using an access ID of your choice. Typically, this access ID is either the default PeopleSoft access ID (often it is SYSADM), or it is an RDBMS ID dedicated to the adapter. In either case, the access ID must have read access to the following PeopleSoft tables and views:

PSDBFIELD	PSRECDEFN	PS_SCRTY_ACC_GRP
PSOPTIONS	PSRECFIELD	PS_TREE_ACCESS_VW
PSOPRCLS	PSTREEDEFN	XLATTABLE
PSOPRDEFN	PSTREENODE	PSDBFLDLABL (release 8.x)
PSPRCSRQST	PSPRCSQUE (release 8.x)	PSCLASSDEFN (release 8.x)

In addition, the access ID must have read access to any PeopleSoft Records that you wish to report against. If your PeopleSoft Records have row-level security, this ID must also have read access to the Query Security Records associated with each.

Note: The PSPRCSRQST and PSPRCSQUE tables are required only for Process Scheduler integration. If you are using this integration, these two files will also require write privileges.

Configuring the Adapter for PeopleSoft

In this section:

In order to configure PeopleSoft, you must add the underlying SQL adapter, register PeopleSoft adapter paths in the catalog path, update the server profile, configure a JSCOM3 listener to ensure password authentication for PeopleSoft 8, and add the first PeopleSoft connection.

Adding the SQL Database Adapter

Example:

Adding an Oracle 8.1 Adapter

The Adapter for PeopleSoft is an enhanced version of the underlying data source adapter. The adapter must be added into the server configuration.

When you add an SQL adapter, there is no need to provide configuration parameters since these connection parameters are stored in the server profile, which is accessible at all times to any user or application connecting to the server.

For most applications, the data access is then secured through procedure logic and operating system security. With PeopleSoft, this is not possible as there may be very large numbers of users who do not have operating system login IDs. Instead, these configuration parameters are supplied later when configuring connections to PeopleSoft data sources.

As a recommended practice when adding the new adapter, you may initially supply the configuration parameters, test data source connectivity, and then delete the connection. While not required, when you attempt to create a connection to the PeopleSoft data source this process may save significant time.

Example: Adding an Oracle 8.1 Adapter

The Web Console provides graphical point and click tools to add and test the adapter.

1. Open the Web Console.
2. Click *Data Adapters*. A new browser window opens.
3. From the Adapters tree, select *Add*, then *Oracle*, and select *version 8.1.x*.
4. Without supplying Oracle configuration parameters, click *Configure*. The Oracle 8.1 adapter is added to your server configuration.

Updating the Application Path

You must add the metadata paths for the Adapter for PeopleSoft into the cataloged application path. These directories are dynamically created during the first-time configuration. You must manually add references to the directories in the edasprof.prf file. You can edit edasprof.prf either through the Web Console *Workspace >> Edit Files* feature or through the file system using a text editor.

Syntax: How to Update the Application Path for the Adapter for PeopleSoft

Add the following code to the edasprof.prf file:

```
APP PATH snapinst snapcat
```

Updating the Workspace Service Profile

In order to support the PeopleSoft integrated security, the adapter must be called during each connection to the server. The easiest way to ensure that the adapter is always called is to add the PSLOGIN procedure as a Service Profile.

Procedure: How to Add PSLOGIN to the Workspace Service Profile

1. Open the Web Console.
2. Select *Workspace* >> *Configure*.

   The Workspace Configuration options window opens.
3. Locate and select the Service link at the top of the page. The Service configuration options appear.
4. Locate the Service profile input option and add *PSLOGIN*.
5. Click *Save*.
6. You will be prompted to restart the Workspace. Click *OK*.

Configuring PeopleSoft Password Authentication

How to:

Configure the JSCOM3 Listener

To leverage integrated authentication against PeopleSoft 8, a JSCOM3 listener must be configured using the Web Console. When completed, all user requests will pass calls to the PeopleSoft 8 Component Interface architecture for authentication. If the configuration is either not using PeopleSoft 8 or the connections are to be configured with TRUSTED authentication, then this step is not necessary.

Note: At this time, the PeopleSoft Component Interface does not support LDAP, even though the PeopleSoft Internet Architecture does read from LDAP repositories in general. This limitation means that if your PeopleSoft site uses LDAP integration for user authentication only, TRUSTED authentication is possible. To leverage your LDAP repository for WebFOCUS, use any of the standard integration methods available. For details, see the *WebFOCUS Security and Administration* manual.

Before configuring JSCOM3, the following conditions must be met:

- Ensure that the Publish and Subscribe services are running on the PeopleSoft Application Server. By turning these on, the jolt listener becomes active. Usually, these services are used for application messaging integration requirements, but the Adapter for PeopleSoft uses the same technology for authentication. No particular component interface must be accessible for the user to authenticate.

- Several API files delivered with PeopleSoft must be made available to the iWay Reporting Server directory. These files are located in the PeopleSoft file server libraries in the following locations:
  - [PeopleSoft file server]\web\PSJOA\psjoa.jar
  - [PeopleSoft file server]\web\jmac\pstools.properties

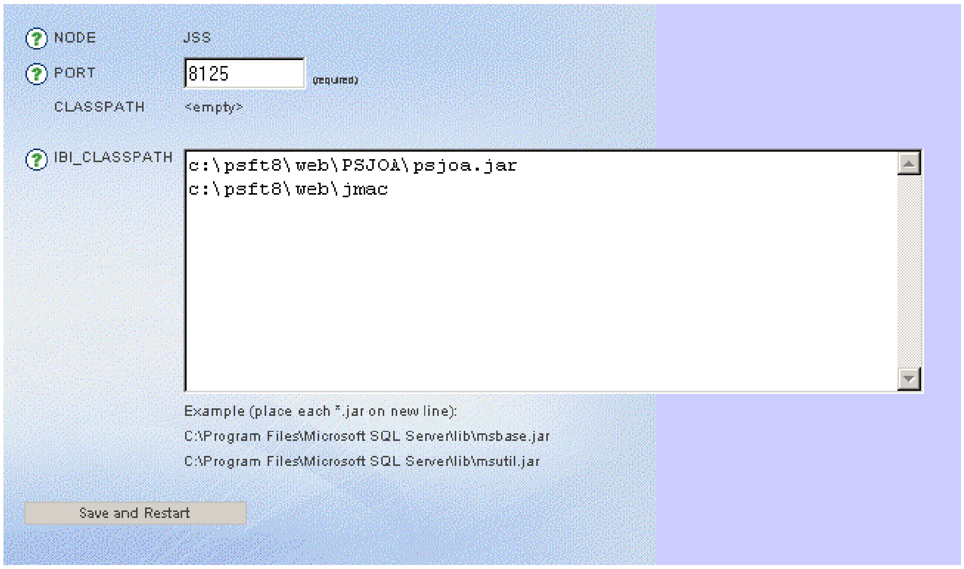
Note: With PeopleSoft 8.4 and higher, the pstools.properties file is no longer required.

If the iWay Reporting Server is installed on the same computer as your PeopleSoft Application Server, then you may configure JSCOM3 to point directly to the files. Otherwise, the files must be copied to the computer with the iWay Reporting Server. In this situation, it is recommended (but not required) that they be placed in your EDACONF\bin directory.

Procedure: How to Configure the JSCOM3 Listener

To configure the JSCOM3 listener:

1. Open up the Web Console.
2. Select *Listeners* from the menu on the left.
3. Under Special Services, click the + button next to *Add* and select *JSCOM3*.
4. Type the JSCOM3 listening port in Services. The examples all use 8103.
5. In the CLASSPATH, type the psjoa.jar file with the full path and the folder name where pstools.properties resides. If your iWay Reporting Server resides on the same computer as your PeopleSoft file server, your screen should look something like this:



NODE JSS
PORT 8125 (required)
CLASSPATH <empty>
IBI_CLASSPATH

```
c:\psft8\web\PSJOA\psjoa.jar
c:\psft8\web\jmac
```

Example (place each *.jar on new line):
C:\Program Files\Microsoft SQL Server\lib\msbase.jar
C:\Program Files\Microsoft SQL Server\lib\msutil.jar

If you do not correctly configure the pstools.properties path or if the file is not in place, the authentication will work, but you will see this message in your jscom3 log file:

```
java.lang.NullPointerException: PSProperties not loaded from file
```

Note: When upgrading the PeopleSoft Application Server after having performed these configuration steps, you must update the PeopleSoft files (psjoa.jar and pstools.properties).

These files are dependant on the minor release level.

Adding the First Connection to PeopleSoft

How to:

Add the First PeopleSoft Connection

Reference:

Administration - Create a New PeopleSoft Connection Window

PeopleSoft Connection Worksheet

Although multiple PeopleSoft connections can be created, the first connection requires slightly different management steps. The reason for this is that before any connections to PeopleSoft data sources can occur, a base layer of application metadata is created and configured. This occurs transparently for the Administrator, but is a slightly different connection management process than when adding additional connections.

To create the first connection:

- Select and type a DBA password.

  The DBA password is the 8-character alphanumeric value that is used as a metadata access key. To access the Web Console, the administrator must know this value. Users outside of the Web Console do not have metadata access without first logging in to PeopleSoft through this integration.

- Add a PeopleSoft Connection.

Procedure: How to Add the First PeopleSoft Connection

1. Open the Web Console.

   You can access the Web Console through a Web browser by opening the URL to the server's HTTP listener port. The default location on a local server is http://localhost:8121/.

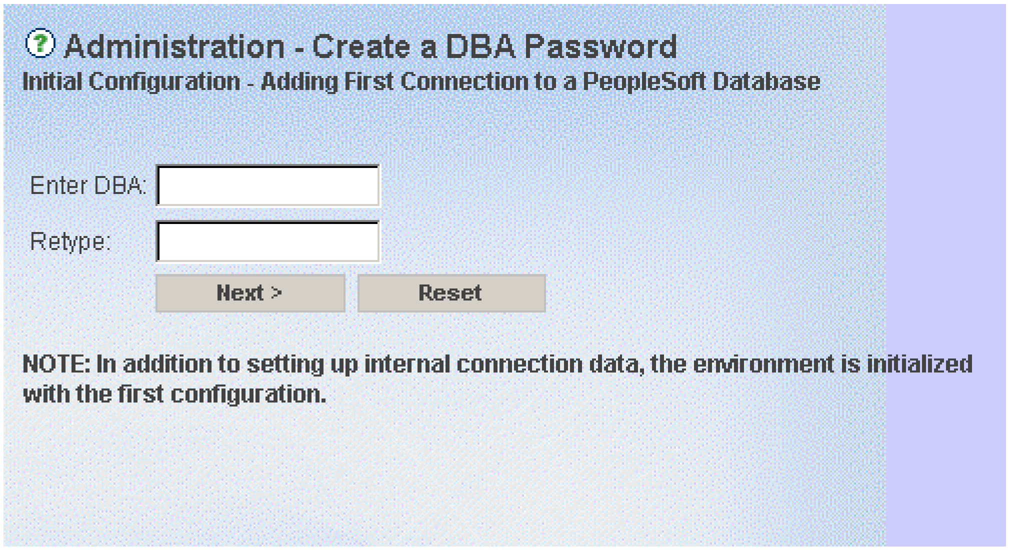
   Note: If the Web Console is set to Protected mode, you must supply a Web Console administrator ID and password to gain access.

2. Select *Data Adapters* in the main window.

   The Configuring Data Adapters window opens.

3. Select *PeopleSoft* in the ERP group under the Add folder and click *Configure*.

   The Administration - Create a DBA Password window opens.



Administration - Create a DBA Password

Initial Configuration - Adding First Connection to a PeopleSoft Database

Enter DBA:

Retype:

NOTE: In addition to setting up internal connection data, the environment is initialized with the first configuration.

4. Type an up to eight-character DBA in the Enter DBA field and the Retype field, then click *Next>*.

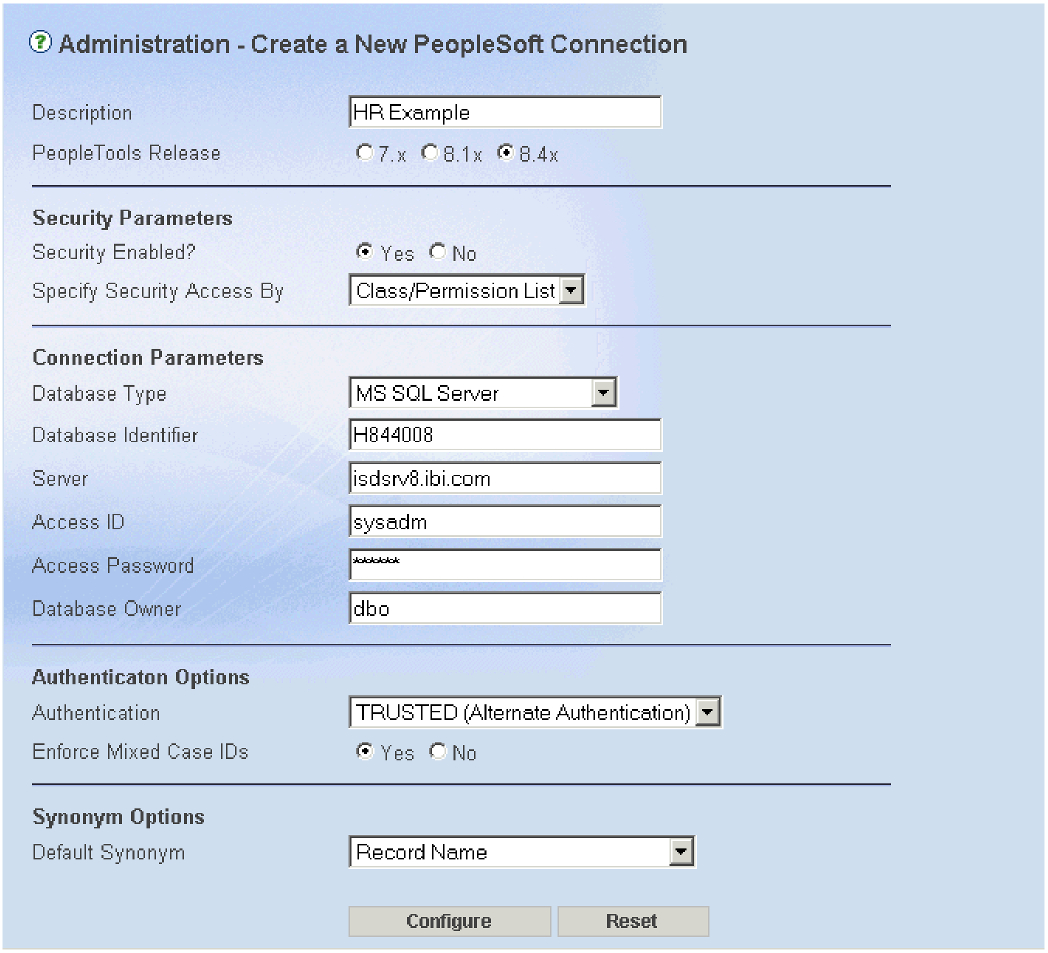
   The Administration - Create a New PeopleSoft Connection window opens.

5. Type the necessary information and click *Next>*. Most of the parameters can be changed in the future, but the connection cannot be created without initially supplying correct data source connectivity information. For details on the required information, see *Administration - Create a New PeopleSoft Connection Window* on page 33-9.

   Tip: A worksheet is provided to assist you in gathering the information you will need to type on this screen. See *PeopleSoft Connection Worksheet* on page 33-12.

   Processing may take several minutes to complete. Once completed, you may begin managing your new PeopleSoft Connection by adding synonyms and administering security access.

Reference: Administration - Create a New PeopleSoft Connection Window



Administration - Create a New PeopleSoft Connection

Description	HR Example
PeopleTools Release	7.x 8.1x [✓ 8.4x]
Security Parameters	
Security Enabled?	[✓ Yes] No
Specify Security Access By	Class/Permission List
Connection Parameters	
Database Type	MS SQL Server
Database Identifier	H844008
Server	isdsrv8.ibi.com
Access ID	sysadm
Access Password	******
Database Owner	dbo
Authenticaton Options	
Authentication	TRUSTED (Alternate Authentication)
Enforce Mixed Case IDs	[✓ Yes] No
Synonym Options	
Default Synonym	Record Name

The Administration - Create a New PeopleSoft Connection window contains the following fields/options:

`Description`

Is a description that identifies the current server configuration.

`PeopleTools Release`

Is the major PeopleSoft PeopleTools release number to access.

Security Parameters

`Security Enabled?`

Determines whether to enforce PeopleSoft data access security. The options are Yes and No. Yes is the default value.

`Specify Security Access By`

Determines the way in which the PeopleSoft administrator enables users to access the adapter and PeopleSoft data. The options are:

- **Class/Permission List.** Manages users in operating class (permission list) grouping.
- **User/Operation ID.** Manages users by individual user IDs.

Connection Parameters

`Database Type`

Is the Data source type. The options are: Oracle, MS SQL Server, and DB2 Universal Database.

`Database Identifier`

Is the data source identifier or system data source name.

If the server is co-located with the data residing in Oracle, do not type anything in the Database Identifier field.

`Server`

Is the name of the Database Server (for SQL Server only, not Oracle or DB2).

`Access ID`

Is the RDBMS login ID for data source connectivity. Typically, it is the same login ID PeopleSoft uses to access the data source. There are minimum data access requirements that the ID must have. For details, see *Setting Up the PeopleSoft Access ID* on page 33-3.

`Access Password`

Is a password associated with the access ID.

`Database Owner`

Is the data source owner identifier or the schema owner depending on your data source. The owner ID is used to fully qualify SQL requests passed to the data source. This allows multiple data source instances or schemas to exist in a single data source. Typical defaults are SYSADM on Oracle and dbo on Microsoft SQL Server.

Authentication Options

`Authentication`

Determines the authentication used in your application. The options are:

- **TRUSTED (Alternate Authentication).** Expects a valid pre-authenticated PeopleSoft user ID to be passed to the server to enable data access. This is useful with LDAP, Operating System security, or some other application that authenticates the user.
- **Password Required (Standard).** Expects a valid user ID and password before allowing data access. This is the default and is recommended for most environments.

`Enforce Mixed Case IDs`

Determines whether passwords are case-sensitive. Select *Yes* to enforce the PeopleSoft 8 use of mixed case user IDs or select *No* to revert to PeopleSoft 7.x case insensitivity.

Synonym Options

`Default Synonym`

Is the type of synonym to default to when creating new metadata. The options are: Record Name, Record Name with Prefix, Record Name with Suffix, Table Name, Table Name with Prefix, Table Name with Suffix.

Reference: PeopleSoft Connection Worksheet

Use the following worksheet to record connection configuration information and keep it for future reference.

Option Name	Response	Description
Description		Row.
DBA		8-character identifier for linking metadata to PeopleSoft security rules. Must be consistent across multiple connections.
PeopleTools Release		Major release of PeopleTools (7.x, 8.1x, or 8.4x)
Security Enabled?		Specifies whether PeopleSoft data security (row-level and access group) is enforced.
Specify Security Access By		Specifies security access; by User (OPRID) or permission list (OPRCLASS).
Database Type		Designation for RDBMS. The options are Oracle, MS SQL Server, or DB2 Universal Database.
Database Identifier		Database System Identifier (SID) or Source Name used by the appropriate DBMS client software on the server.
Server		Name of the Database Server (for SQL Server only, not Oracle or DB2).
Access ID		ID used by PeopleSoft to connect to the RDBMS.
Access Password		Password associated with the access ID.
Database Owner		Data source owner ID for the PeopleSoft data source.
Authentication		Select *Password* to always verify the user ID/password. Select *Trusted* to accept only user ID. Trusted authentication is useful when deploying in previously authenticated environments, such as a Portal.
Enforce Mixed Case IDs		Provides backwards compatibility for sites that previously used a case-insensitive version of PeopleSoft. (Yes or No)

Option Name	Response	Description
Default Synonym		The type of synonym to default to when creating new metadata. • Record Name • Record Name with Prefix • Record Name with Suffix • Table Name • Table Name with Prefix • Table Name with Suffix

Managing PeopleSoft Metadata

In this section:

The Adapter for PeopleSoft enables management of PeopleSoft metadata. Processes exist that manage the accessibility of PeopleSoft Records by maintaining the metadata catalog on the reporting server. Before reports can be created or run, this metadata catalog must contain information about the PeopleSoft Record Definitions.

You can perform the following tasks to accomplish this:

- Create synonyms for PeopleSoft Records.
- Remove synonyms.
- Update synonyms.

Accessing the Adapter for PeopleSoft Administrator

To access the environment for administration, you must have access to the Web Console and knowledge of the DBA.

Procedure: How to Access the Adapter for PeopleSoft Administrator

1. Open the Web Console.

   You can access the Web Console through a Web browser by opening the URL to the server's HTTP listener port. The default location on a local server is http://localhost:8121/.

   Note: If the Web Console is set to Protected mode, you must supply a Web Console administrator ID and password to gain access.

2. Select *Data Adapters* in the main window.

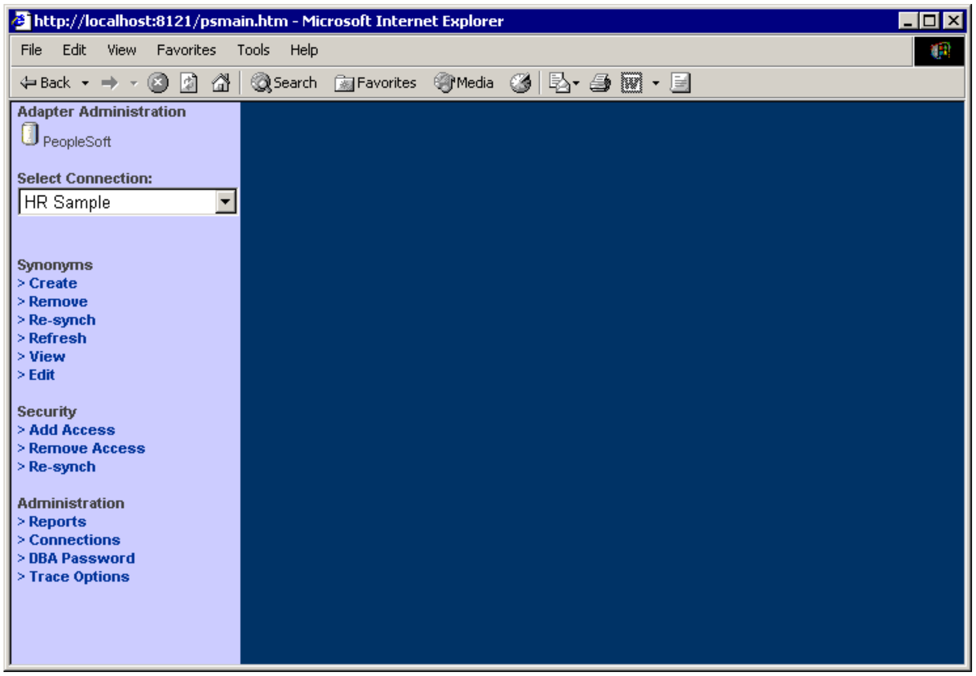
   The Configuring Data Adapters window opens.

3. Select *PeopleSoft* in the Configured folder, and select *Properties* from the pop-up menu.

   The Administration - Enter DBA Password window opens.

4. Type the DBA password, and click *Next*>.

   The PeopleSoft menu opens.



Creating Synonyms

Creating synonyms for PeopleSoft is one of the core functions of the PeopleSoft Administrator. The adapter enables the administrator to choose which PeopleSoft records are defined in the metadata catalog.

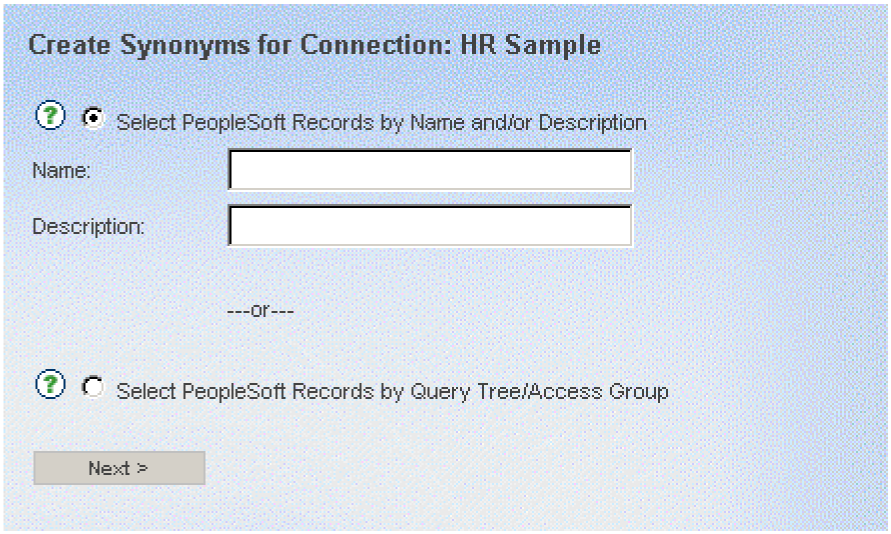
The adapter provides two methods to select PeopleSoft Records to create synonyms:

- **Create synonyms using a filtered search.** A search template option is provided to speed the process of searching for specific PeopleSoft Records. This is a useful method of locating Record Definitions when either the query tree location is not known or specific definitions that meet a filter can be more readily accessed.
- **Create synonyms using a query tree search.** The query tree is a hierarchical logical grouping of the PeopleSoft Record Definitions. This method provides administrators familiar with this organizational structure a way to quickly locate related record definitions. Many sites create a combination query tree and access group definition specifically to group their records for use in reporting.

Procedure: How to Create Synonyms Using a Filtered Search

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 33-14.
2. Click *Create* under the Synonyms group.

   The Create Synonyms for Connection window opens.



Create Synonyms for Connection: HR Sample

[✓ Select PeopleSoft Records by Name and/or Description]

Name:

Description:

---or---

Select PeopleSoft Records by Query Tree/Access Group

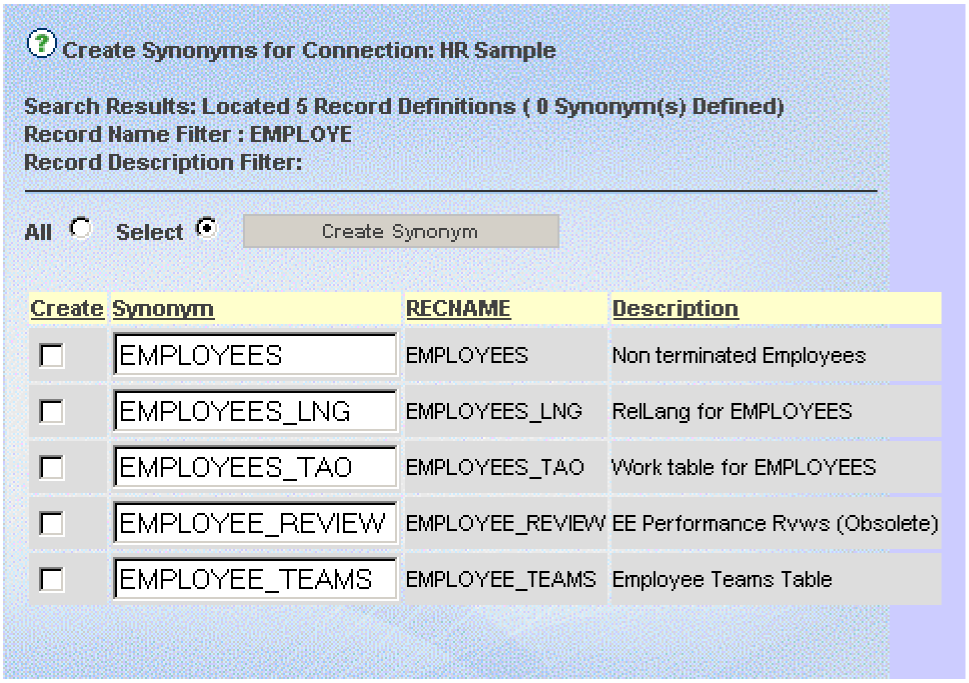
3. Click *Select PeopleSoft Records by Record Name and/or Description* and type a Record name or description to filter by in the available fields, or leave the fields blank to see a full list of Records.

   Note:

   - The Record Description is case sensitive.
   - If you do not use a filter, or your filtered results are very large, the results may be truncated. Use a different filter to reduce the returned results.

4. Click *Next>*.

   The Create Synonyms for Connection window opens.



Create Synonyms for Connection: HR Sample

Search Results: Located 5 Record Definitions (0 Synonym(s) Defined)
Record Name Filter : EMPLOYE
Record Description Filter:

All ○ Select ● Create Synonym

Create	Synonym	RECNAME	Description
☐	EMPLOYEES	EMPLOYEES	Non terminated Employees
☐	EMPLOYEES_LNG	EMPLOYEES_LNG	RelLang for EMPLOYEES
☐	EMPLOYEES_TAO	EMPLOYEES_TAO	Work table for EMPLOYEES
☐	EMPLOYEE_REVIEW	EMPLOYEE_REVIEW	EE Performance Rvws (Obsolete)
☐	EMPLOYEE_TEAMS	EMPLOYEE_TEAMS	Employee Teams Table

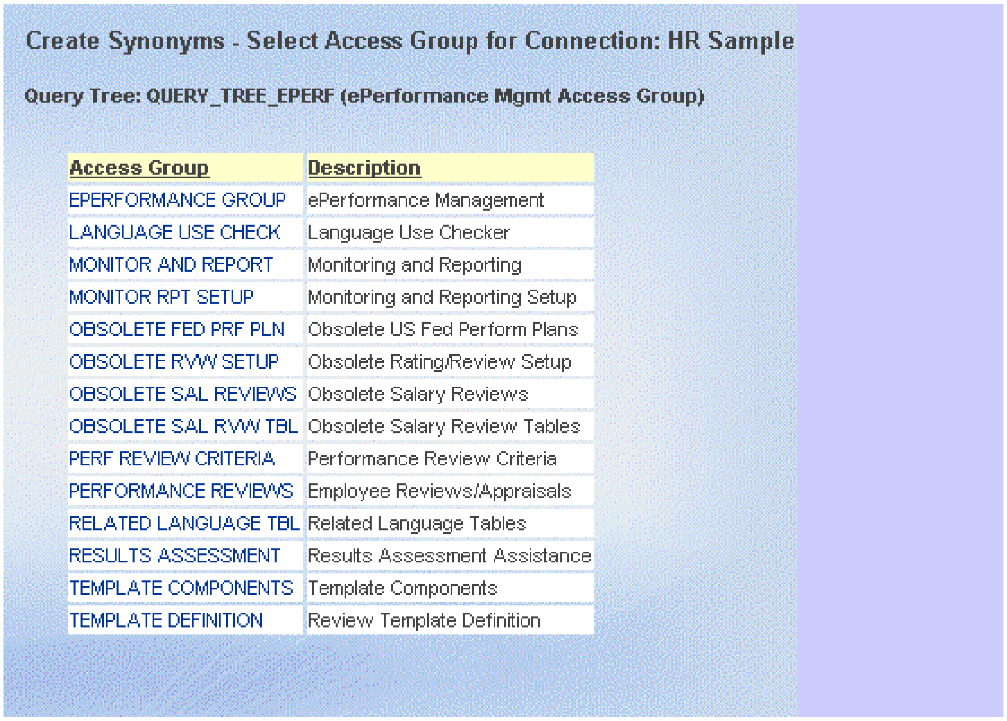
5. Select the check box in the Create column for all the Records you want to create a synonym for, or select *All* to create a synonym for all available Records.
6. Optionally, change the synonym name in the Synonym name column.
7. Click *Create Synonym* to process. Processing may take a few minutes for each selected Record Definition.

Procedure: How to Create Synonyms Using a Query Tree Search

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 33-14.
2. Click *Create* under the Synonyms group. The Create Synonyms for Connection window opens.
3. Click *Select PeopleSoft Records by Query Tree/Access Group* and click *Next>*.

   The Create Synonyms - Select Query Tree for Connection window opens.
4. Select a Query Tree from the table by clicking the hyperlinked name.

   The Create Synonyms - Select Access Group for Connection window reflects your selection:



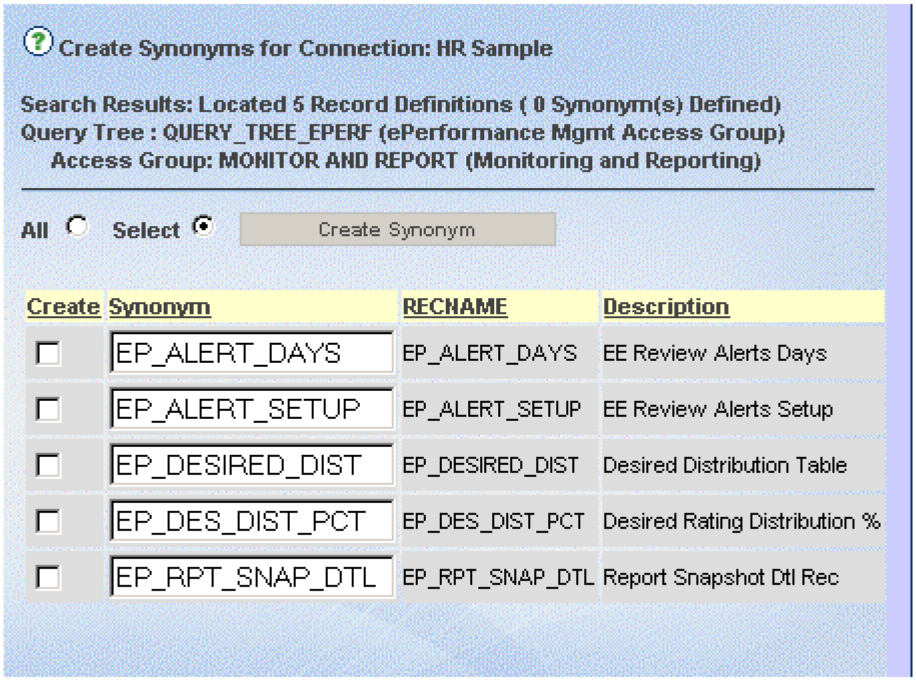
Create Synonyms - Select Access Group for Connection: HR Sample

Query Tree: QUERY_TREE_EPERF (ePerformance Mgmt Access Group)

Access Group	Description
EPERFORMANCE GROUP	ePerformance Management
LANGUAGE USE CHECK	Language Use Checker
MONITOR AND REPORT	Monitoring and Reporting
MONITOR RPT SETUP	Monitoring and Reporting Setup
OBSOLETE FED PRF PLN	Obsolete US Fed Perform Plans
OBSOLETE RVW SETUP	Obsolete Rating/Review Setup
OBSOLETE SAL REVIEWS	Obsolete Salary Reviews
OBSOLETE SAL RVW TBL	Obsolete Salary Review Tables
PERF REVIEW CRITERIA	Performance Review Criteria
PERFORMANCE REVIEWS	Employee Reviews/Appraisals
RELATED LANGUAGE TBL	Related Language Tables
RESULTS ASSESSMENT	Results Assessment Assistance
TEMPLATE COMPONENTS	Template Components
TEMPLATE DEFINITION	Review Template Definition

5. In the Create Synonyms - Select Access Group for Connection window, click a hyperlinked name to drill down on an access group.

   The Create Synonyms for Connection window opens.



Create Synonyms for Connection: HR Sample

Search Results: Located 5 Record Definitions (0 Synonym(s) Defined)
Query Tree : QUERY_TREE_EPERF (ePerformance Mgmt Access Group)
Access Group: MONITOR AND REPORT (Monitoring and Reporting)

All ○ [✓ Select] Create Synonym

Create	Synonym	RECNAME	Description
☐	EP_ALERT_DAYS	EP_ALERT_DAYS	EE Review Alerts Days
☐	EP_ALERT_SETUP	EP_ALERT_SETUP	EE Review Alerts Setup
☐	EP_DESIRED_DIST	EP_DESIRED_DIST	Desired Distribution Table
☐	EP_DES_DIST_PCT	EP_DES_DIST_PCT	Desired Rating Distribution %
☐	EP_RPT_SNAP_DTL	EP_RPT_SNAP_DTL	Report Snapshot Dtl Rec

6. Select the check box in the Create column for all the records you want to create a synonym for, or select *All* to create a synonym for all available records.
7. Optionally, change the synonym name in the Synonym name column.
8. Click *Create Synonym* to process. Processing may take a few minutes for each selected record definition.

Removing Synonyms

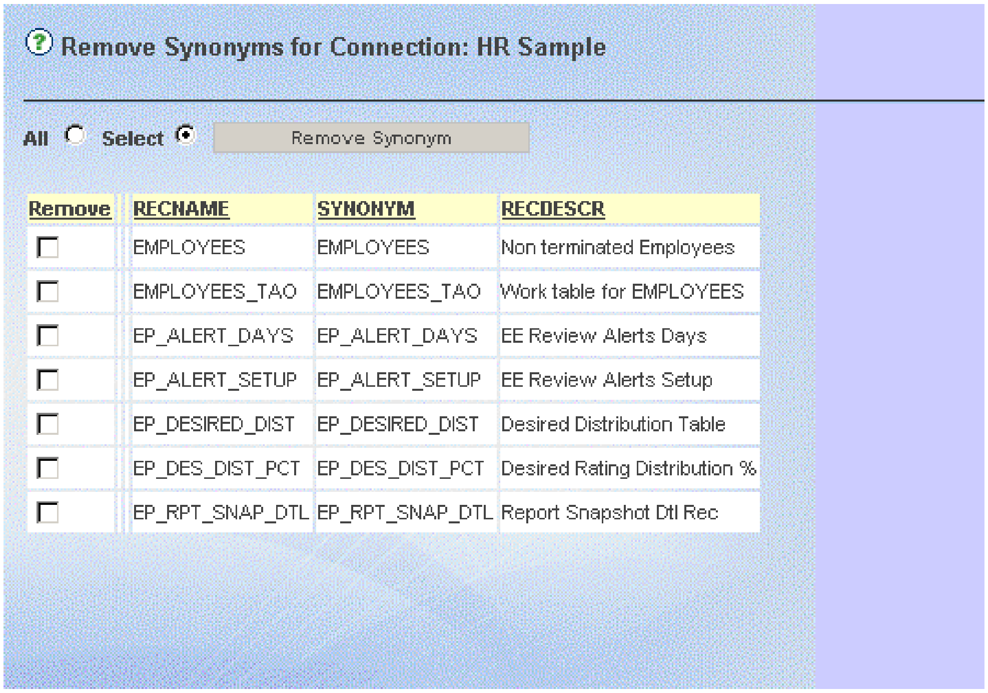
You can remove previously created PeopleSoft synonyms from the metadata catalog with the Remove Synonyms menu option.

Procedure: How to Remove Synonyms

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 33-14.

2. Click *Remove* under the Synonyms group.

   The Remove Synonyms for Connection window opens.



All [✓ Select]

Remove	RECNAME	SYNONYM	RECDESCR
☐	EMPLOYEES	EMPLOYEES	Non terminated Employees
☐	EMPLOYEES_TAO	EMPLOYEES_TAO	Work table for EMPLOYEES
☐	EP_ALERT_DAYS	EP_ALERT_DAYS	EE Review Alerts Days
☐	EP_ALERT_SETUP	EP_ALERT_SETUP	EE Review Alerts Setup
☐	EP_DESIRED_DIST	EP_DESIRED_DIST	Desired Distribution Table
☐	EP_DES_DIST_PCT	EP_DES_DIST_PCT	Desired Rating Distribution %
☐	EP_RPT_SNAP_DTL	EP_RPT_SNAP_DTL	Report Snapshot Dtl Rec

3. Select the check box in the Remove column for each synonym you want to remove, or select *All* to remove all available synonyms.

4. Click *Remove Synonym*.

Updating Synonyms

You can resynchronize your metadata. This option:

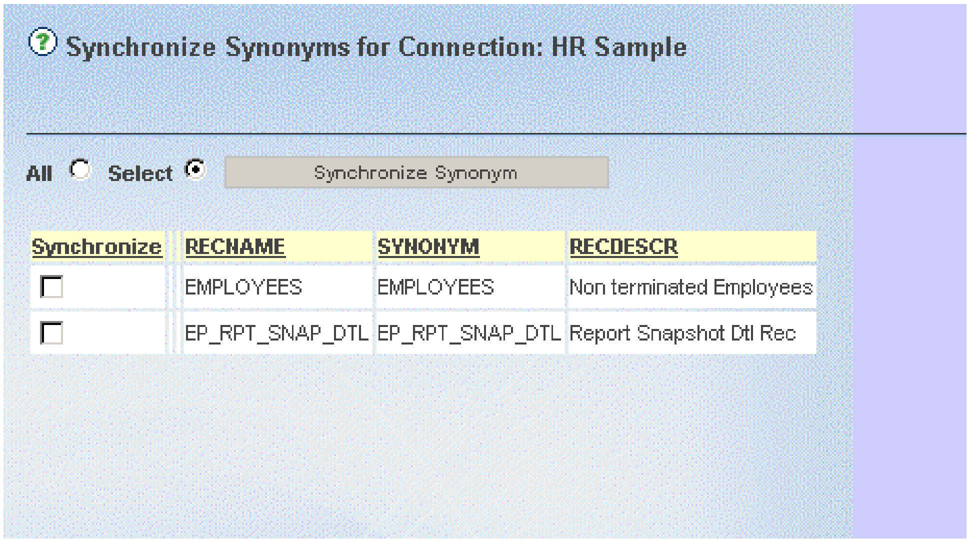
- Queries the internal repository to determine which records were installed for the selected PeopleSoft connection.
- Compares the version number in those records with the version number in the PeopleSoft data source. If none of the records have changed, a message appears.

If there are any records that have changed, they appear in the Select Record(s) to Synchronize page.

Procedure: How to Resynchronize Synonyms

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 33-14.
2. Click *Re-Synch* under the Synonyms group.

   The Synchronize Synonyms for Connection window opens.



Synchronize Synonyms for Connection: HR Sample

All ○ Select ●

Synchronize	RECNAME	SYNONYM	RECDESCR
☐	EMPLOYEES	EMPLOYEES	Non terminated Employees
☐	EP_RPT_SNAP_DTL	EP_RPT_SNAP_DTL	Report Snapshot Dtl Rec

3. Select the check box in the Synchronize column for each synonym you want to resynchronize with PeopleSoft Record Definitions, or select *All* to resynchronize all listed synonyms.
4. Click *Synchronize Synonym*.

Viewing Sample Data

After creating a synonym, you may want to verify that it is using the correct Record Definition. The *Synonyms>View* option on the PeopleSoft menu provides an easy method for listing the existing synonyms and viewing a sample data set from each.

Renaming a Synonym

You can selectively rename a synonym using the *Synonyms>Edit* option on the PeopleSoft menu.

After renaming a synonym, you must update references in existing reports from the old name to the new name using appropriate options in your reporting tool.

Managing PeopleSoft Secured Data Access

An administrator can manage security access according to individual users or a permission list. The decision for how to manage access is made as a connection option, and the two methods are mutually exclusive.

Regardless of the method for granting access rights to users, the process for managing security access is roughly identical. The administrator can affect security access for a user in the following ways:

- Adding security access.
- Removing security access.
- Resynchronizing security access.

Enabling User Access

How to:

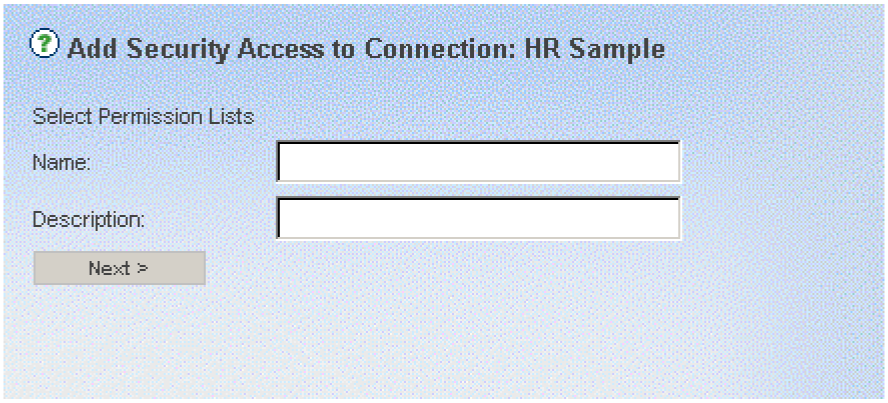
Enable Security Access for a User

Before a PeopleSoft user can run reports against PeopleSoft, security access rights must be granted. The PeopleSoft connection does not automatically enable reporting for every user; instead an administrator must decide which users have reporting access.

Procedure: How to Enable Security Access for a User

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 33-14.
2. Click *Add Access* under the Security group.

   The Add Security Access to Connection window opens.



Add Security Access to Connection: HR Sample

Select Permission Lists

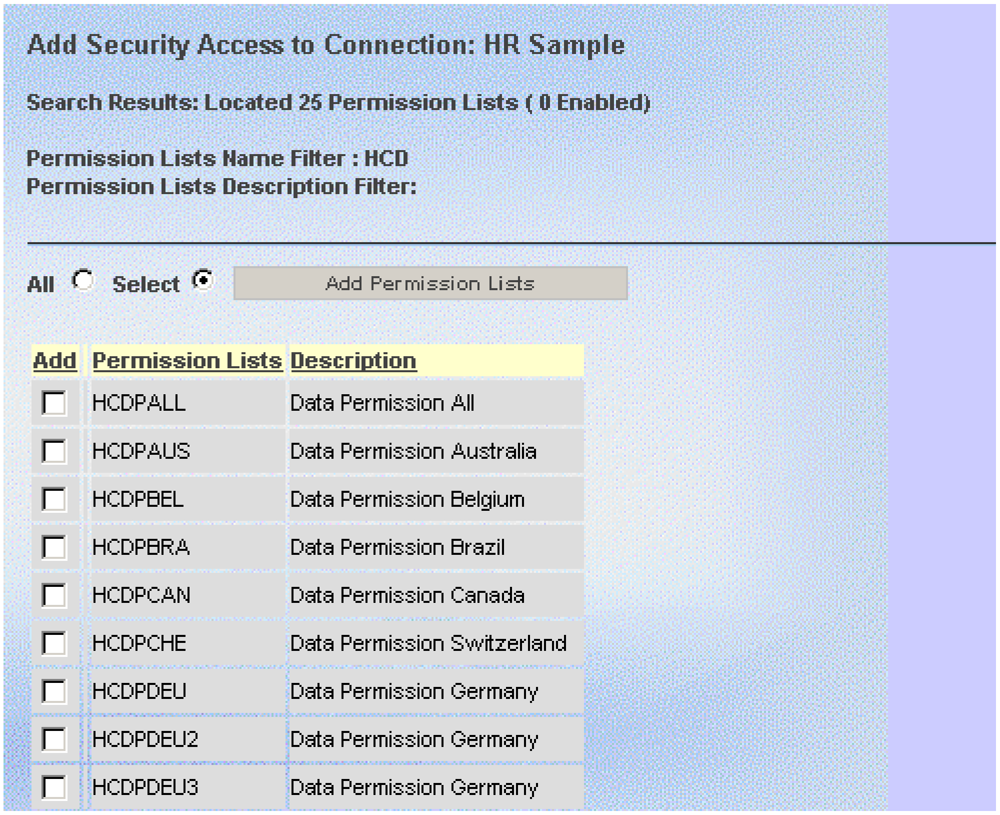
Name:

Description:

3. Select filtering criteria to reduce the list of users, and click *Next>*, or leave the fields blank and click *Next>* to see a full list of users.
   - If your connection is set up to manage security by user ID, then provide user name or description filtering criteria.
   - If your connection is set up to manage security by permission list, then provide relevant permission list criteria.

   Note: If you do not use a filter, or your filtered results are very large, the results may be truncated. Use a different filter to reduce the returned results.

The Add Security Access to Connection window opens.



Add Security Access to Connection: HR Sample

Search Results: Located 25 Permission Lists (0 Enabled)

Permission Lists Name Filter : HCD
Permission Lists Description Filter:

All ○ [✓ Select ●]

Add	Permission Lists	Description
☐	HCDPALL	Data Permission All
☐	HCDPAUS	Data Permission Australia
☐	HCDPBEL	Data Permission Belgium
☐	HCDPBRA	Data Permission Brazil
☐	HCDPCAN	Data Permission Canada
☐	HCDPCHE	Data Permission Switzerland
☐	HCDPDEU	Data Permission Germany
☐	HCDPDEU2	Data Permission Germany
☐	HCDPDEU3	Data Permission Germany

4. Select the check boxes in the Add column for the user IDs or permission lists you would like to add access for, or select *All* to add access for all available users.
5. Click *Add Users* or *Add Permission Lists* to add the selected users.

Note: When you add users by Permission List, the users will only be granted access to Records with which the added Permission Lists are associated.

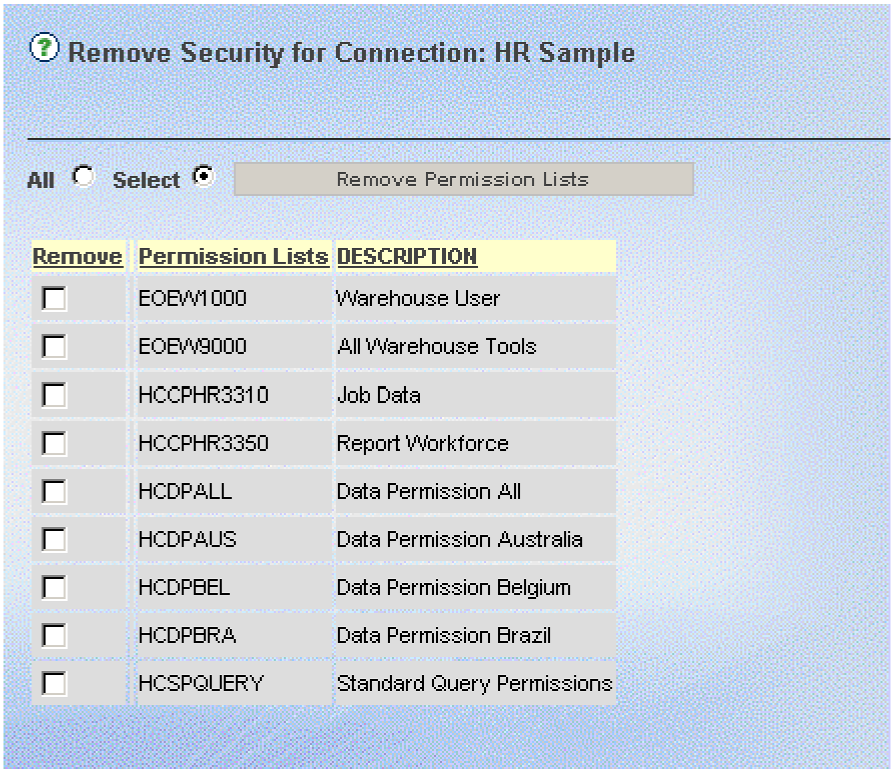
Disabling User Access

You can remove security access for a user from the PeopleSoft menu.

Procedure: How to Remove Security Access for Users

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 33-14.
2. Click *Remove Access* under the Security group.

   The Remove Security for Connection window opens.



Remove Security for Connection: HR Sample

All ○ Select [✓ ●] Remove Permission Lists

Remove	Permission Lists	DESCRIPTION
☐	EOEW1000	Warehouse User
☐	EOEW9000	All Warehouse Tools
☐	HCCPHR3310	Job Data
☐	HCCPHR3350	Report Workforce
☐	HCDPALL	Data Permission All
☐	HCDPAUS	Data Permission Australia
☐	HCDPBEL	Data Permission Belgium
☐	HCDPBRA	Data Permission Brazil
☐	HCSPQUERY	Standard Query Permissions

3. Select the check boxes in the Remove column for the user IDs or permission lists that should be removed, or select *All* to remove access for all available users.
4. Click *Remove Users* or *Remove Permission Lists*.

   The PeopleSoft connection removes all references to these users or permission lists. After they are removed, the associated users cannot access PeopleSoft data through PeopleSoft.

Updating User Access

How to:

Resynchronize Security Rules

You can resynchronize PeopleSoft security metadata when changes occur in the PeopleSoft environment. The following situations require resynchronization:

- A permission list has had query access group privileges modified.
- A Record has been added or removed from a query tree.
- A user has been added to or removed from a permission list with security access enabled. This can occur directly, as in PeopleSoft 7.x, or indirectly through role assignment changes.
- Row-level security class or permission lists are changed.

You do not need to run the security resynchronization routine if row-level security access has changed, based on the security table being updated. This type of security change will be enforced automatically and immediately. If the Query Security Record within a Record Definition has been changed, the synonym must be updated.

Note: A batch mechanism for resynchronization is also available that may be scheduled through a command line execution. For more information, see *Advanced Administrative Topics* on page 33-31.

Procedure: How to Resynchronize Security Rules

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 33-14.
2. Click *Re-Synch* under the Security group. Security rules are resynchronized based upon security access settings.

   Processing may take several minutes. Wait for a completion message before attempting other administrative tasks.

Managing Connections to PeopleSoft

In this section:

Multiple connections to PeopleSoft data sources are possible from within one server configuration. However, depending on your reporting requirements, it may be optimal to create separate configurations for each data source.

Creating multiple connections using the same server configuration is a good option under the following circumstances:

- The site has two or more PeopleSoft application suites and must perform significant combined reporting. A typical example is a site that has both PeopleSoft HRMS and PeopleSoft Financials applications where you must display data from both systems in a single report.
- Two or more PeopleSoft data sources that must be accessed reside on the same system. If the data sources must be accessed for display in a single report output and if the server is located on the same computer system, this architecture can provide optimal performance.
- Two or more PeopleSoft data sources reside in different locations with requirements for moderate to significant consolidated reporting. This architecture provides optimal effectiveness in report development, system maintenance, and performance.

Create separate configurations instead of using multiple connections in these situations:

- Only a single PeopleSoft production data source must be supported. Each development and test instance should have a separate configuration.
- Multiple PeopleSoft data sources exist with only a limited requirement for combining data in reports.

The remaining instructions in this topic assume a single configuration directory structure for one or more data source connections.

Updating a PeopleSoft Connection

How to:

Edit a PeopleSoft Connection

Reference:

Updating Connections

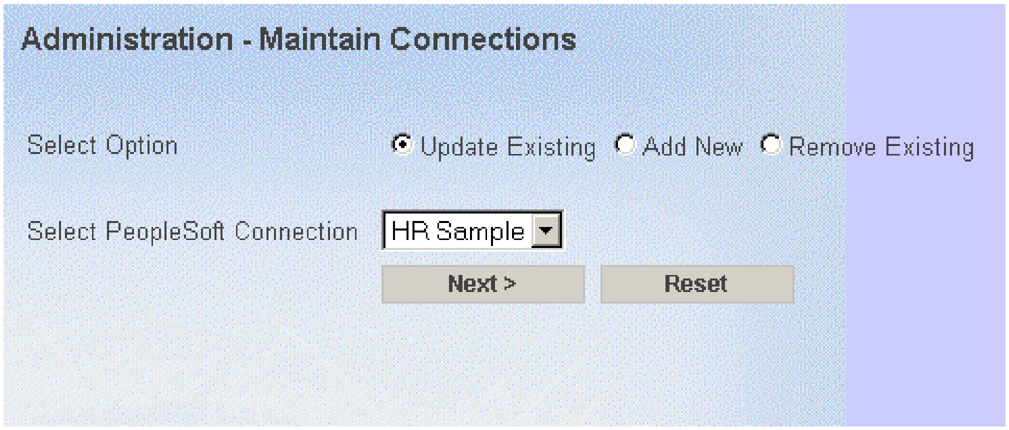
If it becomes necessary to update the connection parameters to a PeopleSoft data source, you can use Administration options to perform the following functions:

- Turn PeopleSoft data access security enforcement on or off.
- Toggle between trusted and password authentication.
- Manage the user access specification type.
- Manage all data source identification information.
- Define default synonym behavior.

Procedure: How to Edit a PeopleSoft Connection

1. From the Web Console, click *Connections* under the Administration group.

   The Administration - Maintain Connections window opens.



Administration - Maintain Connections

Select Option: [✓ Update Existing] Add New Remove Existing

Select PeopleSoft Connection: HR Sample

2. Select *Update Existing*.
3. Select a connection from the Select PeopleSoft Connection drop-down list.

4. Click *Next>*.

   The Administration - Update a PeopleSoft Connection window opens. For details, see *Administration - Create a New PeopleSoft Connection Window* on page 33-9.

5. Make the modifications you wish, and click *Next>*.

Reference: Updating Connections

When you change the *Specify Security Access By* setting, you must resynchronize security before changes will take effect. We suggest that you remove all currently configured users before you make this change, and re-add them after security settings have been modified.

As when you modify authentication information, you must be careful to manage changes to data source connectivity correctly. Always modify your settings in the PeopleSoft Connection first, then change them at the data source. Where possible, keep both access options in place during a configuration conversion process until tests have verified a successful switch.

Adding a PeopleSoft Connection

After you add your first connection, you can specify additional PeopleSoft connections. Note that during this process, you are not prompted for the DBA information prior to managing metadata and connection information since all additional PeopleSoft data sources must use the same DBA in order for base metadata to function properly.

Procedure: How to Add a PeopleSoft Data Source Instance

1. From the Web Console, click *Connections* under the Administration group. The Administration - Maintain Connections window opens.

2. Select *Add New*.

3. Click *Next>*.

   The Administration - Create a New PeopleSoft Connection window opens. For details, see *Administration - Create a New PeopleSoft Connection Window* on page 33-9.

When processing finishes, you can manage your synonyms and security for this connection.

Removing a PeopleSoft Connection

The integration with PeopleSoft depends on valid connection information. If a previously created connection is no longer required, you should remove that connection.

Procedure: How to Remove a PeopleSoft Connection

1. From the Web Console, click *Connections* under the Administration group. The Administration - Maintain Connections window opens.
2. Select a connection from the Select PeopleSoft Connection drop-down list.
3. Select *Remove Existing*.
4. Click *Next>*.

   The Administration - Remove a PeopleSoft Connection window opens.

   You are prompted to confirm. After confirmation, the system removes the selected PeopleSoft Connection and all of its associated metadata.

Using Administrative Utilities

In this section:

The Adapter for PeopleSoft provides administrative tools that enable you to report on connection information, update the DBA Password, and set adapter tracing options.

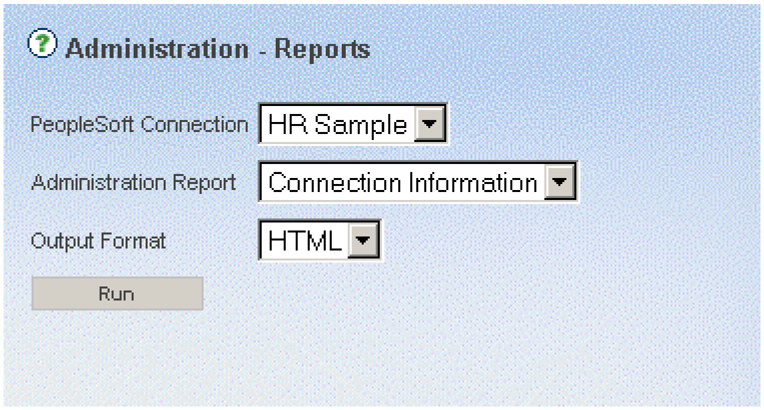
Connection Reports

The Administrator provides reports to make administration easier. You can run reports that provide information on connections, metadata, users, and user access. These reports are accessible from the index.

Procedure: How to Run Connection Reports

1. In the PeopleSoft menu, select the connection you want to manage from the Select Connection drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 33-14.
2. Click *Reports* under the Administration group.

   The Administration - Reports window opens.



Administration - Reports

PeopleSoft Connection: HR Sample

Administration Report: Connection Information

Output Format: HTML

3. If necessary, select the PeopleSoft connection you want to run a report for from the Select PeopleSoft Connection drop-down list.

4. Select a report type from the Administration Report drop-down list. The options are:
   - **Connection Information.** Retrieves statistics for your connection.
   - **Synonyms.** Retrieves a list of available synonyms.
   - **Users/Oprids.** Retrieves a list of users who have access to the current connection.
   - **Access Assigned.** Retrieves a list of the access assigned to each user.
5. Select an output format for the report from the Output Format drop-down list. The options are HTML and PDF.
6. Click *Run*.

The selected report appears.

Updating the DBA Password

The DBA Password option is used for updating the password that is supplied during the initial configuration of the adapter. When the password is updated, all synonyms are updated as well. (Note that this process may take several minutes.)

Tracing Adapter Processing

The adapter tracing utility creates application level traces for support purposes. Typically, tracing is used under the direction of Information Builders Customer Support Services.

Advanced Administrative Topics

In this section:

In addition to the standard administrative options described in this chapter, advanced options and techniques are available to assist in automating security access updates and troubleshooting integration problems. This topic discusses these options and provides additional details on PeopleSoft security integration.

Automating Security Access Updates

Security resynchronization is one of the functions of the PeopleSoft Administrator. When administering PeopleSoft connections through the Web console, you simply select the Re-Synch link under the Security group, and all of the PeopleSoft data security rules are resynchronized.

The batch resynchronization utility enables administrators to schedule, through an operating system command, the execution of this functionality without having to manually open the Administrator tool. By scheduling this process nightly (or otherwise; based on business requirements), administrators are not required to perform this function manually, and they can be assured that the data security rules are current as of the last scheduled run.

The utility is comprised of two files. These files are a procedure (pssecsnk.fex) used in executing the resynchronization routine and a t3i script (pssecsnk.t3i), which is the command line input file. By starting the server with the t3i script as an input, the server performs the required security resynchronization without user intervention.

A server can be accessed locally without the use of client software. This functionality can be started in the following two modes:

- **Interactive Agent** is useful for:
  - Debugging application errors.
  - Troubleshooting connectivity and configuration.
  - Testing performance.
- **Batch Mode** is useful for scheduling batch scripts with operating system tools.

An administrative user with access to the server can use the server in stand-alone mode to perform any of these functions. If PeopleSoft data must be accessed in any of these processes, you must execute the PeopleSoft authentication routine. Because of the processing within the authentication routine, the creation of a custom startup script is recommended.

The two purposes for the custom script are:

- To enable successful execution of the PSLOGIN remote procedure call.
- To ensure that when executing, a separate and temporary workspace is used. This prevents errors due to concurrent processing, including data access errors.

When using the edastart script, temporary files are always written to the current directory. When using edastart without modification, be sure to delete these temporary files from the current directory after execution to avoid unexpected results during run time.

Example: Creating a Custom Server Startup Script for UNIX

The following provides the steps for creating and using a custom startup script in UNIX. A similar script can be created in Windows with appropriate operating system-specific syntax.

Navigate to the EDACONF/bin directory on your server and create a new script with a text editor. The following script is named pslocal. The numbers on the left correspond to the notes explaining the code.

Note: This example creates a unique directory under the $EDACONF/edalocal directory. The edalocal directory must exist in most operating systems before a directory can be created within it.

```
    #!/bin/ksh
    #
    parms=$*
    #
    #  Define the Conf Directory
1.  export EDACONF=/ibi/srv52/wfs
    #
    #  Create unique temporary directory for running locally
2.  EDA_PID=$$
    export EDATEMP=$EDACONF/edalocal/$EDA_PID
    rm -rf $EDATEMP
    mkdir $EDATEMP
3.  cd $EDATEMP

4.  $EDACONF/bin/edastart $parms

5.  cd $EDACONF/bin
    rm -r $EDATEMP
```

The script processes as follows:

1. The export command defines the EDACONF variable to use. This makes the remainder of the script more flexible and portable.
2. These four lines define a unique directory based on a PID and UNIX process ID so that all temporary files will be written to a unique location.
3. The CD command changes the current directory to the new temporary directory.
4. This line calls the delivered edastart script and parameters passed through the script.
5. These two lines change the current directory from the temporary space, and deletes the temporary files.

Procedure: How to Use the PeopleSoft Interactive Agent

1. Navigate to the server configuration directory, then to the bin subdirectory.
2. Execute the Interactive Agent start-up script with the -t option. This is the same for UNIX and Windows platforms.

   ```
   /ibi/srv53/wfs/bin/pslocal -t
   ```

   where:

   `/ibi/srv53/wfs/bin`

   Is the bin directory under the server configuration.

   When you run

   ```
   pslocal -t
   ```

   you see information similar to:

   ```
   RELEASE = R729530B
   GEN_NUM = 001.123456
   SOURCE_DATE = 01/01/2004 19:36:53
   BUILD_DATE = 01/01/2004 02:01:09
   INSTALLATION_DATE= 01/01/2004 02:46:42
   >>
   ```

   At the prompt, you can type commands.
3. To perform any commands that access PeopleSoft data, first you must run the interactive authentication routine. If this procedure is not run, you receive data access errors. To execute the procedure, type

   ```
   >>EX PSLOGIN USERID=PS_Userid, PASSWD=PS_Password
   ```

   where:

   `PS_Userid and PS_Password`

   Is a valid PeopleSoft user ID and password.

You see processing information followed by a FOCUS command prompt. After control returns to the FOCUS command prompt, PeopleSoft data may be accessed with the FOCUS language.

Note: If your configuration has been set up with Trusted authentication, then you are not required to supply the PeopleSoft password. Only a valid user ID is required.

Procedure: How to Run Batch Mode Security Resynchronization From the Command Line

1. Locate the pssecsnk.t3i script in the EDACONF\catalog directory.
2. Using PeopleSoft Batch-Mode, execute with the security synchronization input script. To execute input scripts with the *pslocal* script, use the -f option

   ```
   c:\ibi\srv53\wfs\bin\pslocal -f c:\ibi\srv53\wfs\catalog\pssecsnk.t3i
   ```

   where:

   `c:\ibi\srv53\wfs\catalog`

   Is the directory location of the t3i file.

For details about securing the script, see *Password Security* on page 33-35.

Reference: Password Security

The t3i script is an ASCII text file that must be edited for your particular environment. It contains two FOCUS execute command lines that:

- Log into PeopleSoft.
- Perform the security resynchronization.

The syntax is:

```
< Filename: pssecsnk.t3i >
%CONNECT
%BEGIN
EX PSLOGIN USERID=PS, PASSWD=PS
EX PSSECBAT
%END
%DISCONNECT
%STOP_SERVER
```

You must modify the user ID and password in accordance with one that is appropriate for your environment. The user ID must be granted access in PeopleSoft.

Since the file is in readable text, the administrator must prevent read/write access for anyone not authorized to access the t3i. The best way to prevent unauthorized access is to use operating system security. Windows and UNIX provide ways to prevent unauthorized access to individual files and entire directories. Contact your operating system administrator for assistance in performing this security step.

Procedure: How to Capture Execution Results

Usually, after execution, all temporary work files are deleted immediately. If you must verify work files or see the batch process output file, you can temporarily modify the execution script.

To temporarily prevent the deletion of these processing files, edit the pslocal script found in the EDACONF/bin directory. Insert the appropriate comment character for your environment where the final file deletion step exists.

In UNIX environments, type:

```
# rm -r $EDATEMP
```

After the script has been executed, the Administrator can review the execution results. A batch process output file (pssecsnk.t3o) is automatically created in the EDACONF/edalocal directory, with a unique subdirectory for each execution.

For an illustration, see *Creating a Custom Server Startup Script for UNIX* on page 33-33.

Example: Capturing Execution Results

The following is an example of a successful execution:

```
< Filename: pssecsnk.t3o >
%connect
%begin
ex pslogin userid=ps, passwd=ps
ex pssecbat
%end
SYSPRINT.SCR
0 NUMBER OF RECORDS IN TABLE=       1  LINES=      1
File doesn't exist
 DBNUM: 1 has been re-synchronized, DBA security is: ON
CLOSEALL.ALL
%disconnect
%stop_server
```

Note: As the user ID and password appear in this output file, the file must be protected by operating system security.

Troubleshooting Tips

How to:

Connect to the PeopleSoft Data Sources

Verify Security

These troubleshooting tips help if you are getting FOCUS error messages, such as a FOC1302, or if you are seeing SQL errors when you run reports.

Procedure: How to Connect to the PeopleSoft Data Sources

The most likely source of difficulty in connecting to your PeopleSoft data source is the data source connectivity. If you are experiencing difficulty connecting to the data source:

1. Verify that the data source instance is up and running.
2. Run a client query tool on the server, using the PeopleSoft access ID.
   - If you cannot log in to the data source, there is a problem with the access ID.

     Verify the ID and check to see if the password has changed.

     If the password has changed, then it must also be changed in the PeopleSoft configuration.
   - If you can log in, proceed to Step 3.
3. Run a few simple queries against the PeopleSoft records that are returning errors.

   If you are not getting any rows back, check to see if the PeopleSoft access ID has read authority on the PeopleSoft records you are accessing.

Procedure: How to Verify Security

PeopleSoft enforces all the security mechanisms used by the PeopleSoft Query tool. As of PeopleSoft 7.5, this includes Query Tree security and row-level security. If you are getting unexpected results, it is possible that your PeopleSoft security has not been configured properly.

To verify that security is being properly enforced:

1. Verify that the synonym(s) being queried are correct.

   The Administrator can run a report of installed records to verify. If row-level security is the issue, the server-side ACX file should be reviewed and compared to the record definition in PeopleSoft, including the Security Search record.
2. Create a similar query in PS/Query. The SQL code can be viewed and compared to SQL tracing on the server.

3. Examine the SQL.

   Does it appear to match the business logic behind the original request? If not, check the syntax behind the original FOCUS code. Perhaps you are using a WRITE instead of a PRINT, or perhaps a WHERE clause has been coded incorrectly.

4. Run this SQL directly against the RDBMS using a data source query tool.

   If you are getting the expected results, then there may be a problem with the program logic after the data is returned to FOCUS from the RDBMS. Examine COMPUTE/DEFINE statements for logic errors; or if you are using ON TABLE HOLD, there may be a problem in your code.

   If you are getting the same unexpected results, then examine the underlying data. Perhaps some cross-referenced fields are missing, or perhaps row-level security is eliminating essential rows.

CHAPTER 34

Using the Adapter for PeopleSoft EnterpriseOne

Topics:

The Adapter for PeopleSoft EnterpriseOne allows WebFOCUS and other applications to access PeopleSoft EnterpriseOne data sources. With this adapter, data in the PeopleSoft EnterpriseOne DBMS is displayed using rules contained in dictionary files, thereby ensuring that valid information is returned to the requesting program.

Note: PeopleSoft EnterpriseOne was previously the OneWorld product from J. D. Edwards. The iWay Web Console currently reflects the earlier, J. D. Edwards OneWorld name. This will change in future releases of the iWay software.

This adapter is available for the Windows and OS/400 environments. It is intended for use with WebFOCUS Version 5 Release 3.2 and higher.

This chapter describes the process you must follow to:

- Configure the data adapter required to create synonyms for reporting from a PeopleSoft EnterpriseOne database.
- Create the synonyms for selected tables.
- Configure the Adapter for PeopleSoft EnterpriseOne.
- Convert synonym elements to make them more suitable for reporting from a PeopleSoft EnterpriseOne database.

Preparing the PeopleSoft EnterpriseOne Environment

Although no environment preparation steps are required, ensure that your system complies with all software specifications.

You must also disable Automatic Passthru to ensure proper processing of PeopleSoft EnterpriseOne data.

Reference: Software Requirements

- Any PeopleSoft EnterpriseOne (previously J. D. Edwards One World) Application installation.
- DBMS connectivity, as required by the installed Enterprise One environment. The following DBMSs are currently supported:
    - Oracle
    - Microsoft SQL Server
    - IBM DB/2
    - DB2/400

Note: This adapter can only be used on the Windows and OS/400 operating systems.

Syntax: How to Disable Automatic Passthru

Use the following command to turn Automatic Passthru off:

```
SET APT=OFF
```

Creating Synonyms for PeopleSoft EnterpriseOne

Before you can use the Adapter for PeopleSoft EnterpriseOne, you must create synonyms for the data you will be reporting against using one of the data adapters that is supported on your platform, and set the paths required to access those synonyms:

- In the Windows environment, you can create synonyms using the Adapters for DB2, Oracle, or Microsoft SQL Server.
- In the OS/400 environment, you can create synonyms using the Adapter for DB2.

Procedure: How to Create Synonyms for PeopleSoft EnterpriseOne From the Web Console

1. From the Web Console, configure the data adapter you wish to use to create synonyms for the PeopleSoft EnterpriseOne database. Your configuration creates an entry in the edasprof.prf file for connecting to the PeopleSoft EnterpriseOne database.
2. From the Web Console, use the configured data adapter to create synonyms for the PeopleSoft EnterpriseOne tables you wish to use. During this process, you will select the tables for which you wish to create synonyms and specify the application in which the tables will be stored.

For details about configuration and synonym-creation steps for supported data adapters, see Chapter 8, *Using the Adapter for DB2*, Chapter 23, *Using the Adapter for Microsoft SQL Server*, or Chapter 31, *Using the Adapter for Oracle*.

Configuring the Adapter for PeopleSoft EnterpriseOne

How to:

Configure the Adapter for PeopleSoft EnterpriseOne

Example:

Editing Variables in JDEGDINC.FEX

Once the data adapter has been configured and synonyms have been created, you must configure the Adapter for PeopleSoft EnterpriseOne. This process defines connection and authentication information, establishes the application structure, and creates the required adapter security extract files.

Procedure: How to Configure the Adapter for PeopleSoft EnterpriseOne

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *ERP* group folder, then expand the PeopleSoft EnterpriseOne folder and click a connection. The adapter's Configuration page opens.
3. Specify the Security Connection Type. The options are:
   - **a.** LOGIN_ID. Select this option if every PeopleSoft EnterpriseOne user has a userid on the reporting server system. The reporting server must be secured.
   - **b.** CONN_ID. Select this option if every PeopleSoft EnterpriseOne user's userid/password is stored and is being authenticated by an MRE and/or a self-service application. The IBI_REPORT_USER variable must be set to the PeopleSoft EnterpriseOne User Id.

     For MRE, the userid must be the PeopleSoft EnterpriseOne User Id.

     For Self Service requests, the MR id must be passed to the server. The reporting server must be unsecured.
4. Select a profile from the drop-down list to indicate the level of profile that the server is running: global, service, group, user. The standard iWay global profile, edasprof.prf, is the default.
5. Click *Configure*.

   This creates the entry `jdeow_access=y` in the edaserve.cfg file and adds the following three lines of code in the selected profile:

   ```
   APP PREPENDPATH JOWSEC
   SET USER=JDEOW
   ENGINE JDEOW SET JDEOW SEC_CONN security_connection
   ```

where:

`security_connection`

Is LOGIN_ID or CONN_ID selected in step 3.

Tip: You can open the edaserve.cfg and edasprof.prf files from the navigation pane of the Web Console home page to check these entries. Notice that edaserve.cfg also displays the an access entry for the data adapter you previously configured to create your synonyms (for example, *db2_access=y*).

6. From the Configured Adapters list in the navigation pane, click on PeopleSoft EnterpriseOne, and select *Create JOWSEC app structure* from the menu. This option adds an application called JOWSEC to your application structure. JOWSEC contains 4 subdirectories: bin, ibilist, master, and reference. It also contains a procedure called jdegdinc.fex, which you will be editing in step 8.

   Tip: You need to create the JOWSEC application only once.

7. Click the *Procedures* folder in the navigation pane of the Web Console home page and verify that JOWSEC had been added to the app path.

8. To enable conversion and security extraction, expand the JOWSEC application folder, click *jdegdinc.fex*, and choose *Edit* from the menu. The file opens in a text editor. Scroll to the end of the file to locate a series of -SET commands. For an illustration, see *Editing Variables in JDEGDINC.FEX* on page 34-6.

9. You must set the three variables (`&DBQtable_name`, `&DBTtable_type`, `&DBAconnection`) for each of the following files

   - For the Convert process, the files are F0004, F0005, F9202, F9210.
   - For the Security Extract process the files are F0092 and F00950.

   where:

   `&DBQtable_name`

   The fully qualified name of table:

   For SQLORA this is owner.table.

   For SQLMSS this is database.owner.table.

   For DB2/400, this is libname/table.

   `&DBTtable_type`

   The DBMS table type— that is, SQLORA, SQLMSS, DB2 (on Windows) or DB2 SQL (on OS/400).

   `&DBAconnection`

   The connection name that you used when you created your connection on the Data Adapters page of the Web Console.

The variables you define in the jdegdinc.fex file are used when the CREATE SYNONYM command is run.

10. From the Configured Adapters list in the navigation pane, click on PeopleSoft EnterpriseOne, and select *Refresh Security Extracts*. This option creates security extract files that are stored as FOCUS files in the JOWSEC application. These files provide extracts of data dictionary files so that the information is available locally.

    Note: You must repeat this step whenever security extract information needs to be updated.

Example: Editing Variables in JDEGDINC.FEX

The example below uses DB2/400 and assumes that F0004/F0005 are in library PRODCTL and F9202/F9210 are in library JD7333. No connection attribute is required in this example.

```
-SET &DBQF0004='PRODCTL/F0004';
-SET &DBTF0004='DB2';
-SET &DBAF0004='';
-*
-SET &DBQF0005='PRODCTL/F0005';
-SET &DBTF0005='DB2';
-SET &DBAF0005='';
-*
-SET &DBQF9202='DD7333/F9202';
-SET &DBTF9202='DB2';
-SET &DBAF9202='';

-*
-SET &DBQF9210='DD7333/F9210';
-SET &DBTF9210='DB2';
-SET &DBAF9210='';
-*
-SET &DBQF0092='SYS7333/F0092';
-SET &DBTF0092='DB2';
-SET &DBAF0092='';
-*
-SET &DBQF00950='SYS7333/F00950';
-SET &DBTF00950='DB2';
-SET &DBAF00950='';
-*
```

Converting Synonyms for PeopleSoft EnterpriseOne

Once the Adapter for PeopleSoft EnterpriseOne has been configured, you must build the dictionary repository that is needed for the conversion process. A utility locates the synonyms to be converted and performs the conversion.

Procedure: How to Convert Synonyms for PeopleSoft EnterpriseOne

1. From the Configured Adapters list in the navigation pane, click on PeopleSoft EnterpriseOne, and select *Convert MFD files*.
2. Click *Refresh Repository* to refresh the metadata repository information that is needed for the conversion process.

   This step create files in the reference subdirectory under the JOWSEC application.

   Note: This step must be repeated each time the dictionary information for the PeopleSoft EnterpriseOne tables change.
3. Click *Select Application* to select the application from which synonyms need to be converted.
4. Enter the application name or choose one from the Application drop-down list.
5. Click *Verify*. A statement verifying your selection is displayed.
6. Click *Convert Metadata*.
7. Enter **ALL* or the PeopleSoft EnterpriseOne file name in the input box, or select the file from the drop-down list.
8. You may also select the a format for date fields from the drop-down selection list.
9. Click *Convert Metadata* to add the following to the synonym:
   - An entry to retrieve UDC (user-defined code) descriptors.
   - An entry to convert Julian dates to Gregorian dates.
   - An entry to correctly display decimal precision.
   - Descriptive field names based on data dictionary entries to replace cryptic internal field names. (The latter are retained as aliases.)

You are now ready to report against these tables.

CHAPTER 35

Using the Adapter for PeopleSoft World

Topics:

The Adapter for PeopleSoft World allows WebFOCUS and other applications to access PeopleSoft World data sources. With this adapter, data in the PeopleSoft World DBMS is displayed using rules contained in dictionary files, thereby ensuring that valid information is returned to the requesting program.

Note: PeopleSoft World was previously the World product from J. D. Edwards. The iWay Web Console currently reflects the earlier, J. D. Edwards World name. This will change in future releases of the iWay software.

Overview of the Setup Process

To enable access to PeopleSoft World, perform the following steps:

1. Unload the I52TOOLS library from the CD-ROM.
2. Run either AUTOSQL or AUTO400 to generate metadata for the PeopleSoft World data.

   Note: You will use the Web Console to generate the metadata for PeopleSoft World data because the metadata generated by the Web Console differs in the use of long field names and extra attributes such as title and description. You must use AUTOSQL or AUTO400 to generate your metadata.
3. Run JDECONV to update the metadata with information derived from the PeopleSoft World Data Dictionary.
4. Enable PeopleSoft World security on the server.

Installation Prerequisites

The software requirements for the Adapter for PeopleSoft World are:

- OS/400 Version V4R4 or higher on a RISC system.
- iWay Server Release 5.2 or higher, should be installed and working. See the *iWay Server Installation* manual.

  Note: If the standard IADMIN ID was not used to install the iWay Server, manually change ownership to the ID that was used.

You must also verify that your end-user program (such as WebFOCUS Developer Studio) can access the iWay server.

Unloading the CD-ROM

Note: The I52TOOLS library on the CD-ROM is a limited packaging of specific tools from what was the EDA43TOOLS library from the 4.x Release. The I51TOOLS and the I52TOOLS CD-ROM differ only in the OS/400 library name and the version of the documentation included on the CD-ROM. If you already have an EDA43TOOLS or I51TOOLS library installed, you may continue to use them and these recommendations. You do not need to install the new library.

To unload the CD-ROM, perform the following steps:

1. If *SECOFR Authority is required to access the CD-ROM drive, logon as QSECOFR or with a user ID that has this authority. Otherwise, logon using the IADMIN ID.
2. Load the CD-ROM into the CD-ROM drive on your system.
3. When the drive is ready, use the RSTLIB command to restore the I52TOOLS library. From the command line, the syntax is

   ```
   RSTLIB SAVLIB(I52TOOLS) DEV(device-name)
   ```

   where:

   device-name

   Is the name of the CD-ROM device. For a list of valid drives, use the WRKCFGSTS *DEV command.

When the unload is complete, you are returned to the command line and the library is ready to use.

End users requiring the I52TOOLS can add the library to the path with:

```
ADDLIBLE I52TOOLS
```

Working With QSYS and IFS File Systems

In this section:

QSYS is the original library-based file system of the OS/400 operating system. IFS is a UNIX-like file system consisting of a directory hierarchy and files within the hierarchy; these are also known as *directories* when working with native OS/400 commands. Along with IFS is a UNIX command shell environment that is activated with the QSH command.

To use QSYS members with iWay 5.x servers, you can copy or link files from QSYS to IFS, or you can set EDAPATH to *see* QSYS libraries.

The 4.x release of the iWay server was an OS/400 product using strictly QSYS libraries for the storage of the actual product files and end-user applications. The 5.x release of the server uses a QSYS library (EDAHOMELIB) for the actual product programs and IFS directories and files for other product files and end-user applications. The EDAPATH and APP MAP methods enable you to access existing applications in QSYS libraries without physically moving them from one file system to another.

Since the iWay tools were originally designed to work with a QSYS-based system, they strictly use QSYS libraries as read and write locations. The strict use of QSYS for these tools may necessitate manual steps to copy files between QSYS and IFS depending upon your application needs. The command syntax for this type of copy is documented in this guide and can be scripted if many files are involved.

Copying Files

Files may be copied to and from QSYS and IFS, but one must be careful which copy is being accessed at any given time in case edits were done to one copy but not the other. To avoid confusion, be sure to clean up unused files.

The following example shows how to copy a Master File from one location to the other:

```
CPYTOSTMF FROMMBR('/qsys.lib/myfoo.lib/master.file/foo.mbr')
          TOSTMF('/home/iadmin/ibi/srv52/ffs/catalog/foo.mas')
          STMFOPT(*REPLACE)

CPYFRMSTMF FROMSTMF('/home/iadmin/ibi/srv52/ffs/catalog/foo.mas ')
           TOMBR('/qsys.lib/myfoo.lib/master.file/foo.mbr')
           MBROPT(*REPLACE)
```

Note how each command works only in a single directory and uses a UNIX-like construct, even for the QSYS name. Use the *REPLACE option to overwrite an existing file.

To copy an Access File, the file name and extension are: focsql and .acx. To copy a stored procedure (FOCEXEC), the file name and extension are: focexec and .fex.

CPYTOSTMF and CPYFRMSTMF are OS/400 commands provided by the operation system; they support the use of PF4 for interactive use.

Linking Files

IFS supports the creation of symbolic links between the QSYS file system and the IFS file system. A symbolic link allows the same file to be visible from both file systems. Symbolic links work only when the QSYS target is the real file and the IFS target is the virtual file. It may be thought of as a link that can be created in only one direction: QSYS to IFS. From an application perspective, this allows a file to co-exist in QSYS and IFS. If the file originated in IFS then setting up a link would require copying to QSYS, removing the IFS version, and then creating the link.

Symbolic links are created from the QSH environment as shown in the following example for a Master File:

```
ln -s /qsys.lib/myfoo.lib/master.file/foo.mbr \
      /home/iadmin/ibi/srv52/ffs/catalog/foo.mas
```

To create a link for an Access File, the file name and extension are: focsql and .acx. To create a link for a stored procedure (FOCEXEC), the file name and extension are: focexec and .fex.

Using EDAPATH

EDAPATH, long used on other server platforms, was introduced to the iWay Server for OS/400 in Release 5.1. As of Release 5.2, path searches use the APP method, which was also available in Release 5.1, but was not the default method. In Release 5.2, you can continue to use EDAPATH as documented below if APP is disabled (in the Web Console), but it is advised that you switch to the APP method as describe in *Using APP* on page 35-6.

EDAPATH is the search path for iWay related applications such as FOCEXECs (.fex), Master Files (.mas), Access Files (.acx), and FOCUS data sources. In an IFS context, EDAPATH may be set to one or more directories, separated by colons. In a QSYS context, EDAPATH may be set to one or more IFS references for QSYS libraries, separated by colons. The EDAPATH value may be set to any number of intermixed IFS or QSYS references, separated by colons.

Path search against EDAPATH for QSYS library references makes the standard EDA 3.x and 4.x application library members of MASTER, FOCSQL, and FOCEXEC available for use by iWay 5.x Servers. Thus, any of the tools documented here (that write files to libraries) may continue to be used without the need to copy or link files, provided EDAPATH is set appropriately.

For instance, if there were accounting and shipping libraries respectively named ACCTNG and SHIP libraries from applications built with Release 4.3.1, you would access them by issuing the following:

```
SET EDAPATH = /QSYS.LIB/ACCTNG.LIB : /QSYS.LIB/SHIP.LIB
```

If SHIP applications were moved to IFS, then the specification would be:

```
SET EDAPATH = /QSYS.LIB/ACCTNG.LIB : /home/iadmin/ship
```

EDAPATH is typically set as a command in the global server profile, edasprof.prf. It may also be set via a remote procedure call, or as an environment variable before server start up. This is done by using WRKENVVAR or QSH export of the variable to the list of desired (colon-separated) references.

Using APP

APP has a feature called APP MAP which allows customers to create aliases to directories outside the APPROOT location so that they can then be used in an APP PATH, APP PREPENDPATH, or APP APPENDPATH command. An actual map does not automatically place the location on the path, it is a two-step process. Below is how it would be coded in the edasprof.prf, however, these names and values can also be set using the Web Console.

```
APP MAP ACCTNG /QSYS.LIB/ACCTNG.LIB
APP MAP SHIP /QSYS.LIB/SHIP.LIB
APP PATH QMETADATA SHIP IBISAMP
```

If one of the applications was moved to IFS, then the specification would be:

```
APP MAP ACCTNG /QSYS.LIB/ACCTNG.LIB
APP PATH QMETADATA SHIP IBISAMP
```

Generating Metadata

Generating metadata for PeopleSoft World is a two-step process:

1. Generate Master and Access Files using either AUTOSQL, AUTO400, or the Web Console.
2. Use the JDECONV utility to further configure the Master and Access Files.

Reminder: iWay metadata is made up of a Master File and an optional Access File.

Before using any of the tools, verify that your library list includes tools library and either QTEMP or a writable CURLIB library (use DSPLIBL to view and EDTLIBLE to change).

When creating metadata, the auto facilities automatically create MASTER and FOCSQL files and members as needed in the selected target library if they do not exist. The library and files may also be precreated using the following commands:

```
CRTLIB library-name
```

```
CRTSRCPF library-name/MASTER RCDLEN(92)
```

```
CRTSRCPF library-name/FOCSQL RCDLEN(92)
```

AUTOSQL generates members to MASTER and FOCSQL where AUTO400 only generates members to MASTER as detailed in the respective sections. JDECONV works against existing members in MASTER.

Using AUTOSQL to Generate Master and Access Files

AUTOSQL is the automated tool for describing Master Files and Access Files for DB2/400 tables and accesses the data via OS/400's SQL facility. This tool creates a Master and Access File for each specified target (wildcard specifications are allowed) as a member (of the same name), respectively, in the source physical files MASTER and FOCSQL of the specified library.

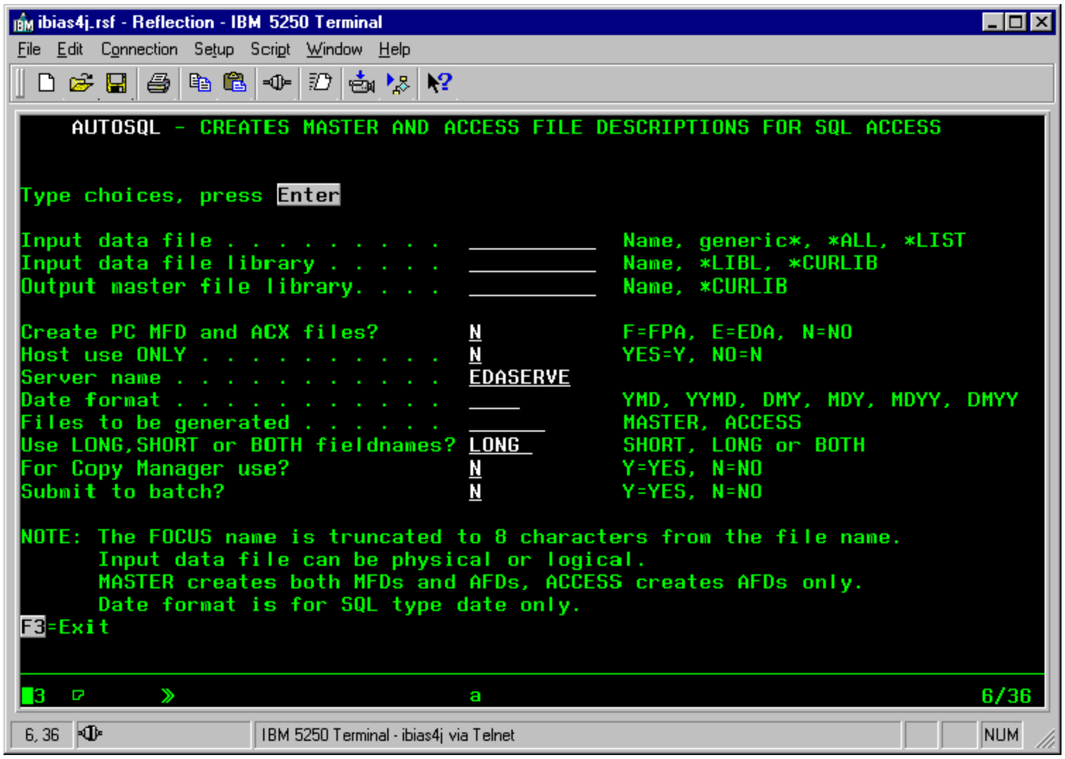
For an iWay 5.x Server to access the resulting descriptions from AUTOSQL, the file(s) may be copied or symbolic linked to a directory on the EDAPATH or APP PATH, or EDAPATH may be set to see the QSYS library.

The adapter suffix in the created Master File member is SQL400 (which is an alias to DB2) and indicates to use the SQL facility for underlying data access. Files that use SUFFIX=SQL400 use an Access File to indicate the physical location.

To run AUTOSQL (it is assumed that library I52TOOLS has been added to your library list). At the command line, enter the following command and press *Enter*:

```
AUTOSQL
```

The following menu appears:


ibias4j.rsf - Reflection - IBM 5250 Terminal
File Edit Connection Setup Script Window Help

```
     AUTOSQL - CREATES MASTER AND ACCESS FILE DESCRIPTIONS FOR SQL ACCESS


Type choices, press Enter

Input data file . . . . . . . . .   __________   Name, generic*, *ALL, *LIST
Input data file library . . . . .   __________   Name, *LIBL, *CURLIB
Output master file library. . . .   __________   Name, *CURLIB

Create PC MFD and ACX files?        N            F=FPA, E=EDA, N=NO
Host use ONLY . . . . . . . . . .   N            YES=Y, NO=N
Server name . . . . . . . . . . .   EDASERVE
Date format . . . . . . . . . . .   ____         YMD, YYMD, DMY, MDY, MDYY, DMYY
Files to be generated . . . . . .   ______       MASTER, ACCESS
Use LONG,SHORT or BOTH fieldnames?  LONG         SHORT, LONG or BOTH
For Copy Manager use?               N            Y=YES, N=NO
Submit to batch?                    N            Y=YES, N=NO

NOTE: The FOCUS name is truncated to 8 characters from the file name.
      Input data file can be physical or logical.
      MASTER creates both MFDs and AFDs, ACCESS creates AFDs only.
      Date format is for SQL type date only.
F3=Exit


 3                                  a                                      6/36
```

Fill in the fields as described in AUTOSQL Parameters and press *Enter*. An operation completed screen displays with a continue prompt.

To rerun AUTOSQL to describe another file, type *Y*.

To end the procedure, type *N*.

Reference: AUTOSQL Parameters

Input Field	Option	Description
Input data file Required, no default	Specifies the files that AUTOSQL will use to create Master and Access Files. You can specify these files in one of four ways:	
	Name	Enter the name of a single file to create one Master File and one Access File. A Master File member will be created in the first writable MASTER file in your library list using the file name. Also, an Access File member will be created in the first writable FOCSQL file in your library list using the file name.
	Generic*	To create a subset of Master Files and Access Files that begin with the same set of generic characters, such as `AB*` or `F09*` or `A1231*` Only those files beginning with the generic prefix will have Master File and Access File metadata created.
	*ALL	All files in the designated library will have Master File and Access File metadata created.
	*LISTname	To use a list of files to be converted, first create that list of file names using STRSEU. That file must be created as a member of the file IBILIST somewhere in your library list. (IBILIST is a standard source physical file with a record length of 92). To convert that list, place the list name – the name of the member – preceded by an asterisk – in this field. For instance, if you created a list called MYLIST as a member of IBILIST, you would enter *MYLIST in this field.
Input data file library Required, no default.	Name	Specifies the library containing the data files to create the Master File and Access File metadata against.

Input Field	Option	Description
Output master file library Required, no default	Name	Specifies the destination library that will contain the Master Files that are created by the metadata process. This library must exist; the utility will not create it for you. You may use the following OS/400 command to create the library: `CRTLIB` *`libraryname`* The Master Files will be created using the source file name as members in the file MASTER. Note that the MASTER file will be created if it is not there.
Create PC MFD and ACX files? default is N	F	Creates SUFFIX =FPA Master Files for use on the PC with the FOCUS Personal Agent Server. These additional Masters are placed in the file FPAMAS in the same library as the OS/400 Master Files.
	E	Creates SUFFIX =EDA Master Files for use with PC WebFOCUS Developer Studio. These additional Masters are placed in the file EDAMAS in the same library as the OS/400 Master Files.
	N	Does not create Master Files for PC use.
Host use ONLY default is N	Y	Yes. The Master File will not be used for access from a client-server environment such as WebFOCUS Developer Studio.
	N	No. The Master File will be used for access from a client-server environment such as WebFOCUS Developer Studio.
Server Name No default		Inserts the name of the Server on OS/400 that will provide access to the data sources. This value is only used for the SUFFIX=EDA Access File for PC access.
Date format Required, no default	YMD YYMD DMY MDY MDYY DMYY	Inserts one of these supported DATE formats. This format will be used only if the field is described as a DATE field type to the OS/400. If that is the case, the USAGE will become the value selected here, and the ACTUAL will be DATE.

Input Field	Option	Description
Files to be generated Required, no default	MASTER	Creates Master (MASTER) Files and Access (FOCSQL) Files for each metadata creation.
	ACCESS	Creates Access (FOCSQL) files only for each metadata creation. Use this option when data files have been moved in your system and you want to rewrite the Access Files versus manually editing.
Use LONG, SHORT or BOTH fieldnames? default is LONG	LONG	For the MASTER file FIELDNAME, uses the DSPFFD Column Heading of the field, if it exists, else use the DSPFFD field "Field". Always use "Field" as the ALIAS in the Master File. For the EDAMAS or FPAMAS file creations, ALIAS always uses value of FIELDNAME so they match. DSPFFD Column Headings with blanks or special characters get filtered into underscores in the resulting field.
	SHORT	For the MASTER file FIELDNAME and ALIAS, always uses DSPFFD field "Field" value. For the EDAMAS or FPAMAS file creations, ALIAS always uses value of FIELDNAME so they match.
	BOTH	For the MASTER file FIELDNAME and ALIAS, uses the DSPFFD Column Heading of the field, if it exists. Otherwise, uses the DSPFFD field "Field". DSPFFD Column Headings with blanks or special characters get filtered into underscores in the resulting field. For the EDAMAS or FPAMAS file creations, ALIAS always uses value of FIELDNAME so they match.
For Copy Manager use? default is N	**Note:** Copy Manager has been renamed DataMigrator.	
	Y	Yes. Field names will be truncated if they exceed the maximum of 30 characters.
	N	No. Field names are not adjusted for length.

Input Field	Option	Description
Submit to batch? default is N	Y	Yes. Runs this process in background in the QBATCH subsystem.
	N	No. Runs this process in foreground.

Using AUTO400 to Generate Master and Access Files

Reference:

AUTO400 Parameters

AUTO400 is the automated tool for describing Master Files for DB2/400 tables and accesses the data via OS/400's OPNQRYF facility. This tool creates a single Master File for each specified target (wildcard specifications are allowed) as a member (of the same name) in the source physical file MASTER of the specified library.

For an iWay 5.x Server to access the resulting descriptions from AUTO400, the file(s) may be copied or symbolic linked to a directory on the EDAPATH or APP PATH, or EDAPATH may be set to see the QSYS library.

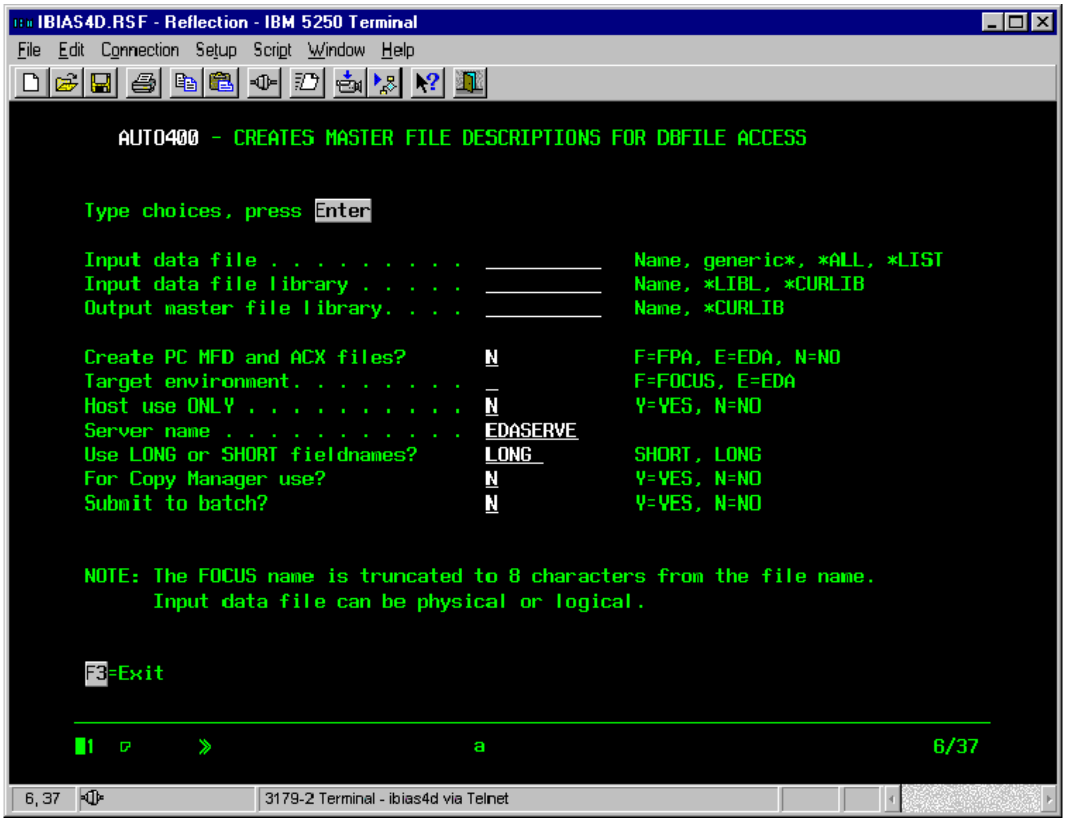
The adapter suffix in the created member is DBFILE and indicates to use the OPNQRYF facility for underlying data access. Files that use SUFFIX=DBFILE also require a FILEDEF to specify the physical location and may be issued either in an application remote procedure call or in the server's edasprof.prf. The syntax for the FILEDEF is:

```
FILEDEF AR DISK ACCTNG/AR
```

To run AUTOSQL (it is assumed that library I52TOOLS has been added to your library list). At the command line, enter the following command and press *Enter*:

```
AUTO400
```

The following menu appears:


IBIAS4D.RSF - Reflection - IBM 5250 Terminal
File Edit Connection Setup Script Window Help

```
   AUTO400 - CREATES MASTER FILE DESCRIPTIONS FOR DBFILE ACCESS

 Type choices, press Enter

 Input data file . . . . . . . . .   __________   Name, generic*, *ALL, *LIST
 Input data file library . . . . .   __________   Name, *LIBL, *CURLIB
 Output master file library. . . .   __________   Name, *CURLIB

 Create PC MFD and ACX files?        N            F=FPA, E=EDA, N=NO
 Target environment. . . . . . . .   _            F=FOCUS, E=EDA
 Host use ONLY . . . . . . . . . .   N            Y=YES, N=NO
 Server name . . . . . . . . . . .   EDASERVE
 Use LONG or SHORT fieldnames?       LONG         SHORT, LONG
 For Copy Manager use?               N            Y=YES, N=NO
 Submit to batch?                    N            Y=YES, N=NO


 NOTE: The FOCUS name is truncated to 8 characters from the file name.
       Input data file can be physical or logical.


 F3=Exit

 1                                a                               6/37
```

Fill in the fields as described in AUTO400 Parameters and press *Enter*. An operation completed screen displays with a continue prompt.

To rerun AUTO400 to describe another file, type *Y*.

To end the procedure, type *N*.

Reference: AUTO400 Parameters

Input Field	**Option**	**Description**
Input data file Required, no default	Specifies the files that AUTOSQL will use to create Master and Access Files. You can specify these files in one of four ways:	
	Name	Enter the name of a single file to create one Master File and one Access File. A Master File member will be created in the first writable MASTER file in your library list using the file name. Also, an Access File member will be created in the first writable FOCSQL file in your library list using the file name.
	Generic*	To create a subset of Master Files and Access Files that begin with the same set of generic characters, such as `AB*` or `F09*` or `A1231*` Only those files beginning with the generic prefix will have Master File and Access File metadata created.
	*ALL	All files in the designated library will have Master File and Access File metadata created.
	*LISTname	To use a list of files to be converted, first create that list of file names using STRSEU. That file must be created as a member of the file IBILIST somewhere in your library list. (IBILIST is a standard source physical file with a record length of 92). To convert that list, place the list name – the name of the member – preceded by an asterisk – in this field. For instance, if you created a list called MYLIST as a member of IBILIST, you would enter *MYLIST in this field.
Input data file library Required, no default.	Name	Specifies the library containing the data sources to create the Master File and Access File metadata against.

Input Field	Option	Description
Output master file library Required, no default	Name	Specifies the destination library that will contain the Master Files that are created by the metadata process. This library must exist; the utility will not create it for you. You may use the following OS/400 command to create the library: `CRTLIB` *`libraryname`* The Master Files will be created using the source file name as members in the file MASTER. Note that the MASTER file will be created if it is not there.
Create PC MFD and ACX files? default is N	F	Creates SUFFIX =FPA Master Files for use with WebFOCUS Developer Studio. These additional Masters are placed in the file FPAMAS in the same library as the OS/400 Master Files.
	E	Creates SUFFIX =EDA Master Files for use with WebFOCUS Developer Studio. These additional Masters are placed in the file EDAMAS in the same library as the OS/400 Master Files.
	N	Does not create Master Files for PC use.
Host use ONLY default is N	Y	Yes. The Master File will not be used for access from a client-server environment such as WebFOCUS Developer Studio.
	N	No. The Master File will be used for access from a client-server environment such as WebFOCUS Developer Studio.
Target environment	F	Creates Master Files specifically for use with FOCUS. Specifically, creates Master Files with Group attribute RECFORM when describing Multi-Record Format Logical files.
	E	Creates Master Files specifically for use with iWay. Specifically, creates Master Files with a Field=RECFORM when describing Multi-Record Format Logical files.
	N	No. The Master File will be used for access from a client-server environment such as WebFOCUS Desktop.

Input Field	Option	Description
Server Name No default		Inserts the name of the Server on OS/400 that will provide access to the data files. This value is only used for the SUFFIX=EDA Access File for PC access.
Date format Required, no default	YMD YYMD DMY MDY MDYY DMYY	Inserts one of these supported DATE formats (). This format will be used only if the field is described as a DATE field type to the OS/400. If that is the case, the USAGE will become the value selected here, and the ACTUAL will be DATE.
Files to be generated Required, no default	MASTER	Creates Master (MASTER) Files and Access (FOCSQL) Files for each metadata creation.
	ACCESS	Creates Access (FOCSQL) files only for each metadata creation. Use this option when data files have been moved in your system and you want to re-write the access files versus manually editing.
Use LONG or SHORT fieldnames? default is LONG	LONG	For the MASTER file FIELDNAME, uses the Row Description of the field, it exists; otherwise, uses the Field name. Uses the Field name as the ALIAS in the Master File. For the EDAMAS or FPAMAS file FIELDNAME, uses the Row Description of the field if it exists; otherwise uses the Field Name. The ALIAS should be the same as the FIELDNAME. The ALIAS of the EDAMAS must match the FIELDNAME of the MASTER file on the host.
	SHORT	For the MASTER file FIELDNAME, uses the field name of the field. Also uses the Field name as the ALIAS in the Master File. For the EDAMAS or FPAMAS file FIELDNAME, uses the field name of the field. The ALIAS should be the same as the FIELDNAME. The ALIAS of the EDAMAS must match the FIELDNAME of the MASTER file on the host.
For Copy Manager use? default is N	**Note:** Copy Manager has been renamed DataMigrator.	
	Y	Yes. Fieldnames will be truncated if they exceed the maximum of 30 characters.
	N	No. Fieldnames are not adjusted for length.

Input Field	Option	Description
Submit to batch? default is N	Y	Yes. Runs this process in background in the QBATCH subsystem.
	N	No. Runs this process in foreground.

Using JDECONV

Reference:

JDECONV Parameters

If you will be accessing PeopleSoft World data, you will need to use the JDECONV utility to apply metadata changes to the file members built by AUTOSQL or AUTO400 tools. JDECONV updates the members (Master Files) with the metadata information dynamically derived from the PeopleSoft World Data Dictionary files. If JDECONV is not run, you will not be able to exploit any changes to the data found in the PeopleSoft World Data Dictionary.

JDECONV can only locate existing Master Files in a library's MASTER file.

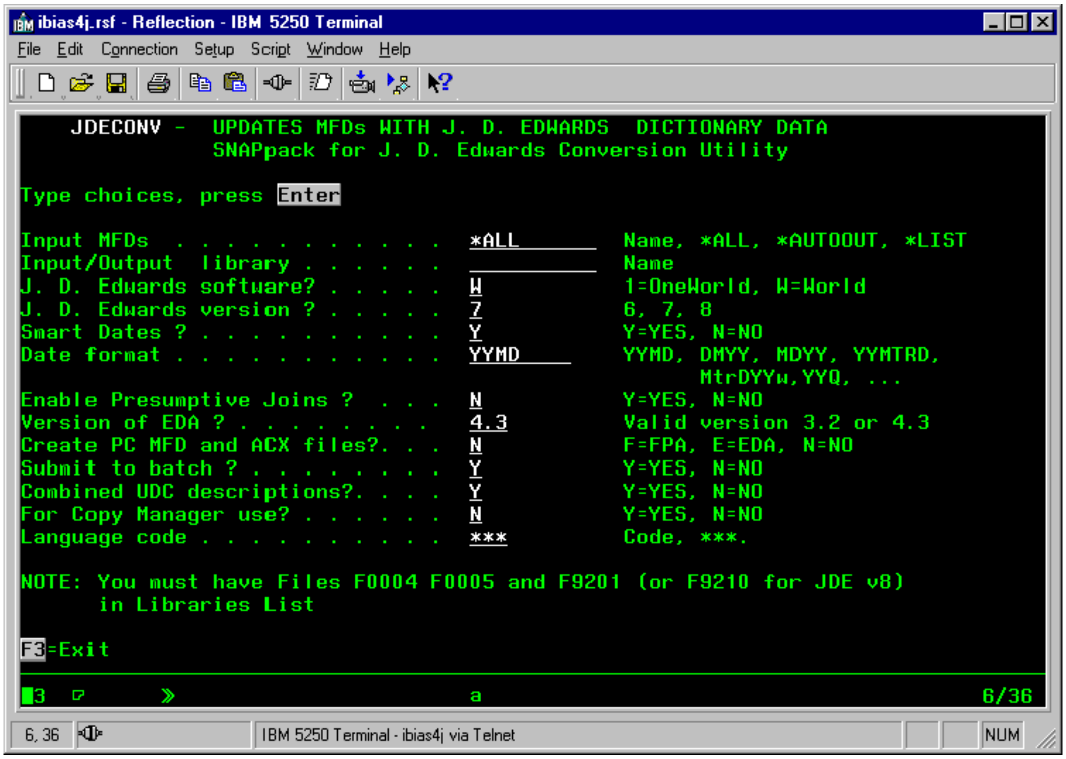
For an iWay 5.x server to access the resulting descriptions converted by JDECONV, the file(s) may be copied or symbolic linked to a directory on the EDAPATH or APP PATH, or EDAPATH may be set to see the QSYS library.

The PeopleSoft World Data Dictionary Library must be present in your library list before running the JDECONV utility (use EDTLIBLE or ADDLIBLE to add). To run JDECONV, (it is assumed that library I52TOOLS has been added to your library list), at the command line enter the following command and press Enter:

```
JDECONV
```

If the PeopleSoft World library is not in the library path a message will display at the bottom of the screen (as shown below).

The following menu appears:



ibias4j.rsf - Reflection - IBM 5250 Terminal
File Edit Connection Setup Script Window Help

```
     JDECONV -  UPDATES MFDs WITH J. D. EDWARDS  DICTIONARY DATA
                SNAPpack for J. D. Edwards Conversion Utility

Type choices, press Enter

Input MFDs . . . . . . . . . . . .  *ALL         Name, *ALL, *AUTOOUT, *LIST
Input/Output  library . . . . . .                Name
J. D. Edwards software? . . . . .   W            1=OneWorld, W=World
J. D. Edwards version ? . . . . .   7            6, 7, 8
Smart Dates ? . . . . . . . . . .   Y            Y=YES, N=NO
Date format . . . . . . . . . . .   YYMD         YYMD, DMYY, MDYY, YYMTRD,
                                                       MtrDYYw,YYQ, ...
Enable Presumptive Joins ?  . . .   N            Y=YES, N=NO
Version of EDA ? . . . . . . . .    4.3          Valid version 3.2 or 4.3
Create PC MFD and ACX files?. . .   N            F=FPA, E=EDA, N=NO
Submit to batch ? . . . . . . . .   Y            Y=YES, N=NO
Combined UDC descriptions?. . . .   Y            Y=YES, N=NO
For Copy Manager use? . . . . . .   N            Y=YES, N=NO
Language code . . . . . . . . . .   ***          Code, ***.

NOTE: You must have Files F0004 F0005 and F9201 (or F9210 for JDE v8)
      in Libraries List

F3=Exit

 3                                     a                                6/36
```

Fill in the fields as described in JDECONV Parameters and press *Enter*. An operation completed screen displays with a continue prompt.

To rerun JDECONV for another file, type *Y*.

To end the procedure, type *N*.

Reference: JDECONV Parameters

Input Field	Option	Description
Input MFDs Required, no default	Specifies the files that JDECONV will operate on. You can specify a file in one of four ways:	
	Name	To convert a single Master File (MFD) insert the name of a single file. The EDA Master File member will be converted and updated.
	Generic*	To create a subset of Master Files and Access Files that begin with the same set of generic characters, such as `AB*` or `F09*` or `A1231*` Only those files beginning with the generic prefix will have Master File and Access File metadata created.
	*ALL	All files in the designated library will have Master Files converted.
	*LISTname	To use a list of files to be converted, first create that list of file names using STRSEU. That file must be created as a member of the file IBILIST somewhere in you library list. (IBILIST is a standard source physical file with an record length of 92). To convert that list, place the list name – the name of the member – preceded by an asterisk – in this field. For instance, if you created a list called MYLIST as a member of IBILIST, you would insert *MYLIST in this field.
Input data file library Required, no default.	Name	Specifies the library containing the data files to create the Master File and Access File metadata against.
J. D. Edwards World default is W	1	Specifies that you are using J. D. Edwards World (PeopleSoft World) on OS/400.
	W	Specifies that you are using J. D. Edwards World (PeopleSoft World) on OS/400.

Input Field	Option	Description
J. D. Edwards World version default is 7	6	Specifies that you are using the 6.x versions of the J. D. Edwards World (PeopleSoft World).
	7	Specifies that you are using the 7.x versions of the J. D. Edwards World (PeopleSoft World).
	8	Specifies that you are using the 8.x versions of the J. D. Edwards World (PeopleSoft World).
Smart Dates default is Y	Y	Yes. Applies the Date format listed below to all J. D. Edwards World (PeopleSoft World) dates within the table(s) to be converted, using a format of P6JUL.
	N	No. Ignores the Date format listed below and displays all date formats as YMD via a DEFINE field using the GREGDT subroutine.
Date format default is YYMD	YMD YYMD DMY MDY MDYY DMYY	Inserts one of these supported DATE formats. This format will be used only if the field is described as a DATE in the Data Dictionary. If that is the case, the USAGE will become the value selected here, and the ACTUAL will be P6JUL.
Enable Presumptive JOINs default is N	Y	Yes. All fields in the MASTER file that contain Data Dictionary Items AN8, MCU and CO will have new DEFINE fields added for XREF files.
	N	No. Does not enable Presumptive JOINs.
Version of EDA default is 4.3	3.2	Specifies that you are using EDA 3.2.2 on OS/400.
	4.3	Specifies that you are using EDA 4.3.1 or iWay 5.x on OS/400.

Input Field	Option	Description
Create PC MFD and ACX files? default is N	F	Creates SUFFIX =FPA Master Files for use with WebFOCUS Developer Studio. These additional Masters are placed in the file FPAMAS6 or FPAMAS7 (depending on your software version) in the same library as the OS/400 Master Files. Use the AS4XFER utility to retrieve these files from the PC.
	E	Creates SUFFIX =EDA Master Files for use with WebFOCUS Developer Studio. These additional Masters are placed in the file EDAMAS6 or EDAMAS7 (depending on your software version) in the same library as the OS/400 Master Files. Use the SEAS4X utility to retrieve these files from the PC.
	N	Does not create Master Files for PC use.
Submit to batch default is Y	Y	Yes. Submits the program to batch.
	N	No. Submits the program to run on-line.
Combined UDC Description default is Y	Y	Yes. Creates one new DEFINE field for each UDC code. This new DEFINE field will be the combination of 2 UDC descriptions for each UDC code.
	N	No. Creates up to two new DEFINE fields for each UDC code. Each new DEFINE field will contain one of the UDC descriptions.
For Copy Manager use?	**Note:** Copy Manager has been renamed DataMigrator.	
	Y	Yes. Fieldnames will be truncated if they exceed the maximum of 30 characters.
	N	No. No truncation of fieldnames will occur.

Input Field	**Option**	**Description**
Language Code default is ***	***	Determines whether you'll be using the field descriptions and titles set within PeopleSoft World data dictionary for the entered Language Code. Any field descriptions and titles set at the Major System are also used. Blank is the standard J. D. Edwards default for Domestic Language. Using the default, *******, will use the field descriptions and titles set within the DB2/400 DDS only.
	Name	Language Code value.

Using Master Files and Access Files

Reference:

Master File Attribute Sources

Access File Attribute Sources

The following two tables describe the attributes of the Information Builders metadata (stored in Master Files and Access Files), and how it relates to PeopleSoft World metadata.

Reference: Master File Attribute Sources

Attribute	**Source**
`FILENAME`	DDS File name
`FIELDNAME`	DDS row description/JDE F9202 table
`ALIAS`	DDS column name
`USAGE`	JDE data display rules/edit rules/JDE display decimals
`ACTUAL`	JDE type and length
`ACCEPT`	JDE acceptance criteria
`TITLE`	JDE column name/JDE F9202 table
`DEFINE`	JDE UDC's/JDE data item class
`Business Unit Security`	JDE MCU
`Search Type Security`	JDE AT1
`User Defined Codes`	DE F0004, F0005 and F9201 tables

Reference: Access File Attribute Sources

Attribute	Source
`TABLENAME`	DDS File name
`KEYS`	DDS key information
`KEYFLD`	DDS key information
`IXFLD`	DDS key information
`WRITE`	"NO"

Enabling PeopleSoft World Security

In order to enable PeopleSoft World security, you must start your server with any of the following parameters:

```
JDED=ON;
```

```
JDEN=ON;
```

```
JDEW=ON;
```

Note that by default, the JDED, JDEN, and JDEW parameters are OFF.

You can set these parameters either by modifying the CL source or by performing the WRKENVVAR command. For more information on the CL source, see the *iWay Server Installation* manual; for more information on WRKENVVAR, see your OS/400 documentation.

The JDED parameter, when set ON, enables automatic execution of PeopleSoft World Business Unit Security. The Server for OS/400 automatically restricts user access to data based on information retrieved from the F0001 and F0006 tables, and by then adding appropriate WHERE conditions to the users's submitted data access request. Once this has been set ON, it cannot be turned OFF until the server is shut down and then restarted (with no parameter settings).

The JDEN parameter, when set ON, enables automatic execution of PeopleSoft World Search Type Security. The Server for OS/400 server automatically restricts user access to data based on information retrieved from the F0005 table, and by then adding appropriate WHERE conditions to the users's submitted data access request. Once this has been set ON, it cannot be turned OFF until the server is shut down and then restarted (with no parameter settings).

The JDEW parameter handles column security per the F9401 table.

Using the Report Library

One of the most powerful features of the Adapter for PeopleSoft World is the ease with which users can customize reports. Within the reporting environments it is a simple matter to add or old columns, to change the display characteristics of report columns, to create new or different page breaks, or to create matrix reports. The list is virtually endless.

As such, the adapter contains a Report Library. The Report Library is a repository of standard PeopleSoft World reports, coded in the WebFOCUS language rather than in RPG. These reports were coded by PeopleSoft World users and have been contributed to this library of standard reports.

Please feel free to use these reports. Change them to suit your needs. You will see a dramatic reduction in coding time when changing the WebFOCUS reports rather than the standard RPG reports. This is one of the major strengths of the WebFOCUS language - productivity.

The following reports are currently included in the Report Library:

WebFOCUS Developer Studio Procedure	**RPG Procedure**	**Report Type**
F094121	P094121	General Ledger
F068515	P068515	EEO
F063002	P063002	Time/Pay

Enabling Tracing

You can easily trace any problem that arises, using the Web Console tracing facility. For more information, see the *iWay Server Administration* manual.

Major System Codes

The following codes apply to J. D. Edwards Versions 6.x, 7.x, and 8.x.

Major System Code and Name		**File Prefix**
00	World Foundation Environment	F00*
01	Address Book	F01*
02	Electronic Mail	F02*
03	Accounts Receivable	F03*
04	Accounts Payable	F04*
05	Stand-Alone Time Accounting	F05*
06	Payroll	F06*
07	Payroll “Enhanced”	F07*
08	Human Resources	F08*
09	General Accounting	F09*
10	Financial Reporting	F10*
11	Multi Currency/Cash Basis	F11*
12	Fixed Assets	F12*
13	Equipment/Plant Management	F13*
14	Modeling, Planning, & Budgeting	F14*
15	Commercial Property Management	F15*
16	Resident Property Management	F16*
17	Property Management Base	F17*
18	Deal Management	F18*
20	Energy Base	F19*
30	Product Data Management	F30*
31	Shop Floor Control	F31*

Major System Code and Name		**File Prefix**
32	Configuration Management	F32*
33	Capacity Requirements Planning	F33*
34	DRP/MRP/MPS	F34*
35	Enterprise Facility Planning	F35*
40	Inventory/OP base	F40*
41	Inventory Management	F41*
42	Sales Order Processing	F42*
43	Purchasing Order Processing	F43*
44	Contract Management	F44*
45	Advanced Price Adjustments	F45*
46	Warehouse Management	F46*
47	Electronic Data Interchange	F47*
48	Workorder Processing	F48*
49	Load and Delivery	F49*
50	Job Cost Base	F50*
51	Job Cost Accounting	F51*
52	Job Cost Billing	F52*
53	Change Management	F53*
55-59	Client Use	F55* F56* F57* F58* F59*
60-69	JDE Internal Custom Programming	F60* F61* F62* F63* F64* F65* F66* F67* F68* F69*
70	Multi-National Products	F70*
71	Client/Server Applications	F71*
72	World Vision	F72*
73	CS - A/P Entry	F73*

Major System Code and Name		**File Prefix**
74	CS - Pay Time Entry	F74*
75	CS - Sales Order Entry	F75*
76	CS - Training and Development	F76*
77	Canadian Payroll	F77*
79	CS - Translation	F79*
80	COBOL Translator	F80*
81	DREAM Writer	F81*
82	World Writer	F82*
83	Management Reporting - FASTR	F83*
84	Distributive Data Processing	F84*
85	Custom Programming	F85*
86	Electronic Document Interchange	F86*
87-98	Miscellaneous Tech	F87* F88* F89* F90* F91* F92* F93* F94* F95* F96* F97* F98*

Frequently Asked Questions

I ran a report against a JDE file, however, no converted precision is coming back, why?

Confirm if your request is using the short JDE field names or the long field names. If your request is using short names, then it sounds like the request is being processed via APT (Automatic Passthru), which goes directly through the SQL engine and bypasses the MASTER file format changes. To solve this issue, add the following 3 lines to the bottom of the edasprof.prf:

```
SQL
SET APT = OFF;
END
```

Remember, in iWay 5.x the edasprof.prf is in ibi/srv5*/*/etc and not in a library as done in 4.x servers.

Which library should I store my MASTER, FOCSQL and FOCEXEC files in?

We highly recommend storing all MASTER, FOCSQL and FOCEXEC files in a separate library or directory from where the server was installed so that you can add to the server's path prior to server start-up time, using either APP PATH or EDAPATH methods. This will avoid not only confusion as to where they reside, but will also make it easier to maintain during server upgrades.

I've been using the Adapter for PeopleSoft World under an EDA 3.2.2 server and I'm upgrading to a newer server level. Will the MASTER files I built under EDA 3.2.2 work with EDA 4.3.1 or iWay 5.x?

No, existing MASTER files built with JDECONV from EDA 3.2.2 are not supported under EDA 4.3.1 or iWay 5.x. Enhancements have been made to the JDECONV tool for 4.3.1 to accommodate the EDA 4.3.1 architecture. Specifically, any fields for which precision is defined in the PeopleSoft World data dictionary now have a JDE suffix added to the USAGE attribute in the MASTER file. For example:

In EDA 3.2.2:

```
FIELD=BAL_FWD, ALIAS='GBAPYC', USAGE=P16.2NS, ACTUAL=P8.0,
 DESCRIPTION='Bal FWD', TITLE='Beg Balance/, PYE Forward', $
```

In EDA 4.3.1 or higher:

```
FIELD=BAL_FWD, ALIAS='GBAPYC', USAGE=P16.2NS, ACTUAL=P8.0JDE,
 DESCRIPTION='Bal FWD', TITLE='Beg Balance/, PYE Forward', $
```

We suggest saving your existing EDA 3.2.2 MASTER files as a backup, and re-creating the metadata with the latest tools.

CHAPTER 36

Using the Adapter for Progress

Topics:

The Adapter for Progress allows applications to access Progress data sources. The adapter converts application requests into native Progress statements and returns optimized answer sets to the requesting application.

Preparing the Progress Environment

Currently, the Progress environment is supported on UNIX and Windows. The Adapter for Progress is available as an ODBC interface on all platforms except Compaq/Alpha Tru64. On Compaq/Alpha Tru64 only the embedded SQL interface is available.

Procedure: How to Set Up the Environment on Windows Using ODBC

On Windows, the Progress environment is set up during the installation of the product.

Procedure: How to Set Up the Environment on UNIX Using ODBC

1. Specify the full name of the directory containing Progress.

   ```
   DLC = /rdbms/pro91d/dlc
   ```

2. Specify the full name of the default Progress startup file.

   ```
   PROSTARTUP = /rdbms/pro91d/dlc/startupyy.pf
   ```

3. Specify the location of the ODBC initialization file.

   ```
   ODBCINI = /usr/odbc/ibiodbc/progress.ini
   ```

4. Specify the path to the Progress shared libraries.

   ```
   LIBPATH=/rdbms/pro91d/dlc/odbc/lib : /rdbms/pro91d/dlc/lib
   ```

Procedure: How to Set Up the Environment on Compaq/Alpha Tru64 (non-ODBC)

1. Specify the location where Progress is installed.

   ```
   PRO_HM = /rdbms/pro91d/qaeda
   ```

2. Specify the file that contains the Progress text messages.

   ```
   PROMSGS = /rdbms/pro91d/dlc/promsgs
   ```

3. Specify the full name of the directory containing Progress.

   ```
   DLC = /rdbms/pro91d/dlc
   ```

4. Specify the full name of the directory where Progress PROBUILD is installed.

   ```
   PROLOAD = /rdbms/pro91d/dlc/probuild
   ```

5. Specify the full name of the default Progress startup file.

   ```
   PROSTARTUP = /rdbms/pro91d/dlc/startupy
   ```

Configuring the Adapter for Progress

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Connecting to a Progress Database Server

For an ODBC interface, the server must be brought up in TCP/IP mode. On Compaq/Alpha Tru64 platforms you may bring up the server in either shared memory or TCP/IP mode.

Shared memory is an area in system memory that multiple users can access concurrently. Windows and most UNIX systems use shared memory. The version number must be the same for all processes that access shared memory configurations. Both the client and server must be running the same release and version number.

Using TCP/IP, you can run different versions of the client and server. For example, if the database version is 8, the client can be version 9. If the database is version 9, a version 8 client can not communicate with it. Clients are able to connect to database servers using TCP/IP within the same version and one version prior. Once the database is brought up using TCP/IP (when the server is started using -H, -S, and -N parameters), the FOCPF global variable must be set under the server.

Example: Setting the FOCPF Global Variable

```
Export FOCPF=/u1/home/tcp.pf
```

The file tcp.pf:

```
-S sssss -H mmmm -N TCP
```

```
-S service name
```

```
-H host name
```

```
-N network type
```

These parameters must match the parameters used when you connected to the database using TCP/IP.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

Example:

Declaring Connection Attributes

In order to connect to an Progress database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Progress database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Progress Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Field	Description
Datasource	Progress data source.
User	Valid user for Progress.
Password	Valid password for Progress.

The Progress non-ODBC adapter configuration screen displays the following fields:

Field	Description
Database name	Progress database name.
Home directory	Full path to the database location.
User	Valid user for Progress.
Password	Valid password for Progress.

4. Indicate the level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default.

   If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)
5. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

```
ENGINE [SQLPRO] SET CONNECTION_ATTRIBUTES [datasource]/userid,password
```

where:

`SQLPRO`

Indicates the Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

datasource

Is the name of the Progress data source you wish to access.

userid

Is the primary authorization ID by which you are known to Progress.

password

Is the password associated with the primary authorization ID.

The SET command also allows you to access a Progress database using the embedded SQL interface:

```
ENGINE SQLPRO SET DATABASE qaeda
ENGINE SQLPRO SET HOME /rdbms/pro91d/qaeda
ENGINE SQLPRO SET USER edaqa
ENGINE SQLPRO SET PASSWORD qaeda3x
```

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Progress database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLPRO SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Overriding the Default Connection

How to:

Change the Default Connection

Example:

Selecting the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLPRO] SET DEFAULT_CONNECTION [connection]
```

where:

SQLPRO

Indicates the Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Progress database server named SAMPLENAME as the default Progress database server:

```
ENGINE SQLPRO SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLPRO] SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

`SQLPRO`

Indicates the Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing Progress Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Progress data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Progress table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

   Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

   - **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.

   - **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.

5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.

6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.

7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value. See *Controlling the Mapping of Large Character Data Types* for more information.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLPRO [AT connection]
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

`SQLPRO`

Indicates the Data Adapter for Progress.

`AT` *connection*

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM baseapp/nf29004 for EDAQA.NF29004 DBMS SQLPRO
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLPRO ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4,     I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4 ,TABLENAME=EDAQA.NF29004,
KEYS=1, WRITE=YES ,$
```

Reference: Access File Keywords

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Name of the table or view. This value can include a location or owner name as follows: `TABLENAME=[location.][owner.]tablename`
CONNECTION	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
WRITE	Specifies whether write operations are allowed against the table.

Data Type Support

The following table lists how the server maps Progress data types. Note that you can:

- Control the mapping of large character data types.
- Change the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Progress Data Type	Data Type		Remarks
	USAGE	**ACTUAL**	
CHAR	A(2000)	A(2000)	
VARCHAR	A(31995)	A(31995)	
DATE	HYYMD	HYYMD	
TIME	HHIS	HHIS	
TIMESTAMP	HYYMDs	HYYMDs	
DECIMAL (p,s)	P14.2	P7	
NUMERIC (p,s)	P14.2	P7	
INTEGER	I11	I4	
SMALLINT	I6	I4	
TINYINT	I6	I4	
REAL	D20.2	D8	
FLOAT DOUBLE PRECISION	D20.2	D8	
LOGICAL	N/A	N/A	No longer supported under SQL-92.
BIT	I11	I4	
BINARY	A4	A4	
VARBINARY	A20	A20	
LVARBINARY	BLOB	BLOB	

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Progress data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Progress Data Type	**Remarks**	**LONGCHAR ALPHA or BLOB**		**LONGCHAR TEXT**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
CHAR (*n*)	*n* is an integer between 1 and 2000	A*n*	A*n*	TX50	TX
VARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*	A*n*	TX50	TX
VARCHAR2 (*n*)	*n* is an integer between 1 and 4000	A*n*	A*n*	TX50	TX
RAW (*n*)	*n* is an integer between 1 and 2000 *m* = 2 * *n*	A*m*	A*m*	TX50	TX

Syntax: How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLPRO] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

`SQLPRO`

Indicates the Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

`ALPHA`

Maps the Progress data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). ALPHA is the default value.

`TEXT`

Maps the Progress data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

`BLOB`

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Progress data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

For OS/390 and z/OS, maps the Progress data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Progress data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Progress Data Type	**Remarks**	**VARCHAR ON**		**VARCHAR OFF**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
VARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*V	A*n*V	A*n*	A*n*
VARCHAR2 (*n*)	*n* is an integer between 1 and 4000	A*n*V	A*n*V	A*n*	A*n*
NVARCHAR2 (*n*)	*n* is an integer between 1 and 4000	A*m*V	A*m*V	A*m*	A*m*

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLPRO] SET VARCHAR {ON|OFF}
```

where:

`SQLPRO`

Indicates the Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Progress data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (A*n*V).

`OFF`

Maps the Progress data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). OFF is the default value.

Changing the Precision and Scale of Numeric Columns

How to:

Override Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLPRO] SET CONVERSION RESET
ENGINE [SQLPRO] SET CONVERSION format RESET
ENGINE [SQLPRO] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLPRO] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLPRO`

Indicates the Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	**MAX Precision**
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLPRO SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLPRO SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLPRO SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLPRO SET CONVERSION RESET
```

Customizing the Progress Environment

In this section:

The Adapter for Progress provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Specifying the Block Size for Array Retrieval

The Adapter for Progress supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Specify the Block Size for Inserting Rows

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE [SQLPRO] SET INSERTSIZE n
```

where:

`SQLPRO`

Indicates the Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Activating NONBLOCK Mode

The Adapter for Progress has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE [SQLPRO] SET NONBLOCK {0|n}
```

where:

`SQLPRO`

Indicates the Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLPRO] SET PASSRECS {ON|OFF}
```

where:

`SQLPRO`

Indicates the Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Progress.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE [SQLPRO] SET TIMEOUT {nn|0}
```

where:

`SQLPRO`

Indicates the Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

`nn`

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Optimization Settings

In this section:

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

How to:

Optimize Requests

Example:

SQL Requests Passed to the RDBMS With Optimization OFF

SQL Requests Passed to the RDBMS With Optimization ON

Reference:

SQL Generation in Optimization Examples

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLPRO] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLPRO`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

OFF instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

<u>ON</u> instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

FOCUS passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

SQL passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLPRO set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLPRO set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLPRO] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLPRO`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLPRO SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLPRO SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLPRO SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Specifying Block Size for Retrieval Processing

How to:

Specify Block Size for Insert Processing

The Adapter for Progress supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Specify Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE [SQLPRO] SET INSERTSIZE n
```

where:

`SQLPRO`

Indicates the Adapter for Progress. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

CHAPTER 37

Using the Adapter for Rdb

Topics:

The Adapter for Rdb allows applications to access Rdb data sources. The adapter converts application requests into native Rdb statements and returns optimized answer sets to the requesting application.

Preparing the Rdb Environment

Oracle Rdb allows two types of installation:

- **Standard.** Supports only one Rdb release level on a machine. No preparation is required to use the adapter in a Standard Rdb environment.
- **Multi Version.** Supports multiple Rdb release levels on the same machine. To switch between releases, Multi Version supplies a script, which sets up logically to point to standard names, such as SQL$, at specific release files, such as SQL$61.EXE, SQL$70.EXE, and SQL$7.1.EXE.

To use the adapter in a Multi Version environment, prior to server startup, your Rdb environment must be set up to access the database file release level that corresponds to the release level selected during configuration of the adapter. To enable switching between Rdb releases, your site might use the Rdb script DECRDB$SETVER, a site-specific script, or a symbol. For details on how Rdb release level switching is implemented at your site, see your OpenVMS Administrator.

Configuring the Adapter for Rdb

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to an Rdb database server, the adapter requires connection and authentication information. You supply this information using the SET SERVER command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	**Description**
Server	Full path to an Rdb data source, or a logical name, such as SQL$DATABASE, which contains the full path. Rdb limits this value to 31 characters. If the full path exceeds this limit, you must specify a logical name to point to the full path. **Full Path.** Full path specification must be in the following form: *device_name*:[*directory*]*file*.RDB. The RDB extension is required. **Logical Name.** To use this method, the logical must be set prior to server startup. The trailing blank is optional.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

The adapter connects to Rdb using the credentials of the operating system user impersonated by the data access agent.

```
ENGINE [SQLRDB] SET SERVER server
```

where:

`SQLRDB`

Indicates the Adapter for Rdb. You can omit this value if you previously issued the SET SQLENGINE command.

`server`

Is the full path to an Rdb data source, or a logical name, such as SQL$DATABASE, which contains the full path. Rdb limits this value to 31 characters. If the full path exceeds this limit, you must specify a logical name to point to the full path.

Full Path. Full path specification must be in the following form: *device_name*:[*directory*]*file*.RDB. The RDB extension is required.

Logical Name. To use this method, the logical must be set prior to server startup. The trailing blank is optional.

Note: You can manually declare connections to more than one Rdb database server by including multiple SET SERVER commands. The actual connection to Rdb takes place when the first query that references the connection is issued. If you issue multiple SET SERVER commands:

- The connection named in the *last* SET SERVER command serves as the default connection.
- The Rdb installation has to be Standard.

Example: Declaring a Connection to ORDERS.RDB

The following example shows how to declare a connection to the ORDERS.RDB data source, located in the OUTBOUND directory on disk.

```
ENGINE SQLRDB SET SERVER DISK$SHIPPING:[OUTBOUND]ORDERS.RDB
```

Overriding the Default Connection

Once connections have been defined, the connection named in the last SET SERVER command serves as the default connection. You can override this default by reissuing the SET SERVER command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLRDB] SET SERVER [server]
```

where:

`SQLRDB`

Indicates the Adapter for Rdb. You can omit this value if you previously issued the SET SQLENGINE command.

`server`

Is the full path to an Rdb data source, or a logical name, such as SQL$DATABASE, which contains the full path. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET SERVER command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET SERVER command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET SERVER command selects the Rdb database server named SAMPLENAME as the default Rdb database server:

```
ENGINE SQLRDB SET SERVER SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLRDB] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLRDB`

Indicates the Adapter for Rdb. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Rdb Metadata

In this section:

Creating Synonyms

Data Type Support

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Rdb data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Rdb table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.
10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)

Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLRDB [AT server] [NOCOLS]
                                                      [AT '']
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLRDB

Indicates the Adapter for Rdb.

AT *server*

Is the service name as previously specified in a SET SERVER command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`AT ''`

Adds CONNECTION=' ' in the Access File. This indicates a connection to a local Rdb database server previously specified in the SET SERVER declaration.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLRDB AT DSN_A
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLRDB ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the Rdb table.
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.

Keyword	**Description**
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following chart provides information about the default mapping of Rdb data types to server data types:

Rdb Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
CHAR (*n*)	A*n*	A*n*	
VARCHAR (*n*)	A*n*	A*n*	
LONGVARCHAR	A*n*	A*n*	
TINYINT (-128...127)	I6	I4	Scale factors (n) in SQL integer data types are equivalent to negative scale factors in Oracle Rdb integer data types. SQL does not support Oracle Rdb positive scale factors.
SMALLINT (-32768... 32767)	I6	I4	
INTEGER (-2^{31}... 2^{31} - 1)	I11	I4	2^{31} = 2,147,483,648.
NUMERIC, DECIMAL (p,s)	D20.2	D8	Same as REAL or DOUBLE PRECISION. 8 bytes; value range 2 * 10^{307} to 2 * 10^{308}.
BIGINT (-2^{63} ... 2^{63} - 1)	D20.2	D8	
REAL	D20.2	D8	32-bit FLOAT(*n*), where *n* is less than 25.
DOUBLE PRECISION	D20.2	D8	64-bit FLOAT(*n*), where *n* is 25 or greater.

Rdb Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
DATE	A*n*	A*n*	Date/Time formats are comprised of several components.
TIME	A*n*	A*n*	Date/Time formats are comprised of several components.
TIMESTAMP	A*n*	A*n*	Date/Time formats are comprised of several components.

Note: TIME, TIMESTAMP, and ANSI dates are not supported using SQL Passthru or CREATE SYNONYM. Fields with these data types can only be accessed using Table Services DML after Master and Access Files are created manually. A manually created Master File defines USAGE/ACTUAL as follows:

- TIME as A8/A8 will retrieve the data in 'HH:MI:SS' format.
- TIMESTAMP as A22/A22 will retrieve the data in 'YYYY-MM-DD HH:MI:SS.SS' format.
- ANSI date as A10/A10 will retrieve the date data in 'YYYY-MM-DD' format.

Joining Rdb Tables in Separate Rdb Databases

In iWay 5.x for OpenVMS, two or more Rdb tables that exist in different physical Rdb databases may be joined. A default iWay configuration can access only one physical Rdb database at a time. The use of multiple databases is accomplished by doing a standard configuration with Rdb for a given database, creating metadata for that database, repeating the configure/metadata step for as many Rdb databases as needed, and making changes to a server's configuration as outlined below:

1. During server configuration, choose the standard option for Rdb relational access and supply the full path name to one of the desired Rdb databases.
2. Edit the EDACONF [.BIN]EDAENV.COM file and add a unique logical name for each desired database file to be accessed. Specify the full path names to each of the databases. For example:

```
$ DEFINE RDBSRV1 DISK$DUA0:[RDB]SCORES.RDB
$ DEFINE RDBSRV2 DISK$DUA0:[RDB]STUDENTS.RDB
```

3. Assuming that there are previously created Rdb Access Files, edit the Access Files for each of the Rdb databases, and change the TABLENAME value to be that of the appropriate logical name. Use a period and the original value (for example, logical.TNAME), and add a parameter of SERVER=logical.

   The following is an example of the respective Access File contents before changes:

   ```
   ..., TABLENAME=SCORES, ...
   ```

   ```
   ..., TABLENAME=STUDENTS, ...
   ```

   An example of the respective Access File contents after changes is:

   ```
   ..., TABLENAME=RDBSRV1.SCORES, SERVER=RDBSRV1, ...
   ```

   ```
   ..., TABLENAME=RDBSRV2.STUDENTS, SERVER=RDBSRV2, ...
   ```

   where:

   `RDBSRV1`

   Is a unique logical for the database in question.

   `RDBSRV2`

   Is a unique logical for the database in question.

   `SCORES`

   Is a table name.

   `STUDENTS`

   Is a table name.

4. Edit the EDASPROF.PRF file, comment out the SET SERVER command for the database specified during configuration, and add:

   ```
   SQL
   SET APT=OFF
   END
   ```

   You can use FOCUS JOIN statements or SQL JOIN syntax to retrieve data. This access method is not supported for use with direct passthru (for example, SET SQLENGINE=SQLRDB) to join the tables. If you wish to use direct passthru for access to one Rdb database, you will need to add back the SET SERVER command in EDASPROF.PRF for that database.

Rdb Database Driver Performance

The default of Rdb is to open tables in read/write mode unless told otherwise. iWay supports read/write operations, so this is appropriate behavior. However, Rdb read/write opens consume more resources and are slower even if just a select is being done. iWay applications can set Rdb opens to Read-only on a per request basis or a per session basis to enhance resource usage and speed performance. In either case, a commit should be done to ensure that prior work is complete.

On a session basis, add the following to the EDASPROF.PRF after the SQL SQLRDB SET SERVER command:

```
SQLRDB COMMIT ;
SQL SQLRDB SET DECLARE TRANSACTION READ ONLY ;
```

On a per request basis (for example, immediately before a TABLE request) the usage is:

```
SQLRDB COMMIT ;
SQL SQLRDB SET TRANSACTION READ ONLY ;
```

CHAPTER 38

Using the Adapter for Red Brick

Topics:

The Adapter for Red Brick allows applications to access Red Brick data sources. The adapter converts application requests into native Red Brick statements and returns optimized answer sets to the requesting application.

Preparing the Red Brick Environment

The Adapter for Red Brick minimally requires the installation of the IBM Red Brick Client 32 ODBC software. After installation, ODBC must be configured to allow you to connect to a local or remote Red Brick database server.

Procedure: How to Set Up the Environment on Windows

On Windows, the Red Brick environment is set up during the installation of Red Brick.

Procedure: How to Set Up the Environment on UNIX

1. Specify the location of the Red Brick database you wish to access using the UNIX environment variable $RB_CONFIG. For example, to set the home directory for the Red Brick software to /rdbms/red61, specify:

   ```
   RB_CONFIG=/rdbms/red
   export RB_CONFIG
   ```

2. Define the $HOME/.odbc.ini file, which describes the Red Brick data sources. You can use the ODBCINI variable instead of $HOME/.odbc.ini.

Configuring the Adapter for Red Brick

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Red Brick Connection Attributes From the Web Console

Declare Connection Attributes Manually

Example:

Declaring Connection Attributes

In order to connect to a Red Brick database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Red Brick database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Red Brick Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Red Brick Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	Description
Data source	Red Brick data source name (DSN). There is no default data source name. You must enter a value.

Attribute	**Description**
User	Authorization by which the user is known to Red Brick.
Password	Password associated with the authorization ID. The password is stored in encrypted form.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

```
ENGINE [SQLRED] SET CONNECTION_ATTRIBUTES [connection]/userid,password
```

where:

`SQLRED`

Indicates the Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the name of the Red Brick Data Source Name (DSN) you wish to access. It must match an entry in the .odbc.ini.

`userid`

Is the primary authorization ID by which you are known to Red Brick.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Red Brick database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLRED SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Overriding the Default Connection

How to:

Change the Default Connection

Example:

Selecting the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLRED] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLRED`

Indicates the Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Red Brick database server named SAMPLENAME as the default Red Brick database server:

```
ENGINE SQLRED SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLRED] SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLRED`

Indicates the Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

`COMMIT`

Is converted to FIN.

Managing Red Brick Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Red Brick data types.

Creating Synonyms

How to:

Create a Synonym Using the Web Console

Create a Synonym Manually

Example:

Using CREATE SYNONYM

Reference:

Managing Synonyms

Access File Keywords

Synonyms define unique names (or aliases) for each Red Brick table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.

Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLRED [AT connection]
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLRED

Indicates the Data Adapter for Red Brick.

`AT connection`

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLRED AT DSN_A
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLRED ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the Red Brick table. The value assigned to this attribute can include the name of the owner (also known as schema) as follows: `TABLENAME=[owner.]table`
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`*
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following table lists how the server maps Red Brick data types.

Red Brick Data Type	Data Type		Remarks
	USAGE	ACTUAL	
CHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 2000
VARCHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 4000
DATE	HYYMD	HYYMD	Date/Time formats are comprised of several components. Not a true data value stored internally.
TIMESTAMP	HYYMDm	HYYMDm	Date/Time formats are comprised of several components. Not a true data value stored internally.
TINYINT	I6	I4	
SMALLINT	I6	I4	
REAL	D20.2	D8	
DOUBLE FLOAT	D20.2	D8	

Red Brick Data Type	Data Type		Remarks
	USAGE	**ACTUAL**	
SERIAL	P20.2	P8 P10	The SERIAL data type is a special case of the INTEGER data type. There can be only one SERIAL column in any table. SERIAL data types are not allowed in precomputed view definitions, including view columns, view table columns, and predicates. When defined, MUST BE specified as NOT NULL.
INTEGER	I11	I4	2^{31} = 2,147,483,648
NUMERIC, DECIMAL	D20.2	D8	Same as REAL or DOUBLE PRECISION

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Red Brick data type VARCHAR. By default, the server maps this data type as alphanumeric (A).

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLRED] SET VARCHAR {ON|OFF}
```

where:

`SQLRED`

Indicates the Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Red Brick data type VARCHAR as variable-length alphanumeric (A*n*V).

`OFF`

Maps the Red Brick data type VARCHAR as alphanumeric (A). OFF is the default value.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLRED] SET CONVERSION RESET
ENGINE [SQLRED] SET CONVERSION format RESET
ENGINE [SQLRED] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLRED] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLRED`

Indicates the Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	**MAX Precision**
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLRED SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLRED SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLRED SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLRED SET CONVERSION RESET
```

Customizing the Red Brick Environment

In this section:

Specifying a Timeout Limit

Obtaining the Number of Rows Updated or Deleted

How to:

Issue the TIMEOUT Command

Obtain the Number of Rows Updated or Deleted

The Adapter for Red Brick provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Red Brick.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE [SQLRED] SET TIMEOUT {nn|0}
```

where:

`SQLRED`

Indicates the Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

`nn`

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLRED] SET PASSRECS {ON|OFF}
```

where:

`SQLRED`

Indicates the Adapter for Red Brick. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLRED] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLRED`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

`setting`

Is the optimization setting. Valid values are as follows:

`OFF` instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

`ON` instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

`FOCUS` passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

`SQL` passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLRED set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLRED set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

How to:

Optimize Requests Containing Virtual Fields

Example:

Using IF-THEN_ELSE Optimization Without Aggregation

Using IF-THEN_ELSE Optimization With Aggregation

Using IF-THEN_ELSE Optimization With a Condition That is Always False

Reference:

SQL Limitations on Optimization of DEFINE Expressions

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLRED] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLRED`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLRED SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLRED SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLRED SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

CHAPTER 39

Using the Adapter for RMS

Topics:

The Adapter for RMS allows applications to access RMS data sources. The adapter converts application requests into native RMS statements and returns answer sets to the requesting application. If the metadata is configured for read/write capabilities, it can insert data from an application into the data source.

Preparing the RMS Environment

RMS is standard and always matches the OpenVMS release level. However, metadata creation is based on CDD Repositories, which require that a proper RDB environment be available. If RDB standard (of any version level) is installed and available globally to all users, then no additional steps are required for use.

If a DCL script (or symbol, if your RDB administrator has set up access in this way) must be run before you use RDB (as for RDB MultiVersion), then you must also run the script or symbol before you start the server. Note that this is only required for metadata creation and is not a requirement for RMS usage.

Configuring the Adapter for RMS

In the Adapter for RMS configuration screen, click *Configure* to add the Adapter for RMS.

Procedure: How to Configure the Adapter From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *Other DBMS* group folder, then expand the RMS adapter folder and click a connection. The Add RMS to Configuration pane opens.
3. No input parameters are required. Simply, click *Configure*. The adapter is added to the Configured adapters folder.

Managing RMS Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the RMS data types.

Three methods are available for metadata creation: the Create Synonym facility on the Web Console and the CREATE SYNONYM command (which describe data files to the server for OpenVMS with a Master and an Access File), and manual creation of a the Master File and Access File with a system editor.

The Master File describes the type of data file you are using, the structure, and the fields it contains. An Access File associates the Master File to the RMS file.

Creating Synonyms

How to:

Create a Synonym From the Web Console

Create a Synonym Manually

Example:

Using CREATE SYNONYM

Reference:

Managing Synonyms

CHECKNAMES for Special Characters and Reserved Words

Synonyms define unique names (or aliases) for each RMS table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Note: The Web Console Create Synonym facility for RMS and the CREATE SYNONYM command replace the RMS Automatic Description Generation facility (AUTORMS) in most situations. However, if you need to install and use AUTORMS, see the *iWay OpenVMS I53TOOLS* documentation on the iWay OpenVMS I53TOOLS CD-ROM.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The Select CDD Repository Records pane (Step 1 0f 2) opens.

4. Enter the following parameters:

Location of CDD definitions	If your repository is in a non-standard location (that is, one which is not declared by CDD$DEFAULT), you can supply the location here. If CDD$DEFAULT is declared, its current value appears in the input box, where you may leave it as is or change it.
Name	Enter a filter for retrieving a partial list of CDD references (for example, those starting with V%). You must supply filter values in *uppercase* characters, followed by a % sign. OpenVMS wildcards are not permitted.
Location of RMS data files	Specify a physical directory where RMS is located. This may be a logical that points to the full directory path or a directory path that uses a logical. OpenVMS wildcards are not allowed.
Name	Enter a filter for retrieving a partial list of RMS file names in the directory (for example, those starting with v%). You must supply filter values in *lowercase* characters, followed by a % sign. OpenVMS wildcards are not allowed.
Extension	Enter a filter for retrieving a partial list of RMS files based on the file extensions within the directory. The normal default for RMS files, dat%, is pre-populated in the input box and must *not* be changed if you wish to search for .dat files. You can change the .dat value by specifying a filter to access files that do not use RMS's normal default (that is, those starting with rms%). You must supply filter values in *lowercase* characters, followed by a % sign. OpenVMS wildcards are not allowed.

Click *Select CDD Repository* to complete this step.

5. The Select Synonym Candidates (Step 2 of 2) pane displays a list of CDD objects and data files from which metadata can be created. If selected, the filtered CDD repository records are displayed at the top of the pane.
6. Enter the following parameters, as required:

Parameter	Description
Select Application	Select an application directory in which the synonym will be stored. The default value is baseapp.
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank. **Tip:** While you can change individual names, entering a prefix or suffix enables you to make global changes that are particularly useful when many synonyms are being created.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

7. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

   When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.
8. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

9. Complete your selection:
    - To select all CDD records in the list, select the check box to the left of the *Default Synonym Name* column heading.
    - To select specific records, select the corresponding check boxes.
10. In the list of CDD objects and data files, use the pull-down menu on the right to match the CDD record with the physical file.
11. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.
12. Click *Create Synonym*. Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```
13. In the message window, you can click the *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.

Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '|', '&', ' (', ') ', ' <', '> ', '' ', ' =', ''''

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM [{app}/]{synonym} FOR {filename} DATABASE DBMS RMS
 [AT {CDD_location}] [DROP]
 [CHECKNAMES][UNIQUENAMES]
END
```

where:

`app`

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

`synonym`

Is an alias for the data source (maximum 64 characters).

`filename`

Is the physical file name, without the path or extension.

`DATABASE`

Is the full path name for the data file. Since data files are typically not in the current directory, you must supply the full path file name and extension. Logical names are permitted.

`CDD_location`

Is the full path name to the CDD (Common Data Dictionary) repository location. If not supplied, the system default or `CDD$DEFAULT` (if declared) is used.

`DROP`

Deletes an existing synonym, if one exists. An error is produced if DROP is not supplied and there is a pre-existing synonym. This option has no effect if there is no pre-existing synonym.

`CHECKNAMES`

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

`UNIQUENAMES`

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

This example generates the synonym from a CDD (Common Data Dictionary) repository, whose location is TSCQDATA:[TSCQDATA.CDDPLUS].

```
CREATE SYNONYM baseapp/VSAM002 FOR VSAM002 DATABASE
  TSCQDATA:[TSCQDATA.RMS]vsam002.dat DBMS RMS
  AT TSCQDATA:[TSCQDATA.CDDPLUS]DROP
END
```

Manually Describing RMS Files

In this section:

These topics outline the procedures necessary for manually describing RMS files.

File Attributes

In this section:

File attributes are FILENAME and SUFFIX, which name the file and describe the file type. For example:

```
FILENAME=LIBRARY1, SUFFIX=RMS,$
```

FILENAME

The FILENAME attribute is optional. It is recommended that you include the file name for documentation purposes. You can specify any name for this attribute, but you should use the same name that you give the Master File.

The syntax is

```
FILE[NAME] = name
```

where:

name

Is any name of 1 to 8 characters.

SUFFIX

The SUFFIX attribute describes the type of data file it will read.

The syntax is:

```
[FILE]SUFFIX = type
```

where:

type

Is a suffix listed in the following table:

Type of File	Suffix	Physical Access Method
Keyed (Indexed) RMS file	`RMS`	`Access File`
Fixed-format sequential RMS file	`FIX`	`FILEDEF`
Comma-delimited sequential RMS file	`COM`	`FILEDEF`

The description of data and relationships between fields within a file is the same for keyed (indexed) RMS files, FIX (fixed-format sequential) files, and COM (comma-delimited) files, except for the following differences:

- Keyed (indexed) RMS uses the keyed RMS File System to access data using information in the Access File declarations. You can override the Access File declaration with a FILEDEF. For more information, see *Assigning a Keyed RMS File to a Master File* on page 39-41. The others use sequential access using a FILEDEF to identify and access the file. The coding of FILEDEFs may be done locally within a procedure (before the access request), or within the EDASPROF.PRF if commonly accessed by multiple users.
- Index related declarations do not apply to fixed-sequential or comma-delimited files.

Segment Attributes

In this section:

SEGNAME

PARENT

SEGTYPE

A Master File can be divided into segments, which are groups of fields that are related to one another. It is not always necessary to divide your file into segments.

Segment declarations identify and describe each segment in a file. They name the segments and indicate the relative positions in the file structure. In files with multiple record types, each record type will be described as a separate segment. Files with mixed singly- and multiply-occurring fields must also be described with separate segments defined for each type of occurrence. The attributes used to describe segments of different record types and multiply-occurring fields are described in *Describing Multiple Record Types* on page 39-25 and in *Describing Embedded Repeating Data* on page 39-31.

The following is an example of a segment declaration:

```
SEGNAME=BOOKINFO, PARENT=PUBINFO, SEGTYPE=S0,$
```

SEGNAME

Each segment declaration starts with the SEGNAME (or SEGMENT) attribute, which names the segment.

The syntax is

```
{SEGNAME|SEGMENT}= name
```

where:

name

Is a unique name of 1 to 8 characters.

PARENT

Files with more than one segment are defined as multi-segment structures.

Segments in a multi-segment structure have a parent/child relationship. Each segment, except the top or “root” segment, is the descendant of another segment called the “parent.” The PARENT attribute is used to identify a segment's parent segment. If no PARENT attribute is specified in the Master File, the default parent segment is the immediately preceding segment, except for the top segment, which has no parent.

The syntax is

```
PARENT = name
```

where:

name

Specifies a SEGNAME in the file.

For example:

```
...PARENT=PUBINFO...
```

SEGTYPE

The SEGTYPE attribute for RMS files is specified as S0.

The required syntax is:

```
SEGTYPE=S0
```

Field Attributes

In this section:

FIELDNAME

ALIAS

USAGE

ACTUAL

Field attributes describe the actual fields in each segment. Each field declaration consists of at least four attributes. They are:

```
FIELDNAME    ALIAS    USAGE    ACTUAL
```

For example:

```
FIELDNAME=PUBNO ,ALIAS=PN ,USAGE=A10 ,ACTUAL=A10 ,$
```

There are also other optional attributes that can be used, such as DESCRIPTION, TITLE, and ACCEPT. DESCRIPTION enables you to include a description of the field. TITLE is the default report column title other than the field name. ACCEPT assigns a list or range of acceptable values to a field. ACCEPT is also used for RECTYPE values. These optional attributes are used by FOCUS, WebFOCUS, and various other client products. For further information about optional attributes, see the appropriate manuals.

FIELDNAME

Field names are unique names of 1 to 66 characters with the exception of indexes, which are limited to12. The field name appears as a default column heading when you name the field in a report request. Field names may consist of any alphanumeric characters, but the first character must be a letter from A to Z. Field names may include embedded blanks, but it is not recommended, and you will need to enclose such a field name in single quotation marks (') if you reference the complete name (including the blank) in a request. Avoid using special characters (+ - $ * /() ' ; . , = and " > <), since they may cause confusion if the field is used in calculations.

The syntax is

```
FIELD[NAME] = name
```

where:

name

Meets the criteria described above.

ALIAS

Aliases are optional field names. Each field can have an alias to be used interchangeably with the field name. The length and format rules for field names apply to aliases. Aliases are not used as column titles.

The syntax is

```
ALIAS = alias
```

where:

alias

Is a name of 1 to 66 alphanumeric characters meeting the same criteria as for field names.

If you omit the alias, you must indicate its absence with either the following entry in the field declaration

```
ALIAS=,
```

or by holding its place with a comma delimiter (,):

```
FIELDNAME=PUBNO , ,USAGE=A10 ,ACTUAL=A10 ,$
```

USAGE

The USAGE attribute describes the way you want to use the field and display its values on reports. This attribute includes the data field type, display length, and any edit options that are to be applied when the field values are printed.

The syntax is

```
USAGE = usage
```

where:

`usage`

Describes the field in three parts: field type, field display length, and edit options.

The values that you specify for type and display length determine the number of print positions allocated for the field in any display or report. Edit options only affect printed or displayed fields; they are not active for extract files or other non-display retrievals.

The following table shows permissible USAGE field types and display lengths:

USAGE	**Length**	**Description**
`A`	1–9,095	Alphanumeric text
`D`	1–19	Decimal, double-precision numbers
`F`	1–9	Decimal, single-precision numbers
`I`	1–11	Integer values (no decimal places)
`P`	1–17	Packed decimal numbers
`YYMD`	10	Displayed as YYMD
`D,W,M,Q or Y`	Date	Date display

The following table shows edit options that may be applied to various A, D, F, I, or P usages:

Edit Option	**Meaning**	**Effect**
`%`	Percent sign	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	Scientific notation Displays only significant digits.
`L`	Leading zeroes	Adds leading zeroes.
`M`	Floating $ (for US code page)	Places a floating dollar sign $ to the left of the highest significant digit. **Note:** The currency symbol displayed depends on the code page used.
`MDY`	Month-Day-Year	Displays alphanumeric or integer data as a date in the form month/day/year.
`N`	Fixed $ (for US code page)	Places a dollar sign $ to the left of the field. The symbol displays only on the first detail line of each page. **Note:** The currency symbol displayed depends on the code page used.
`R`	Credit (CR) negative	Places CR after negative numbers.
`S`	Zero suppress	Zero suppress If the data value is zero, prints a blank in its place.

Note:

- Edit options can be specified in any order.
- The total length of the USAGE specification, including all edit options, may not exceed eight characters.
- Options M and N (floating and fixed dollar sign) both automatically imply option C (comma edit).
- Format type D (double-precision decimal number) implies option C.

The following table shows the various USAGE formats as they would be specified in a Master File. The USAGE formats contain type and length information as well as various edit options. The Value column shows the actual value as it would be read from the external file, and the Display column shows how the number would be displayed on a report.

USAGE	**Value**	**Display**
`I7C`	`47693`	`47,693`
`D10.2S`	`0.00`	
`P8.0CR`	`-4719`	`4,719 CR`
`F8.2MC`	`28148.00`	`$28,148.00`
`I5B`	`-341`	`(341)`
`D7M`	`8741`	`$8,741`
`D7N`	`8741`	`$8,741`
`P6L`	`21`	`000021`
`D12.5`	`E1234.56`	`0.123456D+04`
`I6MDY`	`20675`	`02/06/75`
`A6YMD`	`750601`	`75/06/01`
`A10`	`HELLO`	`HELLO`

ACTUAL

The ACTUAL attribute describes the type and length of your data as it actually exists in a file. The source of this information is the existing description of the file.

The syntax is

```
ACTUAL = actual
```

where:

actual

Is any of the format types listed in the following table:

ACTUAL Type	Definition
A*n*	Alphanumeric characters A to Z, 0 to 9, and other displayable ASCII characters, where *n* equals 1 to 4,095.
D8	Double-precision floating-point numbers stored internally in eight bytes (D_Floating).
F4	Single-precision floating-point numbers stored internally in four bytes (F_Floating).
I*n*	Binary integers: `I1` Single-byte signed, binary integer (signed byte) `I2` 2-byte signed, binary integer (signed word) `I4` 4-byte signed, binary integer (signed longword) `I8` 8-byte signed, binary integer (signed quadword) If an integer field contains an assumed decimal point (that is, if it is a scaled integer), represent the field as I*m*.*n*, where *m* is the total number of storage bytes, and *n* is the number of decimal places. Therefore, `I4.1` means a 4-byte number with one decimal place.
P*n*	Packed decimal format (packed numeric string) where *n* is the number of bytes (1 to 16), each of which contains two digits, except for the last byte, which contains a digit and the sign. For example, P6 means 11 digits plus a sign, packed two digits to the byte for the total of six bytes of storage.

ACTUAL Type	**Definition**

`Z`*`n`*	Zoned decimal format (numeric string) where *n* is the number of digits (1 to 31), each of which takes one byte of storage. The last byte contains a digit and the sign.
	
	There are several standards for zoned data. For Read purposes, only right overpunched standards are supported and can be determined on Read since they are unique.
	
	The specific format to use when writing data must be known for read/write purposes. The default is ASCII right overpunch. To change the default, you must edit the EDACONF [.BIN]EDAENV.COM file and add the following logical:
	
	`DEFINE /NOLOG IBI_ZONED_OUT_TYPE {1|2}`
	
	where:
	
	`1`
	Is the ASCII right overpunched standard (default).
	
	`2`
	Is the EBCDIC right overpunched standard.
	
	Zoned right separate numeric or zoned left overpunched numeric formats are not supported.

`DATE`	DATE indicates that the field is an OpenVMS 64-bit datetime stamp. This usage also requires an A4 filler field immediately following in the Master File.

The following conversions from ACTUAL format to USAGE (display) format are permitted:

ACTUAL	**USAGE**
`A`	`A, D, F, I, P, date format`
`D`	`D`
`DATE`	`date format`
`F`	`F`
`I`	`I, date format`
`P`	`P, date format`
`Z`	`D, F, I, P`

The following table shows the USAGE and ACTUAL formats for COBOL, FORTRAN, PL1, and Assembler field descriptions.

COBOL USAGE FORMAT	BYTES OF COBOL PICTURE	INTERNAL STORAGE	ACTUAL FORMAT	USAGE FORMAT
`DISPLAY`	`X(4)`	`4`	`A4`	`A4`
`DISPLAY`	`S99`	`2`	`Z2`	`P3`
`DISPLAY`	`9(5)V9`	`6`	`Z6.1`	`P8.1`
`DISPLAY`	`99`	`2`	`A2`	`A2`
`COMP`	`S9`	`4`	`I2`	`I1`
`COMP`	`S9(4)`	`4`	`I2`	`I4`
`COMP*`	`S9(5)`	`4`	`I4`	`I5`
`COMP`	`S9(9)`	`4`	`I4`	`I9`
`COMP-1**`	`—`	`4`	`F4`	`F6`
`COMP-2***`	`—`	`8`	`D8`	`D15`
`COMP-3`	`9`	`8`	`P1`	`P1`
`COMP-3`	`S9V99`	`8`	`P2`	`P5.2`
`COMP-3`	`9(4)V9(3)`	`8`	`P4`	`P8.3`
`FIXED BINARY(7)` `(COMP-4)`	`B or XL1`	`8`	`I4`	`I7`

* Equivalent to INTEGER in FORTRAN, FIXED BINARY(31) in PL/1, and F in Assembler.

** Equivalent to REAL in FORTRAN, FLOAT(6) in PL/1, and E in Assembler.

*** Equivalent to DOUBLE PRECISION or REAL*8 in FORTRAN, FLOAT(16) in PL/1, and D in Assembler.

Note:

- The USAGE lengths shown are minimum values. They may be larger if desired. Additional edit options may also be added.
- In USAGE formats, an extra character position is required for the minus sign if negative values are expected.
- PICTURE clauses are not permitted for internal floating-point items.
- USAGE length should allow for maximum possible number of digits.
- In USAGE formats, an extra character position is required for the decimal point.

Describing Indexed Files (SUFFIX=RMS)

RMS keyed (indexed) files may be described using a SUFFIX of RMS, an attribute called GROUP, and a FIELDTYPE of I to designate a key. The GROUP attribute designates a key that is comprised of one or more contiguous fields. Additional keys may be described using either the GROUP or FIELD attribute, and a FIELDTYPE of I.

Segment Name for Indexed Files

The segment name (SEGNAME value) of the first segment in an indexed file (SUFFIX=RMS) must be ROOT. The remaining segments can have any valid segment name. The only exception to this rule is for unrelated record types where the first segment name value must be DUMMY.

Describing Keys

The primary key is defined by the GROUP attribute and an alias value of KEY in the Master File. If there is a secondary key consisting of more than one field, it must also be described by the GROUP attribute and a numbered key. Otherwise, the secondary key can be indicated by using a FIELDTYPE of I in the field description.

The primary key of an RMS file is defined using the GROUP attribute, consisting of one or more fields. A file might only have one keyfield, but it must still be described with the GROUP declaration. The GROUP must contain ALIAS=KEY. Coding KEY without ALIAS= is not sufficient. The GROUP declaration has the following syntax

```
GROUP=keyname, ALIAS=KEY, USAGE=usage, ACTUAL=actual,$
```

where:

keyname

Is a name of 1 to 66 characters.

The secondary keys of an RMS file are indicated by using ALIAS=KEY(n) and FIELDTYPE=I in the GROUP or FIELD declaration for the key. For example

```
GROUP=keyname, ALIAS=KEYn, USAGE=usage, ACTUAL=actual, FIELDTYPE=I,$
```

where:

keyname

Is a name of 1 to 48 characters.

`KEYn`

Indicates the alternate key.

The first alternate key is designated as KEY1, the second as KEY2, and so on.

Alternatively, if the secondary key is made up of only one field, the following syntax can be used:

```
FIELD=keyname, ALIAS=KEYn, USAGE=usage, ACTUAL=actual, FIELDTYPE=I,$
```

GROUP (for Contiguous Keys)

Contiguous keys consist of adjacent fields.

The GROUP attribute is used to define a contiguous primary key or a secondary key that is comprised of more than one field. The syntax is

```
GROUP=groupname,ALIAS=KEY[n],USAGE=usage,ACTUAL=actual,FIELDTYPE=I,$
  FIELD=fieldname,ALIAS=alias,USAGE=usage,ACTUAL=actual,$
  .
  .
  .
  FIELD=fieldname,ALIAS=alias,USAGE=usage,ACTUAL=actual,$
```

where:

groupname

Is a name of 1 to 48 characters.

`KEY[n]`

Indicates a contiguous key. Use only KEY to specify a primary key. Use KEY[*n*] to specify a secondary key, where *n* is a number from 1 to 254 that indicates the key reference number.

`FIELDTYPE=I`

Makes the key accessible for reporting and indicates that the key can be used for direct retrieval. Do not use FIELDTYPE=I for a primary key.

usage

Is the data type and length designation.

actual

Is the data type and length designation.

For example, consider the contiguous key in the first part of the following Master File:

```
FILENAME=MANUALS,SUFFIX=RMS,$
  SEGMENT=ROOT,SEGTYPE=S0,$
  GROUP=MDOCNUM        ,ALIAS=KEY   ,USAGE=A9       ,ACTUAL=A9       ,$
    FIELDNAME=DOCNUM   ,ALIAS=DN    ,USAGE=A5       ,ACTUAL=A5       ,$
    FIELDNAME=CODE     ,ALIAS=CD    ,USAGE=I6       ,ACTUAL=I4       ,$
    FIELDNAME=MRELEASE ,ALIAS=MR    ,USAGE=A7       ,ACTUAL=A7       ,$
    FIELDNAME=MPAGES   ,ALIAS=MP    ,USAGE=I5       ,ACTUAL=I2       ,$
         .
         .
         .
```

GROUP (for Discontiguous Keys)

Discontiguous keys consist of non-adjacent fields. If the GROUP attribute is used with discontiguous keys, the syntax is

```
GROUP=groupname,ALIAS=DKEY[n],USAGE=usage,ACTUAL=actual,FIELDTYPE=I,$
   FIELD=      ,ALIAS=alias  ,USAGE=usage,ACTUAL=actual,$
   .
   .
   .
   FIELD=      ,ALIAS=alias  ,USAGE=usage,ACTUAL=actual,$
```

where:

`DKEY[n]`

Indicates that this GROUP is the key layout for a discontiguous key. The GROUP declaration must explicitly specify ALIAS=DKEY. Use only DKEY to specify a primary key. Use DKEY[*n*] to specify a secondary key, where *n* is a number from 1 to 254 that indicates the key reference number.

`alias`

Is the name of the base field to which the discontiguous key field corresponds. Note that the field name for the key field must be blank.

`usage`

Is the data type and length designation.

`actual`

Is the data type and length designation.

`FIELDTYPE=I`

Makes the key accessible for reporting and indicates that the key can be used for direct retrieval. Do not use FIELDTYPE=1 for a primary key.

The fields that comprise the discontiguous key are described in the order in which they appear in the record. They are then re-described using the GROUP attribute. The order within the GROUP is determined by their order of significance within the discontiguous key.

For example, consider the discontiguous keys in the first part of the following Master File:

```
FILENAME=MANUALS,SUFFIX=RMS,$
   SEGMENT=ROOT,SEGTYPE=S0,$
      FIELDNAME=DOCNUM    ,ALIAS=          ,USAGE=A5   ,ACTUAL=A5       ,$
      FIELDNAME=CODE      ,ALIAS=CD        ,USAGE=I6   ,ACTUAL=I4       ,$
      FIELDNAME=MRELEASE  ,ALIAS=          ,USAGE=A7   ,ACTUAL=A7       ,$
      FIELDNAME=MPAGES    ,ALIAS=          ,USAGE=I5   ,ACTUAL=I2       ,$
   GROUP=MANUALS_KEY      ,ALIAS=DKEY      ,USAGE=A12  ,ACTUAL=A12      ,$
      FIELDNAME=          ,ALIAS=DOCNUM    ,USAGE=A5   ,ACTUAL=A5       ,$
      FIELDNAME=          ,ALIAS=MRELEASE  ,USAGE=A7   ,ACTUAL=A7       ,$
      .
      .
      .
```

Note that within each segment, you can only have one instance of a key reference number. For example, consider the following: KEY5 and DKEY4 can be in the same segment because they do not reference the same key, but KEY4 and DKEY4 cannot be in the same segment because they reference the same key.

A GROUP of DKEY can occur only at the end of the segment, following all of the "real" field definitions since DKEY contains references to "real" fields as base fields.

Note: OCCURS segments cannot contain any key definitions. Non-OCCURS segments must contain either KEY or DKEY definitions.

USAGE and ACTUAL

For multi-field GROUPs, USAGE and ACTUAL formats are always alphanumeric. The ACTUAL attribute is A*n*, where *n* is the sum of the actual lengths of the subordinate fields. The USAGE format is the sum of the internal storage lengths of the subordinate fields.

- Fields of USAGE I have an internal storage length of 4.
- Fields of USAGE F have an internal storage length of 4.
- Fields of USAGE P have an internal storage length of either 8 or 16.
- Fields of USAGE D have an internal storage length of 8.
- Alphanumeric fields have an internal storage length equal to the number of characters they contain as their field length. For example, fields of type A*n* have an internal storage length of *n*.
- Natural date formats that are treated as integers have an internal storage length of 4.

For example, consider the discontiguous keys in a part of the following Master File:

```
.
.
.
FIELDNAME=FIELD1    ,ALIAS=           ,USAGE=P6    ,ACTUAL=P2       ,$
FIELDNAME=FIELD2    ,ALIAS=           ,USAGE=I9    ,ACTUAL=I4       ,$
FIELDNAME=FIELD3    ,ALIAS=           ,USAGE=A2    ,ACTUAL=A2       ,$
GROUP=ALTERNATE     ,ALIAS=DKEY       ,USAGE=A10   ,ACTUAL=A4       ,$
FIELDNAME=          ,ALIAS=FIELD3     ,USAGE=A2    ,ACTUAL=A2       ,$
FIELDNAME=          ,ALIAS=FIELD1     ,USAGE=P6    ,ACTUAL=P2       ,$
.
.
.
```

The GROUP declaration USAGE attribute tells how many positions to use for the group key. If this length is wrong, the group key will not be used correctly.

In this example, the lengths of the ACTUAL attributes for subordinate fields FIELD3 and FIELD1 total 4, which is the length of the ACTUAL attribute of the GROUP key. The lengths of the USAGE attributes for the subordinate fields total 8. However, the length of the GROUP key USAGE attribute is found by adding their internal storage lengths as specified by the field types: 2 for USAGE=A2 and 8 for USAGE=P6, for a total of 10.

Single-Field Secondary Keys

Single-field secondary keys must be described as fields with FIELDTYPE=I. The ALIAS must be the key reference number as described in the RMS File Description Language (FDL), that is, KEY*n* or DKEY*n*, where *n* is a number from 1 to 254. Secondary keys can be described as GROUPs if they consist of portions with dissimilar formats. For example,

```
FILENAME=CUST,SUFFIX=RMS,$
SEGNAME=ROOT,SEGTYPE=S0,$
  GROUP=G                ,ALIAS=KEY     ,USAGE=A10   ,ACTUAL=A10,$
    FIELDNAME=SSN        ,ALIAS=SSN     ,USAGE=A10   ,ACTUAL=A10,$
    FIELDNAME=FNAME      ,ALIAS=KEY1    ,USAGE=A10   ,ACTUAL=A10,FIELDTYPE=I,$
    FIELDNAME=LNAME      ,ALIAS=KEY2    ,USAGE=A10   ,ACTUAL=A10,FIELDTYPE=I,$
```

Here, SSN is a primary key and FNAME and LNAME are secondary keys.

If you are not sure of the secondary keys associated with a given file, you can use the OpenVMS ANALYSE facility. Refer to the OpenVMS documentation for ANALYSE/RMS. The DIRECTORY command with the /FULL qualifier will also show how many keys the file has.

Describing Complex RMS Keyed

These topics discuss various ways of describing complex RMS keyed files.

Describing Multiple Record Types

Files may have records that must be deciphered according to a record type indicator in the record itself. Describing these files involves two things. First, each different record type will need its own segment. Second, the relationship between the different record types (that is, between the different segments) will need to be determined and expressed using the PARENT attribute for each segment.

Using RECTYPE

The syntax for defining a RECTYPE field is

```
FIELDNAME=RECTYPE,ALIAS=alias,USAGE=usage,ACTUAL= actual
[,ACCEPT=list/range],$
```

where:

`RECTYPE`

Is the required field name.

`alias`

Is the primary RECTYPE identifier. If there is an ACCEPT list or range, this value is any valid alias name.

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 an item in 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=A, 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 either value 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.

For example:

```
FIELDNAME=RECTYPE,ALIAS=100,USAGE=P3,ACTUAL=P2,
ACCEPT=70 TO 100,$
```

When using an ACCEPT list or range, also called generalized RECTYPE, the RECTYPE field name is not unique across segments, therefore it must not be used in a WHERE clause of an SQL request. It is the responsibility of the client application to determine if the row for the RECTYPE is required. If the client application cannot do this, you must code individual segments for each RECTYPE.

To illustrate the use of the generalized RECTYPE capability in RMS file descriptions, consider the following record layouts in the DOC file. 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. For example, 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 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=DOC, SUFFIX=RMS,$
SEGNAME=ROOT, SEGTYPE=S0,$
 GROUP=DOCNUM,  ALIAS=KEY,          A5,  A5, $
  FIELD=DOCID,     ALIAS=,          A5,  A5, $
 FIELD=FILLER,     ALIAS=,          A5,  A5, $
 FIELD=RECTYPE,    ALIAS=,          A3,  A3, $
 FIELD=TITLE,      ALIAS=,          A18, A18,$
SEGNAME=MANUALS, PARENT=ROOT, SEGTYPE=S0,    $
 GROUP=MDOCNUM, ALIAS=KEY,          A10, A10,$
  FIELD=MDOCID,    ALIAS=,          A5,  A5, $
  FIELD=MDATE,     ALIAS=,          A5,  A5, $
 FIELD=RECTYPE,    ALIAS=M,         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=S0,    $
 GROUP=CRSEDOC, ALIAS=KEY,          A10, A10,$
  FIELD=CDOCID,    ALIAS=,          A5,  A5, $
  FIELD=CDATE,     ALIAS=,          A5,  A5, $
 FIELD=RECTYPE,    ALIAS=C,         A3,  A3, $
 FIELD=COURSENUM,  ALIAS=,          A4,  A4, $
 FIELD=LEVEL,      ALIAS=,          A2,  A2, $
 FIELD=CPAGES,     ALIAS=,          I5,  A5, $
 FIELD=FILLER,     ALIAS=,          A7,  A7, $
```

Describing Related Record Types

Consider the LIBRARY file that contains three types of records, related by a combination of the key and the RECTYPE attribute. The ROOT records have a key that consists of the publisher's number. The BOOKINFO segment has a key that consists of that same publisher's number, plus a hard or soft-cover indicator. The SERIANO key consists of the first two elements, plus a record type.

In the sample file, the repetition of the publisher's number interrelates the three types of records. The Master File for this file would look like the following:

```
FILENAME=LIBRARY6,SUFFIX=RMS,$
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,SEGTYPE=S0,PARENT=ROOT,$
 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,SEGTYPE=S0,PARENT=BOOKINFO,$
 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,SEGTYPE=S0,PARENT=ROOT,OCCURS=VARIABLE,$
  FIELDNAME=PLOTLINE     ,ALIAS=PLOTL ,USAGE=A10     ,ACTUAL=A10    ,$
```

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 and the processing continues. For greater speed retrieval, you could add search criteria based on the BINDING field, which is also part of the key.

Describing Unrelated Record Types

A file may contain records that are not related to each other. Records with varying RECTYPEs exist independently of each other in a file, and the sequence of records in the file may be random.

Consider our LIBRARY file. Suppose that the file 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 record types cannot be described in a parent/child relationship.

The records simply look like the following:

```
BOOK     MAGAZINE     NEWSPAPER
```

To describe a file with unrelated records, you must make the record types descendants of a root segment named DUMMY.

The following rules apply to the DUMMY segment:

- The root (top) segment name must be DUMMY.
- It can have only one field, with an empty FIELDNAME and ALIAS.
- Both its USAGE and ACTUAL attributes must be A1.

All of the other segments must be descendants of the DUMMY segment.

The Master File for this file would look like the following:

```
FILENAME=LIBRARY3, SUFFIX=FIX,$
SEGMENT=DUMMY,SEGTYPE=S0,$
   FIELDNAME=             ,ALIAS=       ,USAGE=A1      ,ACTUAL=A1      ,$
SEGMENT=BOOK,SEGTYPE=S0,PARENT=DUMMY,$
   FIELDNAME=RECTYPE      ,ALIAS=B      ,USAGE=A1      ,ACTUAL=A1      ,$
   GROUP=PUBNUM           ,ALIAS=KEY    ,USAGE=AI0     ,ACTUAL=A10     ,$
   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,SEGTYPE=S0,PARENT=DUMMY,$
   FIELDNAME=RECTYPE      ,ALIAS=M      ,USAGE=A1      ,ACTUAL=A1      ,$
   GROUP=PERNUM           ,ALIAS=KEY    ,USAGE=A10     ,ACTUAL=A10     ,$
   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=YYMD    ,ACTUAL=DATE    ,$
   FIELDNAME=             ,ALIAS=FILLER ,USAGE=A4      ,ACTUAL=A4      ,$
SEGMENT=NEWSPAP,SEGTYPE=S0,PARENT=DUMMY,$
   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=I4      ,$
   FIELDNAME=             ,ALIAS=FILLER ,USAGE=A4      ,ACTUAL=A4      ,$
   GROUP=PAPVI            ,ALIAS=KEY    ,USAGE=A4      ,ACTUAL=A4      ,$
   FIELDNAME=NVOL_NO      ,ALIAS=NV     ,USAGE=I2      ,ACTUAL=I2      ,$
   FIELDNAME=NISSUE       ,ALIAS=NI     ,USAGE=I2      ,ACTUAL=I2      ,$
```

Describing Embedded Repeating Data

Some records may contain embedded repeating data. Consider the following record layout:

```
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, depending 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 may be different, 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 first segment, called the root segment. Fields C1 and C2, which occur multiple times in relation to A and B, are placed in a descendant segment. You use an additional segment attribute, the OCCURS attribute, to specify that this segment is a multiply occurring segment.

Repeating fields or groups of fields described by the OCCURS attribute are not supported for free-format (comma-delimited) sequential files.

OCCURS

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 another, subordinate segment. The OCCURS attribute appears in the declaration for the subordinate segment.

The syntax is

```
OCCURS = {n | fieldname | VARIABLE},$
```

where:

n

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

fieldname

Names a data field in the parent segment, that is, a counter specifying 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).

When different types of records are combined in one file, 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 structure. This restriction is necessary to ensure that data in the record is interpreted correctly.

Example: Using the OCCURS Attribute

You place the OCCURS attribute in the segment declaration after the PARENT attribute. Consider the following record layout:

```
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 file, you place fields A and B in the root segment, and fields C1 and C2 in the descendant segment.

You describe this file with the following Master File:

```
FILENAME=EXAMPLE1,SUFFIX=RMS,$
 GROUP=ONE    ,ALIAS=KEY    ,USAGE=A2    ,ACTUAL=A2    ,FIELDTYPE=I    ,$
  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    ,$
```

Note: OCCURS is not supported for Write Access. However, if the fields occur a specific number of times, an alternate Master File can be built with the fields described that number of times (for example, PAYMENT_1, PAYMENT_2 if Payment occurs two times). Using the alternate Master, specific instances of the OCCURS can be referenced. If the OCCURS segments will not be referenced during the write, the alternate Master is not needed. However, the message (FOC1305) RMS Duplicate Record will display.

Example: Describing Parallel and Nested Sets of OCCURS Segments

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 Master File.

Sets of repeating fields can be divided into two basic types: parallel and nested. Parallel sets of repeating fields are unrelated (that is, they have no parent/child or logical relationship). Consider the following record layout:

```
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 is unrelated to 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 Master File as children of the same parent, the segment that contains fields A1 and A2. The following Master File illustrates this relationship:

```
FILENAME=EXAMPLE1, SUFFIX=RMS,$
 GROUP=ONE    ,ALIAS=KEY    ,USAGE=A2    ,ACTUAL=A2    ,FIELDTYPE=I    ,$
  FIELDNAME=A1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
  FIELDNAME=A2,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
 SEGNAME=TWO, SEGTYPE=S0, PARENT=ONE,     OCCURS=2,     $
  FIELDNAME=B1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
  FIELDNAME=B2,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
 SEGNAME=THREE, SEGTYPE=S0, PARENT=ONE,   OCCURS=3,     $
  FIELDNAME=C1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
  FIELDNAME=C2,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
```

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 occurs after fields B1 and B2 occur once. Field C1 occurs a varying number of times, as recorded by a counter field, B2. There will not be a set of occurrences of C1 unless C1 is preceded by an occurrence of fields B1 and B2. Fields B1, B2, and C1 are a nested set of repeating fields. The following Master File illustrates this relationship:

```
FILENAME=EXAMPLE2, SUFFIX=RMS,$
 GROUP=ONE    ,ALIAS=KEY    ,USAGE=A2    ,ACTUAL=A2    ,FIELDTYPE=I    ,$
  FIELDNAME=A1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,       $
  FIELDNAME=A2,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,       $
 SEGNAME=TWO,       SEGTYPE=S0, PARENT=ONE, OCCURS=2,       $
  FIELDNAME=B1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,       $
  FIELDNAME=B2,    ALIAS= ,    USAGE=I6,    ACTUAL=I4,       $
 SEGNAME=THREE,     SEGTYPE=S0, PARENT=TWO, OCCURS=B2,      $
  FIELDNAME=C1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,       $
```

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.

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
```

You describe this with the Master File that follows. The PARENT attribute is used to describe the logical relationship of the segments. Note that the assignment of the PARENT attribute shows you how the occurrences are nested:

```
FILENAME=EXAMPLE3,SUFFIX=RMS,$
 GROUP=ONE    ,ALIAS=KEY   ,USAGE=A2    ,ACTUAL=A2    ,FIELDTYPE=I   ,$
  FIELDNAME=A1    ,ALIAS=    ,USAGE=A2    ,ACTUAL=A2    ,    $
  FIELDNAME=A2    ,ALIAS=    ,USAGE=I4    ,ACTUAL=I1    ,    $
 SEGNAME=TWO,PARENT=ONE,OCCURS=2,SEGTYPE=S0,                 $
  FIELDNAME=B1    ,ALIAS=    ,USAGE=A15   ,ACTUAL=A15   ,    $
  FIELDNAME=B2    ,ALIAS=    ,USAGE=I4    ,ACTUAL=I1    ,    $
 SEGNAME=THREE,PARENT=TWO,OCCURS=B2,SEGTYPE=S0,              $
  FIELDNAME=C1    ,ALIAS=    ,USAGE=A25   ,ACTUAL=A25   ,    $
 SEGNAME=FOUR,PARENT=ONE,OCCURS=A2,SEGTYPE=S0,               $
  FIELDNAME=D1    ,ALIAS=    ,USAGE=A15   ,ACTUAL=A15   ,    $
 SEGNAME=FIVE,PARENT=ONE,OCCURS=VARIABLE,SEGTYPE=S0,         $
  FIELDNAME=E1    ,ALIAS=    ,USAGE=A5    ,ACTUAL=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 B2, which is a counter. The value of B2 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 for the 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 example, the value of A2 is 2.
- The number of times Segment FIVE occurs is VARIABLE. The rest of the fields in the record (all those to the right of the first occurrence of E1) as occurrences of E1. To ensure that data in the record is interpreted correctly, 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. There can only be one segment defined as OCCURS=VARIABLE for each parent.

POSITION

The POSITION attribute is an optional attribute. It is used to describe a record in which non-repeating data follows embedded repeating data.

You describe the file as a multi-segment structure, made up of a parent segment and at least one child segment that contains the embedded repeating data. The parent segment is made up of whatever singly occurring fields are in the record, as well as an alphanumeric field (or fields) that indicates where the embedded repeating data appear in the record. The alphanumeric field is a placeholder that is the exact length of the combined embedded repeating data. For example, if you have four occurrences of an 8-character field, the length of the placeholder in the parent segment will be 32 characters.

The POSITION attribute is described in the child segment. It gives the name of the placeholder field in the parent segment.

The syntax is

```
POSITION = fieldname
```

where:

fieldname

Is the name of the field in the parent segment that defines the starting position of the multiply occurring fields.

Consider the following record layout:

```
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 that occupies 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 place holder field in the parent segment.

The Master File for this file would look like this:

```
FILENAME=EXAMPLE3,SUFFIX=RMS,$
 GROUP=ONE   ,ALIAS=KEY   ,USAGE=A2   ,ACTUAL=A2   ,FIELDTYPE=I   ,$
  FIELDNAME=A1     ,ALIAS=   ,USAGE=A2   ,ACTUAL=A2    ,$
  FIELDNAME=QFIL   ,ALIAS=   ,USAGE=A1   ,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    ,$
```

If the total length of the multiple occurrences of the field(s) is greater than 256, you can use a filler field after the place holder field to make up the remaining length. This is because the format of a field cannot exceed 256 bytes.

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

When a segment is coded with ...OCCURS=*n*, POSITION=*fieldname*, ..., all segments that follow using OCCURS=*n* or OCCURS=*fieldname*, must use the POSITION attribute to correctly map data.

ORDER Field

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

You can add a special ORDER field to your Master File to identify the individual occurrences of data uniquely within their parent segment. This is typically done to the screen by the occurrence number versus a field value (for example, the month is 12). The sequence number of each instance of the segment is automatically defined as a virtual field, and does not actually exist within the file.

The following syntax rules apply to the ORDER field:

- It must be the last field described in an OCCURS segment. If you are using MAPVALUE, then MAPVALUE must be the last field preceded by the ORDER field.
- The field name is arbitrary.
- The ALIAS attribute must be ORDER.
- The USAGE attribute must be I*n*, with any appropriate edit options; "*n*" is a number from 1 to 9.
- The ACTUAL attribute must be I4.

For example:

```
FIELD=ACT_MONTH, ALIAS=ORDER, USAGE=I2, ACTUAL=I4,$
```

Order values are assigned sequentially (1,2,3,...) and the order value is reset to 1 when a new instance of the parent segment is retrieved. The value is assigned prior to any selection tests that might accept or reject the record, and can be used in a screening condition. For example, to obtain data for the month of June, type:

```
SUM AMOUNT...
IF ORDER IS 6
```

Example: Describing Repeating Embedded Data With Record Type Indicators

If a file contains records that have repeating embedded data, the OCCURS attribute is used to describe a separate segment for the repeating fields. In some files, however, the repeating fields themselves must be identified according to a record type indicator. Suppose you want to describe a file that, schematically, looks like the following:

```
A  Rectype B C     Rectype  B C
A  Rectype D       Rectype  D
```

You would 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.

This is its Master File:

```
FILE=ABCD, SUFFIX=RMS,$
 GROUP=ONE            ,ALIAS=KEY      ,ACTUAL=A2   ,USAGE=A2   ,$
  FIELDNAME=A            ,ALIAS=      ,USAGE=A2   ,ACTUAL=A2   ,$
 SEGNAME=BC_SEG, SEGTYPE=S0, PARENT=ONE, OCCURS=2,$
  FIELDNAME=RECTYPE      ,ALIAS=1     ,USAGE=I2    ,ACTUAL=I2   ,$
  FIELDNAME=B            ,ALIAS=      ,USAGE=A5    ,ACTUAL=A5   ,$
  FIELDNAME=C            ,ALIAS=      ,USAGE=A5    ,ACTUAL=A5   ,$
 SEGNAME=D_SEG, SEGTYPE=S0, PARENT=ONE, OCCURS=2,$
  FIELDNAME=RECTYPE      ,ALIAS=2     ,USAGE=I2    ,ACTUAL=I2   ,$
  FIELDNAME=D            ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15  ,$
```

Each of the two descendant OCCURS segments in this example depends on the record type indicator that appears for each occurrence.

All the rules of syntax for using RECTYPE fields and OCCURS segments apply to RECTYPEs within OCCURS segments.

Since the 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 files to be described, including those with nested and parallel repeating groups that depend on RECTYPEs. For example:

```
A   Rectype B C   Rectype D   Rectype D   Rectype E   Rectype E
```

In this case, B/C, and D represent a nested repeating group, and E represents a parallel repeating group.

Syntax: How to Describe Repeating Groups Depending on a Preceding Record Type Indicator Using MAPFIELD/MAPVALUE

MAPFIELD is assigned as the ALIAS of the field that will be the record type indicator. You can give this field any name; it is otherwise described according to the usual syntax:

```
FIELD=name, ALIAS=MAPFIELD, USAGE=usage, ACTUAL=actual,$
```

The descendant segments, whose values depend on the value of the MAPFIELD, 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. If an ORDER field is chosen, it must be used before the MAPVALUE. The actual MAPFIELD value is supplied as the MAPVALUE's ALIAS.

The syntax is

```
FIELDNAME=MAPVALUE,ALIAS=alias,USAGE=usage{Pn} ACTUAL= {Pn} ,
ACCEPT=list/range],$
```

where:

`MAPVALUE`

Indicates that the segment depends on a MAPFIELD in its parent segment.

`alias`

Is the primary MAPFIELD value that identifies the segment. 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 a list of one or more lines of specific 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 an item in the list contains embedded blanks or commas, it must be enclosed within single quotation marks ('). The list may contain a single MAPFIELD value. For

```
FIELDNAME=MAPVALUE, 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 either value in 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.

In some cases, the record type indicates what kind of repeating data will follow. Schematically, the records would look like the following:

```
A  B  Recordtype  (1)   C D   C D   C D
A  B  Recordtype  (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 type indicator. The second record has a different record type indicator and contains a different repeating group, this time for E.

Since the OCCURS segments are identified by the record type indicator, rather than the parent A/B segment, you can use the keyword MAPFIELD. MAPFIELD identifies a field like RECTYPE, but, since the OCCURS segments will each have specific values for MAPFIELD, its value is associated with each OCCURS segment by means of a complementary field named MAPVALUE.

The Master File for this file would look like the following:

```
FILENAME=EXAMPLE,SUFFIX=RMS,$
GROUP=ONE           ,ALIAS=KEY        ,ACTUAL=A2     ,USAGE=A2   ,$
FIELDNAME=A         ,ALIAS=           ,USAGE=A2      ,ACTUAL=A2,$
FIELDNAME=B         ,ALIAS=           ,USAGE=A10     ,ACTUAL=A10,$
FIELDNAME=FLAG      ,ALIAS= MAPFIELD  ,USAGE=A1      ,ACTUAL=A1,$
SEGNAME=TWO,SEGTYPE=S0,PARENT=ONE,OCCURS=VARIABLE,$
FIELDNAME=C         ,ALIAS=           ,USAGE=A5      ,ACTUAL=A5,$
FIELDNAME=D         ,ALIAS=           ,USAGE=A7      ,ACTUAL=A7,$
FIELDNAME=MAPVALUE  ,ALIAS=1          ,USAGE=A1      ,ACTUAL=A1,$
SEGNAME=THREE,SEGTYPE=S0,PARENT=ONE,OCCURS=VARIABLE,$
FIELDNAME=E         ,ALIAS=           ,USAGE=D12.2   ,ACTUAL=D8,$
FIELDNAME=MAPVALUE  ,ALIAS=2          ,USAGE=A1      ,ACTUAL=A1,$
```

Assigning a Keyed RMS File to a Master File

In this section:

In order to associate a keyed RMS file to a Master File, an Access File must exist.

The Access File will have the following format

```
RMSFILE    = filename,
[ACCESS    = {SHARED|READONLY},]
[LOCKMODE = {SKIP|KEEP} ,]
[STATMODE = {EXCEPTIONS|ON|OFF},] $
```

where:

filename

Is one of the following:

- A fully qualified file name, including files on remote hosts.
- A logical that points to a local or fully qualified RMS file.

The Access File is also used to specify ACCESS, LOCKMODE, and STATMODE for RMS files.

- ACCESS can have a value of SHARED for read/write access, or READONLY for Read access.
- LOCKMODE is for retrieval only and can have a value of SKIP or KEEP. If you select SKIP, the Adapter for RMS will skip locked records. If you select KEEP, the Adapter for RMS will retrieve locked records.
- STATMODE can have a value of EXCEPTIONS, ON, or OFF. EXCEPTIONS is the default and will display a message when locked records are encountered. ON allows a message to occur (whether or not locked records are encountered), and OFF suppresses locking messages.

See *File and Record Locking* on page 39-42 for more information on ACCESS, LOCKMODE, and STATMODE.

You can override the RMSFILE declaration in the Access File with a FILEDEF. The FILEDEF can be done locally within a procedure (before accessing the file) or within the profile. For example,

```
FILEDEF rmsfile DISK filename
```

where:

rmsfile

Is the logical metadata reference name matching the physical file name.

filename

Is the OpenVMS file name. It may be any valid OpenVMS file specification or logical name pointing to a file.

File and Record Locking

This topic describes features that enable a developer to control file and record locking. Lock conflict, or contention, can occur at file level or record level. File contention is caused by incompatible access by two or more processes. Record contention is caused when a requested lock is incompatible with any existing locks on the record.

The following terms summarize the types of access allowed to RMS files and records:

Read Access	Provides users with access to a file and the records in the file for actions that make no alterations. Read can use any access mode (dependent on site needs) for the actions that are defined in the following tables.
Read/Write Access	Provides users with access to a file and the records in the file for actions that make alterations to records, add records, or delete records. Read/write access requires the use of SHARED as the access mode. All other modes are not supported WRITE operations.

File Locking and the Interface to RMS

When your request attempts to open an RMS file, the Access Parameter within the Access File determines what you can do at a file level with the RMS file, and what actions others can take while you have the file open.

When a request attempts to open an RMS file, the Access Parameter determines which type of access to use. If the Access Parameter specification is incompatible with another process' access to the file, the procedure fails because it is not able to open the file. Additionally, if the file is opened for read, but a subsequent write is attempted, it will fail due to an inappropriate access mode.

Depending on how a file is opened, as specified by the Access Parameter, subsequent opens are limited as follows:

		Processes by Others			
		`READONLY`	`SHARED`	`PROTECTED`	`EXCLUSIVE`
Processes by Server Users	`READONLY`	Allowed	Allowed	Allowed	Denied
	`SHARED`	Allowed	Allowed	Allowed when SHAREd process is doing read-access operation.	Denied

This table describes user access and file sharing options used in response to the following Access File options for the Access Parameter.

Access Parameter	**User Access**	**File Sharing to Other Processes**
`READONLY`	Read	Read and Write
`SHARED`	Read and Write	Read and Write

Record Locking

RMS requests are no-lock for each record retrieved for READ operations.

Handling Locked Records During Table Read Request

In this section:

LOCKMODE

STATMODE

RMS files can be accessed while they are simultaneously accessed by other programs. For instance, the file might be in use by another program that is maintaining it by adding, deleting, or updating records.

When a READ operation is rejected due to a lock conflict by another process, aborts the request and displays informative messages. You can override this behavior on a file-by-file basis using the LOCKMODE and STATMODE options in the Access File. LOCKMODE and STATMODE do not apply to WRITE requests such as UPDATE or DELETE.

LOCKMODE

LOCKMODE can affect record retrieval in one of two ways. Each of these settings and the affect they have on the retrieval process is as follows:

- Omitting LOCKMODE from the acx file.

  To halt retrieval when locked records are encountered, omit LOCKMODE from the acx file. A (FOC1325) RMS READ LOCK ABORT ERROR will be sent if locked records are encountered during record retrieval. Records retrieved before the lock is encountered may have already been sent to the client and should be discarded.

- Setting one of two acceptable values in the acx file.

- To retrieve all records, including locked records, code LOCKMODE=KEEP in the acx file. For example:

  ```
  LOCKMODE = KEEP
  ```

- To retrieve all records, except locked records, and complete the request, code LOCKMODE=SKIP in the acx file. For example:

  ```
  LOCKMODE = SKIP
  ```

STATMODE

STATMODE controls whether or not a message is sent to the client program about retrieved records. STATMODE has three settings:

- ON

  To receive a message giving the statistics of data retrieval, set STATMODE to ON in the Access File. The message displays, in addition to the report request, after data retrieval is complete. For example:

  ```
  STATMODE = ON
  ```

  A FOC1320 message displays when no locked records are encountered. For example:

  ```
  (FOC1320) RMS STATS  :(SEGCAR   ) Reads = 1, Skips = 0, Keeps = 0
  ```

  If LOCKMODE is set to KEEP a FOC1324 message displays. For example:

  ```
  (FOC1324) RMS KEEP   :(SEGCAR   ) Reads = 1, Skips = 2, Keeps = 0
  ```

  If LOCKMODE is set to SKIP, a FOC1322 message displays. For example:

  ```
  (FOC1322) RMS SKIP   :(SEGCAR   ) Reads = 1, Skips = 0, Keeps = 2
  ```

- OFF

  Set the STATMODE to OFF if no messages are to be sent when locked records are encountered. The client application will not have any indication whether or not locked records were encountered during record retrieval.

- EXCEPTIONS

  To receive a message giving the statistics of data retrieval, only if locked records are encountered, set the STATMODE to EXCEPTIONS. The message displays, in addition to the report request, after data retrieval is complete. No message displays if no locked records are encountered during retrieval. For example:

  ```
  STATMODE = EXCEPTIONS
  ```

  If LOCKMODE is set to KEEP a FOC1324 message displays. For example:

  ```
  (FOC1324) RMS KEEP   :(SEGCAR   ) Reads = 1, Skips = 2, Keeps = 0
  ```

  If LOCKMODE is set to SKIP, in addition to the report output, a FOC1322 message displays. For example:

  ```
  (FOC1322) RMS SKIP   :(SEGCAR   ) Reads = 1, Skips = 0, Keeps = 2
  ```

Access File Examples

The following is an example of a Remote Node:

```
RMSFILE=HOST1::DISK$PROG:[PROD.INFO]EMPINFO.DAT,
ACCESS=SHARED,$
```

The following is an example of a Full Path as defined by a logical and a file name:

```
RMSFILE=DATADIR:PERSON.DAT,ACCESS=READONLY,$
```

The following is an example of a Logical Name:

```
RMSFILE=MYLOGICAL,ACCESS=SHARED,$
```

The following is an example of a Full Path on a file using LOCKMODE and STATMODE:

```
RMSFILE=DISK100:[DATA]PERSON.DAT,
ACCESS=READONLY,LOCKMODE=SKIP,STATMODE=OFF,$
```

Retrieving Data From RMS Files

In this section:

The following topics outline the procedures necessary for retrieving data from RMS files.

Index Selection

By default, the primary key is used for retrieval of records from indexed RMS files. You can override this default with the Automatic Index Selection (AIS).

The primary benefit of keys is improved efficiency in record retrieval. They provide an alternate, more efficient, retrieval method and can be used with screening tests on the selected key which are translated into direct reads. Indexes can also be used to control the order of retrieval of records.

To screen more than one field when working with group keys that are comprised of multiple fields, use a slash (/) character to separate the components of the group key. Screening is limited to NE, IS, and EQ relationships. The syntax is:

```
... groupname {NE}[']value/value.../value[']
              {IS}
              {EQ}
```

where:

groupname

Is the GROUP field name in the Master File.

The slash (/) separates the parts of the group field. If you omit the optional quotation marks, embedded blanks in the value will be removed. Use single quotations marks (') around the values to preserve the values exactly as typed.

Automatic Index Selection (AIS)

The Automatic Index Selection (AIS) facility automatically selects a key for direct access to an indexed file based on a request.

Requirements for Using AIS

AIS automatically uses a key for direct retrieval when an applicable screening condition is reached in a request. AIS is used when all of the following requirements are met:

- Primary keys are described in a Master File with the ALIAS=KEY or ALIAS=DKEY attribute placed on the GROUP declaration (not on every field in the GROUP). Note that the primary key must always be described as a GROUP.
- Secondary keys are described with the FIELDTYPE attribute on the GROUP or on the FIELD declaration for a secondary key not belonging to a group.
- There is an optimizable screening condition on a key field. If a key is defined as a GROUP, the screening condition must be on either the GROUP name, or on the first field of the key. Only those key fields in a screening condition with one of the following logical relations are used for index selection:
    - Equal
    - Less than or equal to
    - Less than
    - Greater than
    - Greater than or equal to
- AIS applies only to keys in the root segment.

Syntax for RMS Master File Attributes

This topic details the syntax for the attributes used to describe files, segments, and fields.

File Attributes

File Declaration:

```
FILE[NAME]=filename, [FILE]SUFFIX=type ,$
```

Attribute	Function	Value
`FILE[NAME]`	Identifies the Master File.	All characters and digits, but no embedded blanks. Eight characters maximum.
`[FILE]SUFFIX`	Identifies the type of file to which the Master File applies.	RMS.

Note: The SUFFIX values listed here pertain to the RMS files; the SUFFIX parameter can have other values for data that resides on other Database Management Systems that can be accessed. Consult the appropriate Adapter manuals for these products.

Except for the following differences, the description of data and relationships between fields within a file is the same for keyed (indexed) RMS files, FIX (fixed-format sequential) files, and COM (comma-delimited) files. Keyed (indexed) RMS uses the RMS File System to access data using information in the Access File declarations. The others use sequential access using a FILEDEF to identify and access the file. Index related declarations do not apply to fixed-sequential or comma-delimited files.

Segment Attributes

Segment Declaration:

```
{SEGNAME|SEGMENT}=segname[,SEGTYPE=S0][,PARENT=parent_name]
               [          {n         }]
               [, OCCURS= {fieldname}]   [, POSITION=fieldname],$
               [          {VARIABLE }]
```

Attribute	Function	Value
`SEGNAME` `SEGMENT`	Identifies a collection of data fields that are related.	All characters and digits, except special characters. Up to eight characters in length.
`SEGTYPE`	Identifies the physical storage of the segment.	S0.
`PARENT`	Identifies the “parent” or owner of the current segment.	Any valid segment name previously defined in the Master File.
`OCCURS`	Identifies the field as a multiply-occurring field and specifies how the number of occurrences will be determined.	Any integer from 1 to 4095 or any valid field name or VARIABLE.
`POSITION`	Identifies the field in the parent that marks the beginning of the multiply-occurring fields.	All characters and digits, except special characters.

Field Attributes

Field Declarations:

```
                {fieldname}         {alias                 }
FIELD[NAME]=    {FILLER    } ALIAS= {rectype identifier},USAGE=usage,
                {RECTYPE   }        {ORDER                 }
                {MAPVALUE  }        {KEY(n)                }
                                    {MAPFIELD              }

ACTUAL=actual  [,FIELDTYPE =I,ACCEPT{list }]
               [,INDEX=I             {range}]
               [,TITLE='text'][,DESCRIPTION]=description],$

GROUP= {groupname|keyname} ,ALIAS={DKEY(n)|KEY(n)}
       ,USAGE=usage,ACTUAL=actual [,FIELDTYPE=I],$

[DEFINE name/format=expression;$]
```

Attribute	Function	Value
FIELDNAME	Uniquely identifies a data item in the file.	All characters and digits, with a maximum of 64.
ALIAS	Alternate field name, sometimes used to provide additional information about the field.	All characters and digits, except special characters, with a maximum of 64.
USAGE	Describes the format of the field as interpreted for usage (display) purposes.	See *RMS Attribute Summary* on page 39-52.
ACCEPT	Assigns a list or range of acceptable values for RECTYPE or MAPVALUE.	See *RMS Attribute Summary* on page 39-52.
ACTUAL	Describes the format of the field as it exists in the external file.	See *RMS Attribute Summary* on page 39-52.
FIELDTYPE	Identifies a group field on keyfield.	Can be only one character, which is I.
TITLE	Provides one or more lines of text to be used as an alternate column heading for the field name on reports.	All characters and digits with a maximum of 64.

Attribute	**Function**	**Value**
`DESCRIPTION`	Documents the meaning of a field in the Master File.	All characters and digits, with a maximum of 44.
`GROUP`	Identifies a key.	All characters and digits, except special characters, with a maximum of 48.
`DEFINE`	Creates a temporary field for reporting purposes.	Mathematical or logical statement made up of constants, field names, or other DEFINE fields.

RMS Attribute Summary

This topic contains a summary of the attributes.

Throughout the summary we refer to special characters and denote this reference with an "*". You should avoid using the special characters listed below, as they may impose operational restrictions in some environments.

+ (plus sign)	' (apostrophe)	" (double quotation marks)
- (hyphen)	; (semicolon)	< (less than sign)
$ (dollar sign)	b/ (blank)	> (greater than sign)
* (asterisk)	, (comma)	= (equal sign)
/ (slash)	| (concatenation symbol)	. (period)
() (parentheses)		

Attribute:	**ACCEPT**
Length:	1 to 255 characters
Example:	`ACCEPT = A OR B OR C`

Function:	Is optional, and enables you to assign a list or range of acceptable values to a RECTYPE or MAPVALUE field in a file.
	
	The syntax is
	
	`ACCEPT = {`*`list`*`|`*`range`*`}`
	
	where:
	
	`list`
	Is a string of acceptable values:
	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 line. The list is terminated by a comma (,).
	
	`range`
	Gives the range of acceptable values:
	value1 TO value2
	For example:
	`ACCEPT = 150 TO 1000`

Attribute:	**ACTUAL**	
Length:	1 to 8 characters.	
Function:	Describes the type and length of a field as it actually exists in a file. The source of this information is an existing description of the file. The following chart shows the data format types that the server reads:	
	Type	**Meaning**
	`An`	Alphanumeric characters A–Z, 0–9, and other ASCII display characters, where *n* = 1–256.
	`D8`	8-byte double-precision floating-point numbers.
	`F4`	4-byte single-precision floating-point numbers.
	`In`	Binary integers: I1 Single-byte binary integer I2 2-byte binary integer I4 4-byte binary integer I8 8-byte binary integer
	`Pn`	Packed decimal internal format. The length is the number of bytes, each of which contains two digits, except for the last byte which contains a digit and the sign (+ or -). P6 means 11 digits plus a sign, packed two digits to the byte, for a total of six bytes of storage, where *n* = 1–8.

Function:	**Type**	**Meaning**

(continued)	`Zn`	Zoned decimal format (numeric string) where *n* is the number of digits (1 to 31), each of which takes one byte of storage. The last byte contains a digit and the sign.
		There are several standards for zoned data. For Read purposes, only right overpunched standards are supported and can be determined on Read since they are unique.
		The specific format to use when writing data must be known for read/write purposes. The default is ASCII right overpunch. To change the default, you must edit the EDACONF [.BIN]EDAENV.COM file and add the following logical:
		`DEFINE /NOLOG IBI_ZONED_OUT_TYPE {1|2}`
		where:
		`1`
		Is the ASCII right overpunched standard (default).
		`2`
		Is the EBCDIC right overpunched standard.
		Zoned right separate numeric or zoned left overpunched numeric formats are not supported.

	`DATE`	Unless your file was created by a program, all of the characters will be in ASCII format type A (alphanumeric). The server permits the following conversions from ACTUAL format to USAGE (display) format:

USAGE	**ACTUAL**
`I`	`I`
`P`	`P`
`P,D,I,F`	`Z`
`D`	`D`
`F`	`F`
`P`	`I8`
`YYMD`	`DATE`

For a complete list of date formats, see *Segment Attributes* on page 39-11.

Note: The ACTUAL value of DATE indicates that the field is an OpenVMS 64-bit datetime stamp. This usage also requires an A4 filler field immediately following the date field in the Master File.

Attribute:	**ALIAS**
Alias:	SYNONYM
Length:	1 to 12 characters
Permitted Values:	All characters and digits.*
Example:	`ALIAS=CTY`
Function:	Is an alternate name to identify a data field. Since references to data fields are based on the names, or aliases, or shortest unique truncation, it is useful to make the ALIAS a short abbreviation representative of the data. Short, simple names are best, with no embedded blank spaces or special characters.* For example, PART CODE or PART-CODE should be PARTCODE or PART_CODE, or PC. Names must begin with a letter, but can contain numbers. Do not use names that might conflict with report column names (C1, C2, ...), row names (R1, R2, ...) or HOLD file aliases (E01, E02, ...). Also, avoid reserved keywords such as BY or ACROSS. ALIAS has specialized functions when used with KEY definitions, RECTYPE and MAPFIELD, as discussed at the beginning of this manual.

Attribute:	**DEFINE**
Example:	`DEFINE PROFIT/D8 = RETAIL_COST - DEALER_COST;$`
Function:	Is optional, and enables you to store definitions for temporary fields in your Master File for reporting purposes. The syntax is `DEFINE` *`name[/format]`* `=` *`expression`*`;$` where: *`name`* Is a 1 to 48 character field name for the defined field. *`format`* Is the optional display format for the defined field, separated from *name* by a slash (/). The display format for the field follows the rules for USAGE formats described under field attributes. The default value is D12.2. *`expression`* Can be either a mathematical or logical statement, made up of constants, database fields, and temporary defined fields. The expression must end with a semicolon (;) followed by a dollar sign ($). A DEFINE is placed at the end of the segment it is associated with. A defined field stored in a Master File can refer only to fields in its same segment. To create a defined field that refers to fields in other segments, create a temporary field using the DEFINE command prior to your report request.

Attribute:	**DESCRIPTION**
Length:	1 to 44 characters
Permitted Values:	All characters and digits.
Example:	`DESCRIPTION=STANDARD COST CATEGORY,$`

Function:	Is optional, and is used only for documenting the meaning of a field in the Master File. It is ignored during processing. Since a Master File is a comma-delimited file that the server can read directly, it is possible to prepare reports in various formats whose subjects are the names and attributes of the data fields.
	
	If the DESCRIPTION contains a comma (,), the entire text must be enclosed in single quotation marks ('). For example:
	
	`DESCRIPTION='TOTAL COST, NOT NORMALIZED',$`
	
	Another way to add comments to a Master File is to place them after the dollar sign ($).
	
	Descriptions of DEFINE fields in the Master File are placed on separate lines.
	
	For example:
	
	`DEFINE ITEMS_SOLD/D8 = INVENTORY - ORDERED`
	`;DESC=DAMAGED ITEMS NOT INCLUDED,$`
	
	Note: When used on a DEFINE expression, the dollar sign ($) does not immediately follow the semicolon (;).
	
	The semicolon (;) after a DEFINE must display on the same line as the attribute, which follows the DEFINE.

Attribute:	**FIELDNAME**
Alias:	FIELD
Length:	1 to 64 characters
Permitted Values:	All characters and digits.*
Example:	`FIELD=INVENTORY`
Function:	Identifies the data items in a file. Names must be unique within a file.* The full field name is used as the default title for the data printed on reports. Hence, names representative of the data should be selected. Names must begin with a letter, though they may contain numbers. Avoid names that might conflict with report column names (C1, C2, ...), row names (R1, R2, ...) or HOLD file aliases (E01, E02, ...). Also avoid reserved keywords such as PRINT, BY, ACROSS, and so on. FIELDNAME has specialized functions when used with RECTYPE and MAPVALUE, as discussed at the beginning of this manual.

Attribute:	**FIELDTYPE**
Alias:	INDEX
Length:	1 character
Permitted Values:	I
Example:	`FIELDTYPE=I`
Function:	Identifies a group field or a field that serves as a secondary key, which indicates that this secondary key can be used for direct retrieval. This attribute only applies to keyed (indexed) RMS files.

Attribute:	**FILENAME**
Alias:	FILE
Length:	1 to 8 characters
Permitted Values:	All characters and digits, but no embedded blanks.*
Example:	`FILENAME=EQUIP`
Function:	Is optional, and is used for documentation purposes. It should correspond to the external file name of the Master File.

Attribute:	**GROUP**
Alias:	KEY
Length:	1 to 66 characters
Permitted Values:	All characters and digits, but no embedded blanks.*
Example:	`GROUP=PUBKEY`
Function:	Is used to describe the primary key in a keyed file or a field that is composed of subfields. The USAGE format and ACTUAL format of the GROUP is calculated using the USAGE and ACTUAL formats of the FIELDS that comprise the GROUP. Names must begin with a letter, though they may contain numbers. Avoid using names that might conflict with report column names (C1, C2, ...), row names (R1, R2, ...), or HOLD file aliases (E01, E02, ...).

Attribute:	**PARENT**
Alias:	PARENT
Length:	1 to 8 characters
Permitted Values:	All characters and digits, but no embedded blanks.
Example:	`PARENT=CARSEG`
Function:	Is the name of the segment that is the parent or "owner" of the current segment. In the Master File, the information describing the parent must precede any reference to it as a parent. The first, or "root" segment, in a file cannot have a parent by definition, but the name "SYSTEM" may be used, or the attribute left blank. Every other segment must have a parent named. If no parent is named, the immediately preceding segment will be used by default.

Attribute:	**SEGNAME**
Alias:	SEGMENT
Length:	1 to 8 characters
Permitted Values:	All characters and digits, but no embedded blanks.*
Example:	`SEGNAME=MODSEG`
Function:	Identifies a collection of data fields. Segments which have a relationship to each other must have unique names within a given file description.

Attribute:	**SUFFIX**	
Alias:	FILESUFFIX	
Length:	1 to 8 characters	
Permitted Values:	RMS	
Example:	`SUFFIX=RMS`	
Function:	In order to identify the type of file the description applies to, the SUFFIX is given one of the following values:	
	Value	**Meaning**
	`RMS`	A keyed RMS file, which uses an Access File to locate and access data using the RMS Index.
	`ISAM`	A keyed RMS file. ISAM is supported as an alternate keyword for backward compatibility to FOCUS 6.x applications.

Note: SUFFIX values for other proprietary databases that can be accessed are described in the appropriate manuals for these options.

Attribute:	**TITLE**
Length:	1 to 64 characters.
Permitted Values:	All characters and values.
Example:	`TITLE='Products'`

Function:	Is optional, and enables you to supply an alternate column title to replace the field name that is normally used.
	
	The syntax is
	
	`TITLE='`*`text`*`'`
	
	where:
	
	`text`
	Must be enclosed within single quotation marks (') and can contain 1 to 64 characters.
	
	`TITLE`
	cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself. To display the TITLE on more than one line in the report, use commas to divide the text. To include blanks at the end of a column title, simply type them in and enter a slash (/) in the final blank position.
	
	TITLE attributes do not apply when direct operations (PCT., AVE., ...) are used on the field. To rename such columns, use the AS phrase.

Changes:	You can override both field names and TITLE attributes with AS phrases in your request. You can issue a SET TITLE command to change the default titles to either the field names or titles supplied in the Master File. You can change the TITLE attribute in the Master File. **Note:** Client tools using the API do not have access to the TITLE attribute. The TITLE attribute is available only with WebFOCUS.

Attribute:	**USAGE**
Length:	1 to 8 characters
Function:	Defines the report display format of the data field. This attribute defines the data field type, length, and any edit options that are to be applied when the field values are printed. Special date formats, which are actually an extended series of edit options, can also be used. The syntax is `USAGE=`*`usage`* where: *`usage`* Consists of three parts: *`tllleeeee`* `t` = field type `lll` = field display length `eeeee` = edit options

Permissible USAGE field types and lengths are shown in the chart below:

USAGE	**Length**	**Description**
`A`	1–9,095	Alphanumeric text
`D`	1–19	Decimal, double-precision numbers
`F`	1–9	Decimal, single-precision numbers
`I`	1–11	Integer values (no decimal places)
`P`	1–17	Packed decimal numbers
`YYMD`	10	Displayed as YYMD

Edit options only affect printed or displayed fields. They are not active for extract files.

The following table summarizes the edit options and includes sample USAGE values:

Edit Option	**Meaning**	**Effect**
`%`	Percent sign	Percent sign Displays a percent sign along with numeric data. Does not calculate the percent.

Edit Option	**Meaning**	**Effect**
`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	Scientific notation Displays only significant digits.
`L`	Leading zeroes	Adds leading zeroes.
`M`	Floating $ (for US code page)	Places a floating dollar sign $ to the left of the highest significant digit. **Note:** The currency symbol displayed depends on the code page used.
`MDY`	Month-Day-Year	Displays alphanumeric or integer data as a date in the form month/day/year.
`N`	Fixed $ (for US code page)	Places a dollar sign $ to the left of the field. The symbol displays only on the first detail line of each page. **Note:** The currency symbol displayed depends on the code page used.
`R`	Credit (CR) negative	Places CR after negative numbers.
`S`	Zero suppress	Zero suppress If the data value is zero, prints a blank in its place.

Read/Write Usage Limitations of the Adapter for RMS

In this section:

The adapter has certain limitations when used for write purposes. The following topics discuss these issues and possible alternatives that may be used.

OCCURS Statements in a Master File

OCCURS statements in a Master File are not directly supported by the write portion of the adapter. If an OCCURS statement is for a specific number of instances, an alternate Master File may be coded by using specific naming (for example, AMTDUE OCCURS=12 would be coded as AMTDUE01, AMTDUE02, and so on). The write application must then reference the alternate Master File using the specific fields.

For example:

```
SEGNAME=MTH, SEGTYPE=S0, PARENT=YEAR, OCCURS=12,$
FIELD=PAYABLES, AP, I8 , I4 , $
```

would be coded as

```
SEGNAME=MTH, SEGTYPE=S0, PARENT=YEAR, $
FIELD=PAYABLES01, AP01, I8 , I4 , $
FIELD=PAYABLES02, AP02, I8 , I4 , $
FIELD=PAYABLES03, AP03, I8 , I4 , $
FIELD=PAYABLES04, AP04, I8 , I4 , $
FIELD=PAYABLES05, AP05, I8 , I4 , $
FIELD=PAYABLES06, AP06, I8 , I4 , $
FIELD=PAYABLES07, AP07, I8 , I4 , $
FIELD=PAYABLES08, AP08, I8 , I4 , $
FIELD=PAYABLES09, AP09, I8 , I4 , $
FIELD=PAYABLES10, AP10, I8 , I4 , $
FIELD=PAYABLES11, AP11, I8 , I4 , $
FIELD=PAYABLES12, AP12, I8 , I4 , $
```

Commit Processing

The Adapter for RMS, in conjunction with RMS itself, treats a single SQL statement as a complete unit of work. This means that the server automatically commits after every SQL statement. If the front-end application sends an SQL COMMIT or SQL ROLLBACK statement(s), it will be ignored by the adapter.

SQL Commands

The following are acceptable SQL commands:

Command	Function
`SELECT`	Retrieves data for the entire table (*) or for specified columns.
`DELETE`	Removes one or more records from an RMS keyed file.
`INSERT INTO`	Adds data to an RMS keyed file.
`UPDATE`	Updates values of one or more columns in a record of an RMS keyed file.

SQL, MODIFY and MAINTAIN Operations

Releases prior to 5.2 of the server did not support multi-record SQL INSERT, DELETE and UPDATE except when done as a PREPARE plus MODIFY and MAINTAIN were not supported. This limitation is removed as of the 5.2 release. Recoding existing applications that used PREPARE is not required, all methods are supported.

CHAPTER 40

Using the Adapter for SAP Business Intelligence Warehouse (BW)

Topics:

The Adapter for SAP BW allows WebFOCUS for SAP Business Intelligence Warehouse (BW) and other applications to access SAP BW data sources through the SAP BW Multi-dimensional (OLAP) model. The adapter converts data or application requests into native SAP BW statements and returns optimized answers sets to the requesting program.

For sample requests and output, see *Producing SAP BW Requests* on page 40-46.

Preparing the SAP BW Environment

In this section:

Accessing Multiple Systems

SAP BW remote communications require:

- One of the following operating systems: Windows, Sun Solaris, HP-UX, or AIX. Availability may depend on operating system release level.
- SAP BW BASIS 4.6 or higher and SAP BW 2.0B or higher.
- Installation of the SAP BW RFC/SDK on the server system. For details, see your SAP documentation.
- A TCP/IP connection to the SAP BW destination machine.

The following SAP Authorization profiles are required at a minimum. This list may change depending on your SAP BW Release and/or site specific Authorization Profiles. These authorization requirements are determined based on a 3.0A system.

LIST OF AUTHORIZATION OBJECTS, FIELDS AND REQUIRED VALUES

S_RFC: Authorization check for RFC access

Field	Value
ACTVT (Activity)	16 (execute)
RFC_NAME (Name of RFC to be protected)	RSAB, RSOB, SYST
RFS_TYPE (Type of RFC object to be protected)	FUGR (function group)

S_RS_ADMWB: Administrator Workbench - Objects

Field	Value
ACTVT (Activity)	03 (Display), 16 (Execute)
RSADMWBOB (Administrator Workbench object)	INFOOBJECT

S_RS_COMP: Business Explorer - Components

Field	**Value**
ACTVT (Activity)	03 (Display), 16 (Execute)
RSINFOAREA (InfoArea)	* (All or specific InfoAreas)
RSINFOCUBE (InfoCube)	* (All or specific InfoCubes)
RSZCOMPID (ID of a reporting component)	* (All)
RSZCOMPTP (Component type)	CKF, REP, RKF, STR, VAR

S_RS_COMP1: Business Explorer - Components: Enhancements

Field	**Value**
ACTVT (Activity)	03 (Display), 16 (Execute)
RSZCOMPID (Name (ID) of a reporting component)	* (All)
RSZCOMPTP (Type of a reporting component)	CKF, REP, RKF, STR, VAR
RSZOWNER (Owner (Person Responsible) for a Reporting Component)	* (All)

S_RS_ICUBE: Administrator Workbench - InfoCube

Field	**Value**
ACTVT (Activity)	03 (Display))
RSICUBEOBJ (InfoCube Subobject)	DATA
RSINFOAREA (InfoArea)	* (All) or specific InfoAreas
RSINFOCUBE (InfoCube)	* (All) or specific InfoCubes

S_RS_ISET: Administrator Workbench -InfoSet

Field	**Value**
ACTVT (Activity)	03 (Display))
RSINFOAREA (InfoArea)	* (All) or specific InfoAreas
RSINFOSET (InfoSet)	* (All) or specific InfoSets
RSISETOBJ (InfoSet-Subobject)	DATA

S_RS_ODSO: Administrator Workbench - ODS Object

Field	**Value**
ACTVT (Activity)	03 (Display))
RSINFOAREA (InfoArea)	* (All) or specific InfoAreas
RSODSOBJ (ODS Object)	* (All) or specific ODS Objects
RSODSPART (Subobject for ODS Object)	DATA

Accessing Multiple Systems

The query adapter can operate across multiple R/3 systems. For each system, the query adapter requires one BW logon consisting of client, user, and password. This logon must be:

- RFC-enabled.
- Authorized for all clients that you may access.

You may need additional authorizations, depending on the SAP BW data you wish to access. For details, see your SAP BW administrator.

Configuring the Adapter for SAP BW

In this section:

Declaring Connection Attributes

How to:

Declare Connection Attributes Manually

Declare SAP BW Connection Attributes From the Web Console

Configure the Adapter on a MVS or OS/390 and z/OS Environment

Configuring the query adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to SAP BW, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Manually add the command in the global server profile (edasprof.prf) or in a user profile (user.prf).
- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the global server profile (edasprof.prf).

You can declare connections to more than one SAP BW data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to SAP BW takes place when the first query that references the connection is issued. If more than one SET CONNECTION_ATTRIBUTES command contains the same system, the query adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Syntax: How to Declare Connection Attributes Manually

Explicit. The user IDs and passwords are specified for each connection and passed to SAP BW for authentication and request execution.

```
ENGINE [BWBAPI] SET CONNECTION_ATTRIBUTES system/userid,password:'client'
```

Password Passthru. The user ID and password received from the client application are passed to SAP BW for authentication and request execution.

```
ENGINE [BWBAPI] SET CONNECTION_ATTRIBUTES system/:'client'
```

where:

`BWBAPI`

Indicates the Adapter for SAP BW. You can omit this value if you previously issued the SET SQLENGINE command.

`system`

Is the SAP BW System ID. It must point to a valid system entry in etc/sapserv.cfg.

`userid`

Is the SAP BW user ID.

`password`

Is the password for the user logon.

Note: To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

`client`

Is the SAP BW client for the user logon. This value must be enclosed in single quotation marks.

Procedure: How to Declare SAP BW Connection Attributes From the Web Console

To declare SAP BW connection attributes from the Web Console:

1. Launch the Web Console.
2. Click *Adapters* on the left pane to open the Adapter Configuration Window.
3. Click *Add*, then *SAP BW* to open the Add BW System Connect Parameters page in the right pane.

The configuration page prompts you for the following connection attributes. The settings are stored in etc/sapserv.cfg and the information is stored by blocks. A block is identified by the system ID (usually the three character name of the SAP BW instance).

User Authentication Parameters

The major work of the query adapter is to translate the user request into code that can be understood by SAP BW. For this purpose it requires a SAP BW user ID with a given set of privileges. In the following table this user ID is referred to as IBI_USER. After the request has been generated, and is ready for execution, it can be executed either under the IBI_USER ID, or under a less privileged user, referred to as USER; results are then based on the user's credentials.

System	Three-character long alphanumeric ID that is used to describe the SAP connection. Using the instance name of the SAP installation is a common practice but not mandatory.

Security	There are two methods by which a user can be authenticated when connecting to an SAP BW instance: **Explicit.** The user IDs and password are explicitly specified for each connection and passed to SAP BW for authentication and request execution. **Password Passthru.** The user ID and password received from the client application are passed to SAP BW for authentication and request execution.
IBI_CLIENT	SAP BW Client for the IBI logon, maximum three characters.
CLIENT	SAP BW Client for the user logon, maximum three characters.
IBI_USER	SAP BW user ID for the IBI logon.
USER	SAP BW user ID for the user logon.
IBI_PASSWORD	SAP BW password for the IBI logon, maximum eight characters.
PASSWORD	SAP BW password for the user logon, maximum eight characters.

Function Group Parameters

FPOOL	Does not apply for the Adapter for SAP BW unless the BW system is converted to SAP R/3. Leave the default value for BW.
DATACLASS	Does not apply. Leave the default value.
TMPDIR	Does not apply. Leave the default value.
RANGEBEG	Does not apply. Leave the default value.
RANGEEND	Does not apply. Leave the default value.

Connection Parameters

APPGRP	Does not apply for the Adapter for SAP BW unless the BW system is converted to SAP R/3. Leave the default value for BW.
MSGHOST	Does not apply. Leave the default value.
HOST	Host name of the SAP BW application server.

GWHOST	Host name of the machine where the SAP BW gateway process is running. In the case where there is only one SAP BW application server, gwhost and host will be the same.
SYSNR	SAP BW system number that is two characters.
LANGUAGE	Language installed on SAP BW to be used with the Adapter for SAP BW. It is one character long and defined in the SAP system. (for example, E for English, D for German).

Parameters for Mixed Character Code Set

The following parameters apply only when the server platform and the SAP instance platform do not use the same character code set (ASCII or EBCDIC). See *Supporting Mixed Code Page Environments* on page 40-9 for detailed information.

Parameter	**Description**
eda2sap	Server to SAP host conversion table, as generated in *Preparing the SAP BW Environment* on page 40-2. This is the conversion file where the server code page is the source and the SAP code page is the target. In most cases, this parameter will be blank.
sap2eda	SAP host to server conversion table file name, as generated in *Preparing the SAP BW Environment* on page 40-2. This is the conversion file in which the SAP code page is the source and the server code page is the target. In most cases, this parameter will be blank.

Note: If the SAP BW system is running on an MVS/USS environment, see *How to Configure the Adapter on a MVS or OS/390 and z/OS Environment* on page 40-9.

Once all necessary parameter values are entered, click *Configure* to save the entry in the edasprof.prf file.

After configuring the adapter, it is necessary to install the static function modules in the SAP data dictionary to complete the configuration.

Procedure: How to Configure the Adapter on a MVS or OS/390 and z/OS Environment

To configure the adapter if running on an MVS or OS/390 and z/OS environment, the SAP Code Page 0126 must be installed on the BW server. This code page will be provided with one of the upcoming SAP Basis Support Packages, possibly with 45. Follow the instructions from SAP to install the Code Page.

1. Create two conversion tables by following the instructions in *Supporting Mixed Code Page Environments* on page 40-9, and transfer the tables to the $EDACONF/etc directory.

   For example, if the iWay server environment uses the code page 1100, then two conversion tables, 11000126.CDP and 01261100.CDP, should be created and transferred to the $EDACONF/etc directory.

2. Edit the edastart.bat file and add the following two lines:

   ```
   export SAP_CODEPAGE=0126
   export PATH_TO_CODEPAGE=$EDACONF/etc/
   ```

   Note: Make sure to include the trailing "/" for the value of the PATH_TO_CODEPAGE parameter.

3. Assign the correct code page files to the parameters of eda2sap and sap2eda by using the Web Console or manually editing. Make sure to type CDP in capital letters. For example:

   ```
   eda2sap=01261100.CDP
   sap2eda=11000126.CDP
   ```

Supporting Mixed Code Page Environments

SAP BW, as well as adapters, can be installed on various platforms. If SAP BW is installed on a platform that uses a code page or character set different from the adapter platform, two character conversion tables must be generated to translate the different character sets, one for each direction of data flow.

For example, suppose that SAP BW is installed on an OS/400 EBCDIC machine and the Adapter for SAP BW is installed on an Intel Windows ASCII machine. If a request is going from the server to SAP BW, an ASCII to EBCDIC translation is required. If SAP BW is sending data back to the Adapter for SAP BW, an EBCDIC to ASCII translation is required. This section describes how to create conversion tables and where they must reside.

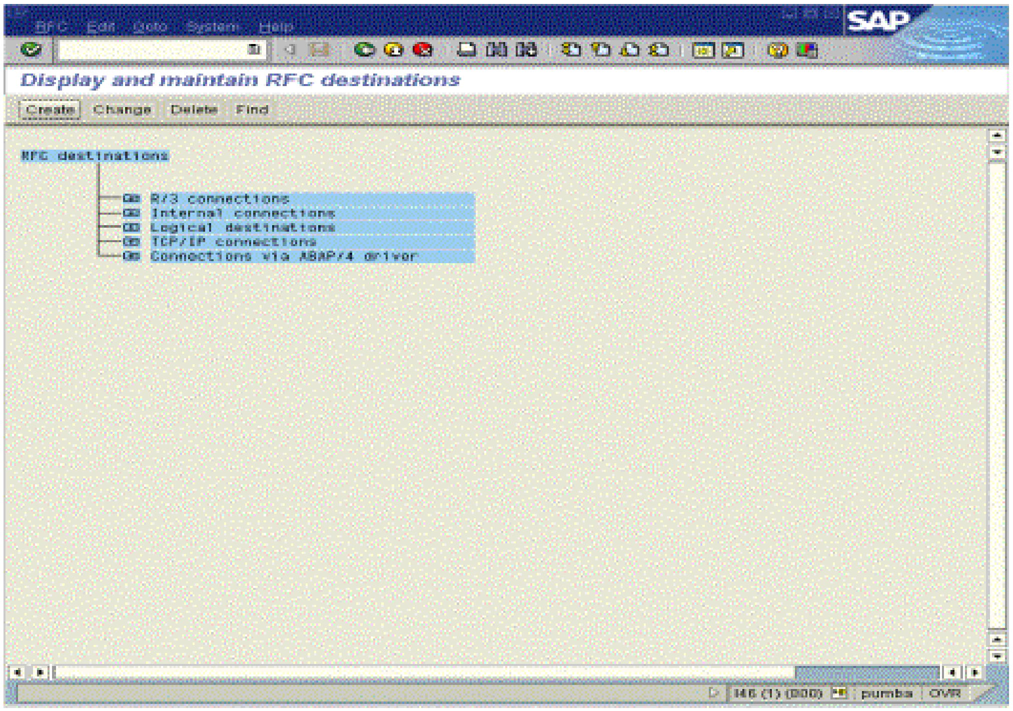
Character Conversion Tables

SAP BW provides a transaction, SM59, to generate conversion tables for RFC-based applications such as the Adapter for SAP BW. Once these tables are created, they must be copied to the adapter.

Procedure: How to Generate Conversion Tables

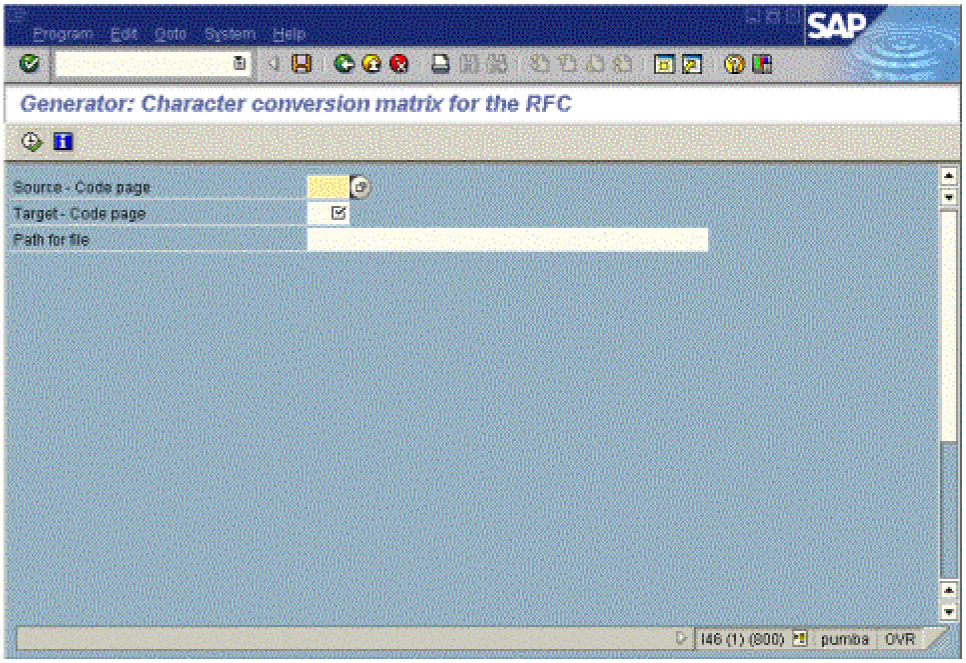
1. Type transaction SM59 at the SAP GUI command line.

   The following window opens.


RFC Edit Goto System Help
Display and maintain RFC destinations
Create Change Delete Find
RFC destinations
R/3 connections
Internal connections
Logical destinations
TCP/IP connections
Connections via ABAP/4 driver
I46 (1) (000) pumba OVR

2. Select the *Generate Conv.* Tab option from the RFC pull-down menu.

   The following window opens.



Generator: Character conversion matrix for the RFC

Source - Code page
Target - Code page
Path for file

3. Type a value for the Source code page, Target code page, and a valid path on the application server where the conversion table file is created. Optionally, you can supply a valid path and a unique file name.

4. Click *Execute* .

   If you did not specify a unique file name, the default file name of the newly created conversion table will contain the source and target code page values. For example, if you specified a source code page of 0102 (OS/400 for some CUAs) and a target code page of 1101 (7 bit USA ASCII pur), the default file name is 01021101.cdp. This is your OS/400 (EBCDIC) to ASCII conversion table file.

   It is now necessary to create another conversion table file for translating code pages in the opposite direction.

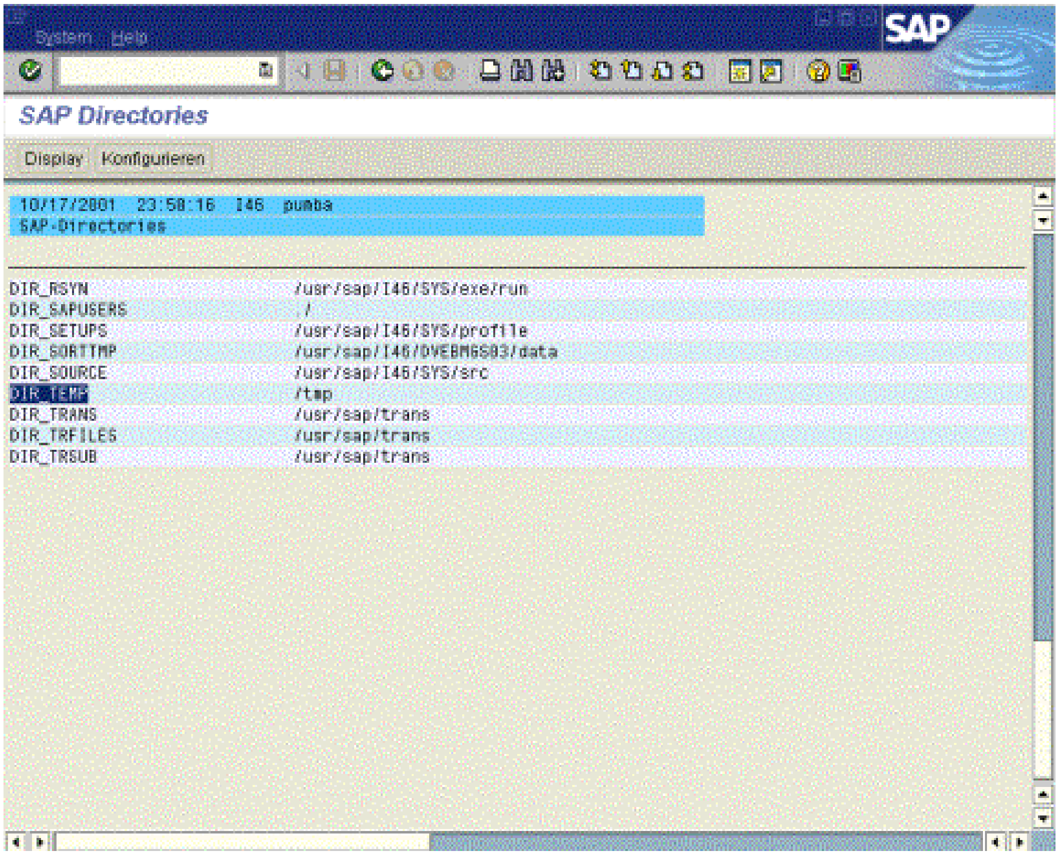
5. Switch the source and target code pages and generate the conversion table.

   Continuing with previous example, you should have a source code page of 1101 and a target code page of 0102, resulting in a default file name of 11010102.cdp. This is your ASCII to OS/400 (EBCDIC) conversion table file.

Note: If a unique file name was used for the first conversion table, be sure you specify a unique file name for the second conversion table.

Procedure: How to Download the Conversion Tables to the Server

Once these conversion table files are created, you can download them from the SAP application server directory to the server configuration ETC directory. Use SAP BW transaction AL11 to browse and download the two conversion table files to a local file on the desktop or Windows machine where the server is installed. Note that SAP GUI is required on the machine to which you are downloading the conversion table files. The following window represents transaction AL11:

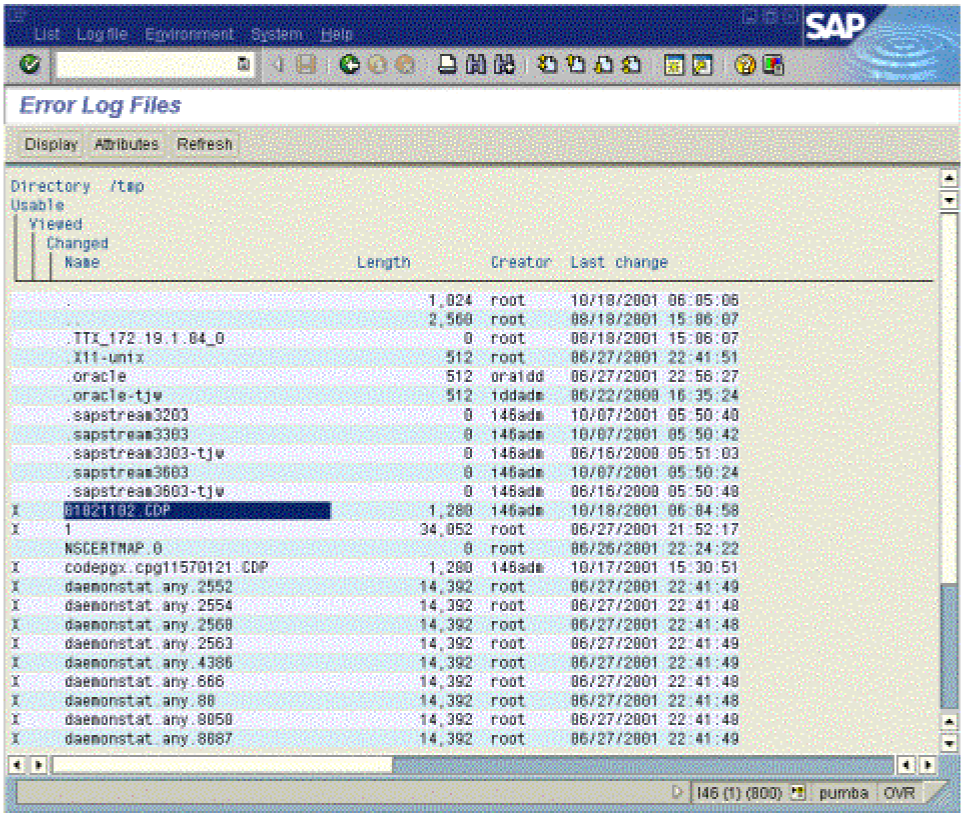


```
10/17/2001  23:50:16  I46  pumba
SAP-Directories

DIR_RSYN                    /usr/sap/I46/SYS/exe/run
DIR_SAPUSERS                ./
DIR_SETUPS                  /usr/sap/I46/SYS/profile
DIR_SORTTMP                 /usr/sap/I46/DVEBMGS03/data
DIR_SOURCE                  /usr/sap/I46/SYS/src
DIR_TEMP                    /tmp
DIR_TRANS                   /usr/sap/trans
DIR_TRFILES                 /usr/sap/trans
DIR_TRSUB                   /usr/sap/trans
```

1. Double-click the directory where the conversion table files were created.

   The following window opens.

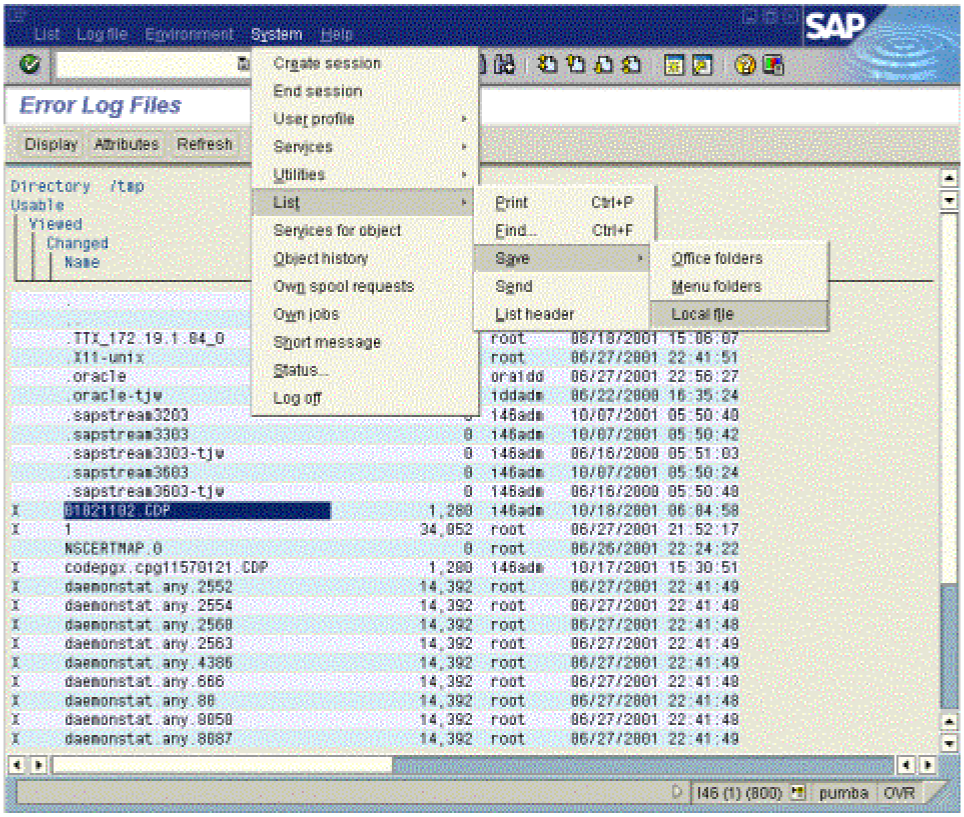

List Log file Environment System Help
Error Log Files
Display Attributes Refresh
146 (1) (800) pumba OVR

```
Directory  /tmp
Usable
 Viewed
  Changed
   Name                              Length      Creator  Last change

   .                                  1,024      root     10/18/2001 06:05:06
   ..                                 2,560      root     08/18/2001 15:06:07
   .TTX_172.19.1.04_0                     0      root     08/18/2001 15:06:07
   .X11-unix                            512      root     06/27/2001 22:41:51
   .oracle                              512      oraidd   06/27/2001 22:56:27
   .oracle-tjw                          512      iddadm   06/22/2000 16:35:24
   .sapstream3203                         0      146adm   10/07/2001 05:50:40
   .sapstream3303                         0      146adm   10/07/2001 05:50:42
   .sapstream3303-tjw                     0      146adm   06/16/2000 05:51:03
   .sapstream3603                         0      146adm   10/07/2001 05:50:24
   .sapstream3603-tjw                     0      146adm   06/16/2000 05:50:48
X  01021102.CDP                       1,280      146adm   10/18/2001 06:04:58
X  1                                 34,052      root     06/27/2001 21:52:17
   NSCERTMAP.0                            0      root     06/26/2001 22:24:22
X  codepgx.cpg11570121.CDP            1,280      146adm   10/17/2001 15:30:51
X  daemonstat.any.2552               14,392      root     06/27/2001 22:41:49
X  daemonstat.any.2554               14,392      root     06/27/2001 22:41:48
X  daemonstat.any.2560               14,392      root     06/27/2001 22:41:48
X  daemonstat.any.2563               14,392      root     06/27/2001 22:41:49
X  daemonstat.any.4386               14,392      root     06/27/2001 22:41:49
X  daemonstat.any.666                14,392      root     06/27/2001 22:41:48
X  daemonstat.any.80                 14,392      root     06/27/2001 22:41:48
X  daemonstat.any.8050               14,392      root     06/27/2001 22:41:48
X  daemonstat.any.8087               14,392      root     06/27/2001 22:41:49
```

2. Double-click the conversion table file to view its contents.

3. Select the List option from the System pull-down menu and select *Save to Local File*, as illustrated in the following window:



```
Directory  /tmp
Usable
 Yiewed
  Changed
   Name

      .
      ..
      .TTX_172.19.1.84_0                          root     08/18/2001 15:06:07
      .X11-unix                                   root     06/27/2001 22:41:51
      .oracle                                     oraldd   06/27/2001 22:56:27
      .oracle-tjw                                 lddadm   06/22/2000 16:35:24
      .sapstream3203                              146adm   10/07/2001 05:50:40
      .sapstream3303                          0   146adm   10/07/2001 05:50:42
      .sapstream3303-tjw                      0   146adm   06/16/2000 05:51:03
      .sapstream3603                          0   146adm   10/07/2001 05:50:24
      .sapstream3603-tjw                      0   146adm   06/16/2000 05:50:40
X     01021102.CDP                        1,280   146adm   10/18/2001 06:04:58
X     1                                  34,052   root     06/27/2001 21:52:17
      NSCERTMAP.0                             0   root     06/26/2001 22:24:22
X     codepgx.cpg11570121.CDP             1,280   146adm   10/17/2001 15:30:51
X     daemonstat.any.2552                14,392   root     06/27/2001 22:41:49
X     daemonstat.any.2554                14,392   root     06/27/2001 22:41:48
X     daemonstat.any.2560                14,392   root     06/27/2001 22:41:48
X     daemonstat.any.2563                14,392   root     06/27/2001 22:41:49
X     daemonstat.any.4386                14,392   root     06/27/2001 22:41:49
X     daemonstat.any.666                 14,392   root     06/27/2001 22:41:48
X     daemonstat.any.80                  14,392   root     06/27/2001 22:41:48
X     daemonstat.any.8050                14,392   root     06/27/2001 22:41:48
X     daemonstat.any.8087                14,392   root     06/27/2001 22:41:49
```

A dialog box displays prompting on a format for the transfer file.

4. Select unconverted format and click *Continue*.

5. Type a valid file name and click *Transfer*. This transfers the file to your local desktop.

6. Use any available editor on your desktop and remove the first three lines from the conversion table file. The following is an example of the text that should be removed:

```
Directory /tmp
Name: 01021102.CDP
----------------------------------------------
```

After the edits have been made, configure the server with the Adapter for SAP BW. Once the adapter is configured, the conversion table files must be moved to the server's EDAHOME ETC directory. Make sure both conversion table files are moved.

Extracting Data With SAP BW Queries

In this section:

Queries are the methods for extracting data from the InfoCube. A query must be created or must preexist for reporting to be performed. Queries form query cubes that are used in creating report data.

Imagine the InfoCube as a multi-dimensional cube. A subcube (query) is cut from the InfoCube by the selection of characteristics and Key Figures. In this way, the data of the InfoCube can be quickly targeted and evaluated. The more precisely the query is defined, the smaller the subcube and the quicker the query can be navigated and refreshed. The query evaluates the entire data set of the InfoCube. Selecting certain characteristics means that they can be more closely analyzed while others remain unspecified. The resulting Key Figures are aggregated across all characteristic values for the unspecified characteristics.

A default navigational state is also established in the query definition when you arrange the characteristics and Key Figures in the rows and columns of the query.

A query is defined in SAP BW using the Excel-based BEx Analyzer. Once a query has been defined and released, the OLAP processor requests the data from the query and presents the current view of the stored data.

Only the data that is currently requested is transferred from the InfoCube to the query. The OLAP processor builds the query from the InfoCube data and provides methods for navigating through the data in several dimensions. Since a query preselects information, the same InfoCube can yield dramatically different results depending on the query used to view its contents.

Defining New Business Explorer Queries

How to:

Select an InfoCube to Query

Create a Query

Filter a Query

You must define a SAP Business Explorer query before reporting from SAP BW InfoCube data. This query serves as a template for data extraction from the cube.

Note: The information that follows is based on SAP BW Business Explorer documentation. SAP BW BEx documentation is available from http://help.sap.com.

Procedure: How to Select an InfoCube to Query

1. The Business Information Warehouse must contain at least one InfoCube before you can define a new query. Start the Business Explorer Analyzer.
2. From the BEx toolbar, choose *Open*.

   You will see the selection screen for all existing workbooks.
3. Choose *Queries*.

   The selection screen displays all available queries.
4. Choose *New*.

   You will see the selection screen for all InfoCubes for which you can define a new query or queries.
5. Select the InfoCube that has the data on which the query should be based. To display technical names for InfoCubes, set the Technical name on/off icon to *On*.

   Tip: Since InfoCubes can have similar names for different elements, when building the query we recommend that you use technical names, rather than friendly names, as a reference. For related information, see *Understanding Field Names* on page 40-21.

   The available objects in the InfoCube you have selected are displayed as a directory tree in the left part of the screen.

Next, you will select the objects for the query and drag them to the appropriate boxes to build the query.

Procedure: How to Create a Query

The right area of the BEx screen contains selection boxes for the filter selection, the rows, the columns, and the free characteristics of the query.

Perform the following steps to create a query:

1. Click the plus or minus sign to the left of the dimension or Key Figures you want to query.

   The object list will expand and display a list of all the available Key Figures or characteristics.

2. Drag and drop characteristics and Key Figures from the InfoCube into the selection box of the query definition.

   These may be filters, rows, columns, and free characteristics.

Procedure: How to Filter a Query

You can filter queries in order to place restrictions on them. Filter selection restricts the entire query, meaning that all of the InfoCube data is aggregated with the filter selection. To select fields you want to use to filter the query, complete the following:

1. From the object list of the InfoCube, select the characteristics or the key figure upon which the query should be based.

   Note: Since they are used in definitions, fields selected as filters are not displayed in the Adapter for SAP BW metadata. They are used to screen the data and thus contain no information to be reported on. If you wish to screen data and report from it, see *Restricting Query Characteristics* on page 40-18 or *Restricting and Calculating Key Figures* on page 40-18.

2. Drag the object to the Filter box.

3. Right-click the object in the filter box. A dialog box opens showing the possible filter definitions for the object.

4. Select either a single member, a range of members, or a variable for the filter.

Restricting Query Characteristics

When defining a query, you may restrict characteristics to a single characteristic value, a value interval, a hierarchy node, or a characteristic value variable.

Procedure: How to Restrict Characteristics

1. Choose the characteristic from the InfoCube for which you want to select a value range.
2. Drag the characteristic into the appropriate selection box of the query definition (rows or free characteristics).
3. Select the characteristic you wish to restrict (or filter). Using the right mouse button, select *Restrict* from the Context menu.
4. Choose whether you want to restrict the characteristic to a single value, a value interval, or a hierarchy node.

   Tip: You can enter the characteristic values or hierarchy nodes you want to use, or you can display a list of all possible values by clicking the magnifying glass to the right of the input field.
5. Confirm your entries by clicking *OK*.

Restricting and Calculating Key Figures

How to:

Restrict Key Figures

Calculate Key Figures

Define a Formula

You can restrict Key Figures to characteristic values, characteristic value intervals, or hierarchy nodes. For example, a restricted key figure would be Sales revenue in 1st quarter.

You can also restrict the Key Figures of the InfoCube for the query definition, or, using a formula, you can calculate new Key Figures from the (basic) Key Figures:

- **Restricted Key Figures.** (Basic) Key Figures for the InfoCube that are restricted (filtered) by selecting one or more characteristics.
- **Calculated Key Figures.** Formulas that consist of (basic) Key Figures for the InfoCube and/or calculated Key Figures that have already been created.

Procedure: How to Restrict Key Figures

1. Drag a (basic) key figure into the key figure selection box. Alternatively, select the header of the selection box for rows or columns and, using the right mouse button, select *New Structure* from the Context menu.
2. Select the Structure directory, and, using the right mouse button, choose *New Selection* from the Context menu. The New Selection screen opens.
3. Enter a description of the restricted key figure in the text fields located in the upper part of the screen.
4. Underneath the text fields, on the left, is the directory of all the objects available in the InfoCube. Use the empty field on the right-hand side of the screen for the definition of the new selection.
5. Using drag and drop, choose a key figure from the InfoCube, and restrict it using a selection of one or more characteristic values.
6. Select *OK*. The newly restricted key figure is defined in the structure.

Procedure: How to Calculate Key Figures

1. Create a new structure in the rows or columns of the query definition by highlighting the row or column directory using the right mouse button and selecting *New Structure* in the Context menu.
2. Drag a (basic) key figure of the InfoCube into the directory of the new structure.
3. Select the Structure directory, and choose *New Formula*. The Formula Definition screen opens.
4. Enter a description of the formula in the text fields located in the upper part of the screen.

Note: The entry field for the formula is underneath the text fields. In the bottom left of the screen are all of the operands available for the formula definition. These are the Key Figures that you have already defined in the structure, and all of the formula variables in the Variables directory that have been created in the variable maintenance.

The functions available as operators are on the right-hand side of the screen. These are symbols for the basic arithmetic operations and directories with calculation functions such as percentage or trigonometric functions. To the right of the operators is a number block.

Procedure: How to Define a Formula

1. Choose the operands you want to use, and insert them in the entry field for the formula by double-clicking or by using drag and drop.
2. Choose the calculation functions you want to use by either clicking the symbols for the basic arithmetic operations, double-clicking to select the individual values, or dragging the entire key figure into the formula box.
3. Select the number values for the formula by clicking the number block.
4. Define your formula using the available operands and operators.

   If you want to use a variable that is not contained in the operands, you must create the variable first.
5. Check the formula definition for correctness by pressing the scale icon.
6. Enter the name of the formula column in the description box.
7. Select *OK*. The newly calculated key figure is defined in the structure.

Viewing Query Properties and Releasing for OLAP

- To view the properties of a query, click the Query Properties icon on the toolbar. The Query Properties dialog box opens.
- To release a query for OLAP, click the Query Properties icon on the toolbar and check *Release for OLE DB for OLAP* in the Query Properties dialog box.

This enables the query to be displayed as a QUERY_CUBE for reporting purposes. The query elements (hierarchy levels, measures, variable, and properties) will be mapped to corresponding OLAP elements to create a synonym.

Managing SAP BW Metadata

In this section:

SAP BW field names are generated from the Level Caption loaded in BW Master Data and obtained from the BW Query Cube. These captions provide friendly (person-readable) field names. However, the Adapter for SAP BW enables you to use either friendly or technical (invariant) field names when generating metadata. Technical field names are most useful for stable queries, while friendly field names are most useful for queries that may change frequently.

Note: You must generate an SAP BEx Query, release it for OLAP, and run the query *before* you can create a synonym. You will complete those tasks using the SAP Business Explorer, a tool that creates the SAP BW queries that iWay will report on. If you have not completed those tasks, see *Extracting Data With SAP BW Queries* on page 40-15, then return to generating metadata.

Understanding Field Names

How to:

Example:

The Adapter for SAP BW enables you to use either friendly or technical (invariant) field names when generating metadata. Use the technical field name approach when you are developing stable queries that are not often changed. Use the friendly field name approach when you are developing a rapid ad-hoc query, or when your query needs to be changed often.

Friendly names are generated from metadata element captions. Technical names are generated from the UNIQUE_NAME of the respective metadata elements: for example, Levels and Measures. The UNIQUE_NAME always contains the name of an object that is unambiguous within the context of the cube. Technical names are always SAP Language independent, and are less changeable across systems and releases. Since friendly field names can change when you migrate between SAP BW release levels or when you move reports between systems, technical field names are recommended in these circumstances.

For both friendly and technical field names, the names are upper case and special characters (for example, spaces and commas) are automatically replaced with underscores. Suffixes are sometimes added to provide field name uniqueness through the Master File.

Example: Friendly and Technical Field Names

Friendly Field Name

```
"Sales Volume"Title "Sales Volume"
```

Technical Field Name

```
"3IPY96CDK740I7444ILK15U1V" Title "Sales Volume"
```

Procedure: How to Set Synonym Properties

Once you have created the queries, and before you create synonyms, you must set synonym properties.

1. Launch the Web Console.
2. Click *Configure*, then click *Edit Files*.
3. Open and edit the Edasprof.prf server profile.

   At the end of the text in the file, add the following line:

```
ENGINE BWBAPI SET FIELDNAMES {ALL|MEASURES|LEVELS|PROPERTIES|VARIABLES} {TECHNICAL|FRIENDLY}
```

   You may select any combination of Measures, Levels, Properties, and Variables or choose ALL. Then, for the option(s) you have selected, you may choose to generate either a technical or a friendly field name. Your last combination of selections overrides any previous combination.

4. Navigate to the adapter configuration page. You will see all validly specified SAP BW connections.

Creating Synonyms

Synonyms define unique names (or aliases) for each SAP BW table or view that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for the adapter.

4. To display all available queries, leave the *Filter by Catalog and Cube* check box blank.

   To filter by Catalog or Cube name, click the check box and enter the name of the Catalog (InfoCube) or Cube (Query Cube) in the appropriate input box. You can use the wildcard character (*) as needed. For example: `0CCA*`

   For a query cube, enter the full catalog name, followed by a forward slash (/), followed by the query cube name. You can use the wildcard character (*) as needed. For example: `0CCA_C01/ZALL*`

5. Click *Select Candidates*. The Create Synonym for SAP BW (Step 2 of 2) pane opens. All queries that meet the specified criteria appear.

6. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.

7. If you have queries with identical names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll queries, assign the prefix HR to distinguish the synonyms for the human resources queries. Note that the resulting synonym name cannot exceed 64 characters.

If all queries have unique names, leave prefix and suffix fields blank.

8. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

9. Select the queries for which you wish to create synonyms:

- To select all queries in the list, select the check box to the left of the *Default Synonym Name* column heading.
- To select specific queries, select the corresponding check boxes.

10. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

11. Click *Create Synonym*.

Synonyms are created and added under the specified application directory.

A status window displays the message:

```
All Synonyms Created Successfully
```

12. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.

Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Mapping Metadata

Example:

Mapping Illustrations

Dimension: 0MATERIAL

Reference:

Mapping BW InfoCubes to OLE/DB Cubes

Syntax Conventions for Master Files

OLAP (OLE DB standard) uses the following terms for row sets: Hierarchies, Levels, Measures, Members, and Dimension Properties. The corresponding terms are mapped in BW and the Adapter for SAP BW.

It is a good practice to regenerate the adapter's metadata whenever data is loaded into the cube. Regeneration keeps data up-to-date and enables the adapter to accurately record the extent to which each dimension hierarchy expands.

Reference: Mapping BW InfoCubes to OLE/DB Cubes

Business Information Warehouse (BW) InfoCubes are similar to OLE/DB cubes. Both are the central metadata objects and containers for data used for reporting. InfoCubes contain two types of data: Key Figures and characteristics. For related information, see *Extracting Data With SAP BW Queries* on page 40-15. For examples of these mappings, see *Mapping Illustrations* on page 40-27.

OLAP	**BW**	**FOCUS**
Cube	Query	Master File.
Dimension	Characteristic	Internal use appears as comment in Master File (see above).
Hierarchy	Hierarchy	Dimension (A Adapter for SAP BW Dimension is a group of fields connected by means of the WITHIN keyword. The WITHIN keyword relates inner values to outer values for summary purposes.
Level	Level	Field (the WITHIN keyword points to the parent field).
Member	Characteristic Value	Field values.
Dimension Property (including special property Key)	Attribute	Field (only printable when the reference Dimension Hierarchy is used as a BY field).
No OLAP mapping	SAP BW variable	Field (specially mapped in a Master File) that takes optional or required input.
Measure	Key Figure (Measure)	Numeric fields upon which FOCUS verbs (PRINT/SUM) can perform operations.

Example: Mapping Illustrations

The following lines of syntax illustrate the makeup of the Adapter for SAP BW metadata. Each is listed with its corresponding definition:

```
FILENAME=WAREHOUS, SUFFIX=BWBAPI,$
```

This denotes the cube descriptor. The FILENAME=*value* is generated from the query cube name.

```
SEGNAME=WAREHOUS, SEGTYPE=S0,$
```

This is the segment name (used internally in FOCUS only). The SEGNAME=*value* is generated in the same way as FILENAME=*value*.

```
$ DIMENSION -MEASURES-
```

This is the beginning of the measures section of the Master File. The Fact Table is logically a dimension and is displayed accordingly in the Master File.

```
$ CARDINALITY=n
```

The cardinality of a dimension defines how many members can be retrieved for a report. If the Cardinality of a dimension is high, you should consider a WHERE test to restrict the returned values.

```
FIELD=STORE_INVOICE, ALIAS='STORE INVOICE', USAGE=D10.2, ACTUAL=D8.2,
  MISSING=OFF,$
```

This describes the measure. The first section in the Master File defines the fields that comprise the Fact Table for your cube.

```
$ DIMENSION: dimension_name
```

This is the Dimension section comment. The Adapter for SAP BW synonym logic puts dimension names in comments before descriptions of the corresponding hierarchies and increases the readability of the Master File. All columns that follow a dimension comment are members of the dimension.

Note: Multiple hierarchies for the same dimension display as separate dimension entries.

```
FIELD= PRODUCT_FAMILY, ALIAS='PRODUCT FAMILY', WITHIN= *PRODUCT,
  USAGE=A20, ACTUAL=A20, MISSING=OFF,$
```

This is the top level of a dimension hierarchy. Each subsequent level of the hierarchy will have a corresponding field name. Each level of the hierarchy has a following comment line with the level unique cardinality (count of members for the level).

Note: The WITHIN=*value* cannot exceed 66 characters. If the hierarchy unique name is longer than 65 characters (plus 1 character for *), Create Synonym generates a message. Although OLAP supports summary levels (the root level in the hierarchy), these are not displayed in the Master File.

```
FIELD=STORE_MANAGER, ALIAS='STORE MANAGER', USAGE=A20, ACTUAL=A20,
  MISSING=ON,$
```

This describes the Dimension property. Any fields listed in a dimension section of the Master File that do not have an assigned hierarchy (for example, no WITHIN), represent dimension properties. In this example, the property stores the store manager's name for the store in the rollup.

Reference: Syntax Conventions for Master Files

The following conventions must be followed in a Master File that is used with SAP BW:

- **MISSING=OFF is required.** There is no null data in a cube. Joins are not supported in convention with BW syntax. All fields in the Master File against a real cube should be explicitly set to MISSING=OFF to prevent any attempt to retrieve missing data, which would cause OLAP to return an error.
- **Quotes are required.** Any field or hierarchy name containing blanks must be enclosed in single quotation marks.
- **Hierarchy levels.** Hierarchy levels do not have clearly defined data types. As a result, the levels are always represented as alpha.
- **Property fields.** Can only be used if their associated dimension has been activated. For example, if you need to use BY <Property> you need to also have BY <dimension> NOPRINT in the same request. When you use "by dimension" in the request it will activate that dimension and all its properties. The NOPRINT option ensures that the field is not displayed in the resulting report. For related information, see *Reporting Rules* on page 40-31.
- **Special Property - Key.** BW dimensions have a default hierarchy noted by the name of the dimension appended with level_01. These default hierarchy fields have a special property called a Key that contains the BW internal reference code for the field. A Key may be printed for only the default hierarchy and may be used only when it is referenced. A Key field may also be used in WHERE or DEFINE statements, providing the dimension field is referenced. For related information, see *Reporting Rules* on page 40-31.

Example: Dimension: 0MATERIAL

The following example shows what can be expected when selecting the fields in a report. It is important to remember that two fields are returned for the same data value.

```
Field:0MATERIAL_LEVEL01
Field:0MATERIAL_KEY
```

as displayed in a report:

```
0MATERIAL_LEVEL_01  0MATERIAL_KEY  TOTAL_SALES
Fitdrink 2000(CAN)  R100032           $1000.00
```

Both fields represent the *same* characteristic value.

0MATERIAL_LEVEL_01 is the dimension level containing the captions for all members of the dimension.

0MATERIAL_KEY contains the SAP BW technical name for the members of the dimension.

Variable Types

SAP BW supports four types of variables:

- **Members.** Can be displayed in a separate TABLE request. When used in a TABLE query, a search criterion must be supplied if the variable type is mandatory (and there is no default value). The search criterion must render a set of variable values that correspond to the variable, which may be single value, interval, or complex (for complex there is no restriction on combinations). Each search predicate (an AND logical expression) can reference only one variable and no other cube elements.
- **Nodes.** Can be displayed in a separate TABLE request. Node type variables have a single value selection type.
- **Hierarchies.** Can be displayed in a separate TABLE request. Hierarchy type variables have a single value selection type.
- **Numeric.** Cannot be displayed.

For related information, see *Reporting With Variables* on page 40-33.

Overview of SAP BW Reporting Concepts

In this section:

The SAP Business Information Warehouse (BW) is a data warehouse solution constructed with specialized content for reporting and analysis. Data is extracted from source systems, or InfoSources. After being scrubbed (that is, revised in programmatic ways for easier reporting), the data is loaded into InfoObjects (special reporting tables) that are combined in business-relevant ways to become InfoCubes (multi-dimensional analysis structures).

InfoCubes contain Key Figures (numeric value fields) and characteristics that describe the Key Figures (organization and classification information).

Sample Key Figures are:

- Value, which includes costs, sales, and sales deductions.
- Quantity, which includes number of employees, sales quantity, and amount of billings posted.

Sample characteristics are SAP BW organizational units, such as:

- Controlling area
- Company code
- Business area
- Division

Characteristics form dimensions that enable you to analyze the Key Figures from multiple perspectives.

Reporting Rules

Key Figures. Key Figures that are pre-aggregated in SAP BW must use the SUM verb (most InfoCube data). Non aggregated Key Figures can use the detail verb PRINT. (See your data administrator for details.)

Properties. Properties must reference their parent dimension levels. (For related information, see *Syntax Conventions for Master Files* on page 40-28).

Variables. Variables must supply the appropriate syntax for the type (see *Reporting With Variables* on page 40-33).

Hierarchies. A single hierarchy from the same dimension can be represented on the report.

Cardinality. Because SAP BW is a multi-dimensional database, each field represents a multiplicative value for the total number of cells returned. (If the Cardinality in the Master File for the dimension is very large, you may want to use a WHERE test to limit the result.) For example:

```
TABLE FILE BW
SUM NET_SALES GROSS_SALES TOTAL_SALES (MEASURES CARDINALITY 3
BY COMPANY_CODE (DIMENSION 1 CARDINALITY 100)
BY REGION (DIMENSION 2 CARDINALITY 6)
END
```

is calculated as follows

(100 X 6 X 3 = 1,800 possible cells)

where:

Dimension 1 = 100 members X
Dimension 2 = 6 members X
3 Measures = 1,800 possible cells

Adding the following line of code

```
BY SALES_REP (DIMENSION 3 CARDINALITY 50)
```

changes the calculation to

(100 (D1) X 6 (D2) X 50 (D3) X 3 (measures) = 90,000 (possible cells)

General Tips for Reporting

When creating a report request:

- Do not use PRINT against Key fields inside OLAP cubes. Calculated and pre-aggregated fields need to be referenced by the equivalent of the verb that generated them. Reference count fields using the COUNT command, rather than PRINT or SUM. Reference SUM fields (aggregations) with the SUM command.

  When reporting from Operational Data Store (ODS) or InfoSet Queries (ODS joins), depending on the summarization setting of the key fields, you may use the PRINT verb. When using PRINT against ODS objects, the BY fields (not to be confused with Key Figures Fields) determine the summarization level of unique values.

- To maximize reporting efficiency, always sort in the same direction as the hierarchy for the cube.

For example, if the hierarchy for a dimension is

```
Country > State_Province > City
```

sorting by City first, then State_Province results in an inefficient request. Also, because of the hierarchical structure of OLAP, you retrieve only one BY field for each SUM or PRINT field in the report.

Using Columnar Reports to Slice Cubes

Example:

Displaying Single Dimensions in a Slice Report

Displaying Multiple Dimensions in a Slice Report

Reference:

Valid and Invalid Slices-Cube Slicing Logic

If you run a standard TABLE or FML report against a cube, you are running what is known as a Slice report. This two-dimensional report resembles a slice of a larger multi-dimensional cube. A Slice report flattens out the multi-dimensional structure of a cube to show information from one angle.

To create a valid Slice report, you must understand the cube's data structure and the logic and assumptions that the Adapter for SAP BW uses when slicing a cube.

Reference: Valid and Invalid Slices-Cube Slicing Logic

The Adapter for SAP BW has built-in logic that enables it to determine the rows and cells to extract from rollups in order to deliver Slice reports. TABLE uses the lowest level BY phrase for each dimension to determine at which rollup to locate the lookup for aggregations. It then rolls the result up from there, performing aggregations on the data as needed to ensure the consistency of the results.

If you do not specify a BY field in your TABLE request, you may receive invalid data or a message. This happens because without a BY field to determine the level of rollup at which to perform the data lookup, TABLE does not know where to source the result.

Example: Displaying Single Dimensions in a Slice Report

```
TABLE FILE MYCUBE
SUM STORE_COST STORE_SALES
BY COUNTRY_LEVEL_01 ON TABLE COLUMN_TOTAL
END
```

This report displays aggregate data sorted against one dimension in the source. The standard measures such as STORE_COST and STORE_SALES are referenced with SUM. Column totals are requested in the report.

Example: Displaying Multiple Dimensions in a Slice Report

```
TABLE FILE MYCUBE
SUM STORE_COST STORE_SALES
PRINT PROFIT
BY COUNTRY_LEVEL_01 BY STATE_PROVINCE_LEVEL BY CITY_LEVEL_01
END
```

This report displays the same data as the previous one, except that it sorts against both dimension hierarchies in the cube. The order of the sort fields follows the logical order of the dimension hierarchy.

Reporting With Variables

A BW query uses variables to supply values at execution time. A variable can have an Entry_Type of either Mandatory/Required or Optional. All mandatory variable values must be supplied in the TABLE or SQL request. A WHERE or IF statement in the request provides the runtime value(s). For example, if the variable COMPANY_CODE is mandatory, it will supply values for company code 430 at runtime:

```
WHERE COMPANY_CODE_N EQ '430'
```

All variables passed to BW have a format of 124 alpha bytes. The synonym's Master File breaks variables into two parts: variable_C and variable_N.

- The caption (_C) is the first 60 bytes and is an internal reference for BW.

- The name (_N) is the last 64 bytes in which the actual key value will be sent to the query.

In the previous example, the name (_N) supplies the variable value. You can use either the caption or the name for screening, but you must supply appropriate values for each:

```
company_code_n : WHERE COMPANY_CODE_N EQ 430
company_code_c : WHERE COMPANY_CODE_N EQ 'IDES USA'
```

Tip: If the associated dimension of this variable is available in the query, do not build the selection on that field, but, rather, use the name. In other words, WHERE COMPANY_CODE_LEVEL_01 EQ 430 *will not* populate the variable, while COMPANY_CODE_N will. Additional non-variable selections are available, but they are less efficient. The non-variable conditions are applied after the BW answer set has been extracted.

Variables cannot be displayed in a request. They should be used only for applying selection criteria. You can display the dimensions or measures with which they are associated. Following are annotated examples of a valid request and an invalid request.

Valid request:

`SUM ACCUMULATE_BALANCE`	Measure ACCUMULATE_BALANCE (SUM verb must be used.)
`   BY COMPANY_CODE_LEVEL_01`	Dimension level COMPANY_CODE_LEVEL_01
`   BY COUNTRY_KEY_`	Property COUNTRY_KEY_ belongs to COMPANY_CODE_LEVEL_01
`WHERE COMPANY_CODE_N EQ '430'`	Variable

Invalid request:

`SUM ACCUMULATE_BALANCE`	Measure
`   BY COMPANY_CODE_N`	Variable (Variable cannot be printed, COMPANY_CODE_LEVEL_01 should be used.)
`   BY COUNTRY_KEY_`	Property (Property belongs to a dimension level and both must be used in the request.)
`WHERE COMPANY_CODE_N EQ '430'`	Variable

You cannot display a variable as a BY field. Dimension properties must be used in combination with their reference dimension. Variables can have a selection type of single, interval, or complex.

- Single type variables must have only one value supplied in the selection condition.
- Interval variables can have a single value or a range of values.
- Complex variables have no restrictions.

To supply a range, use two WHERE statements or combine them into one statement using AND. For example, this report needs to specify two years:

```
WHERE FISCAL_YEAR_N EQ "2001"
        OR FISCAL_YEAR_N EQ "2002"
WHERE FISCAL_YEAR_N GE "2001"
WHERE FISCAL_YEAR_N LE "2002"
WHERE FISCAL_YEAR_N GE "2001"
      AND FISCAL_YEAR_N LE "2002"
WHERE FISCAL_YEAR_N IN ("2001", "2002")
WHERE FISCAL_YEAR_N FROM "2001" TO "2002"
```

Selections that use an OR connector, like the first sample above, can only test against the same field or related hierarchy. You cannot combine variable selection and non-variable selection in the same WHERE statement (screening predicate).

Working With Hierarchies

In SAP BW release 3.0A and earlier, one characteristic of the query can be expanded in a hierarchical display.

Reference: Creating Dynamic Hierarchies

The Adapter for SAP BW generates each level of a hierarchy with a field name corresponding to each level. For a dimension called Business Number and a hierarchy named States, level 1 will be States and level 2 will be the Business Number. You can create a report referencing all the fields of a hierarchy at once, or create it as an OLAP trigger report, referencing the top level of the hierarchy and enabling interactive drill down.

In SAP BW 3.0B and higher, multiple characteristics of the query can be expanded in a hierarchical display.

Creating Dynamic Dimensions

SAP BW, you can:

- Create dynamic dimensions by using the Create New Field (Define) function.
- Use hierarchy levels and dimensions that are not contained in the query cube, but are defined or calculated at runtime.

Example: Creating a Dimension

The following request creates a dimension called Daily Sales, which concatenates the company name, the current date (picked up from an the &DATE variable), and the Sales figures for that date.

```
DEFINE Daily Sales/A124= company|&date|'Sales'
END

TABLE FILE MYQUERY
SUM Retail_Amount
BY Daily Sales
END
```

The output is:

```
Daily Sales                       Retail_Amount
ACME   11/15/2002                        10,000
SMCE   11/15/2002                        15,000
```

BW Explorer

In this section:

Using BW Explorer

How to:

Switch the BW System to an R/3 System

Use the BW Explorer

Create Synonyms for BW Database Tables

WebFOCUS for SAP BW utilizes BAPIs to access to BW data. Under certain circumstances, it might be suitable to use the R/3 adapter for BW, which actually creates dynamic ABAP/4 programs to read the data.

Procedure: How to Switch the BW System to an R/3 System

To switch the BW system to an R/3 system:

1. Follow the steps defined in *Preparing the SAP R/3 Environment* of the *Adapter for SAP R/3* manual to create the Development Class or Package and the Function Group in the BW system and activate them.

   Note: The SAP BW user ID should have the authorizations listed in the *Adapter for SAP R/3* chapter, in addition to the authorizations listed in this chapter.

2. Follow the steps defined under the reference topic *Declaring SAP BW Connection Attributes From the Web Console*. Make sure to enter the function group name created in the previous step in the FPOOL field.

3. Once the Adapter for BW is configured, expand the Adapters tree on the left pane and navigate to the newly created SAP BW system under the Configured - SAP BW tree branch. Right-click the connection name and select *Properties* from the pop-up menu.

   The connection parameters are shown in the left pane. Scroll down to see the *Switch BW System to SAP* check box next to the Configure button.

4. Select the check box and click *Configure*.

   The SAP BW system has been moved under the Configured - SAP R/3 branch.

5. Click the system name.

   The following menu items are shown:

Initialize BW Explorer	Creates synonyms for the BAPIs that BW Explorer uses.
Start BW Explorer	Starts the BW Explorer.
Test	Runs a test procedure. The test procedure looks for an SAP R/3 table. The table does not exist in a BW system. Hence, the output would be 0 records.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Delete	Deletes the synonym.

6. Select *Properties* from the pop-up menu.

7. Scroll down and click *Install SAP Components* to install the necessary function modules into BW system.

   For detailed information, refer to the *Adapter for SAP R/3* topic in this manual.

8. Click the system name to open the pop-up menu.
9. Select *Initialize BW Explorer.*

   This might take a few minutes. Once finished, a message similar to the following will appear:

```
 0  TRANSACTIONS:   0   TOTAL =   4   ACCEPTED=    4   REJECTED=    0
      SEGMENTS:         INPUT =   4   UPDATED=     0   DELETED=     0
```

Using BW Explorer

Before using the BW Explorer, the connection for BW System should be switched to R/3. Refer to the steps defined under *Switch the BW System to an R/3 System* on page 40-37.

Procedure: How to Use the BW Explorer

To use the BW Explorer:

1. Launch the Web Console.
2. Select *Data Adapters*.
3. Expand the SAP R/3 tree under the Adapters - Configured tree branch.
4. Click the connection name and select *Start BW Explorer* from the pop-up menu.

BW Explorer uses a set of BAPIs and displays the information as FOCUS reports.

- **List of Catalogs.** This report uses the BAPI_MDPROVIDER_GET_CATALOGS to list all the InfoProviders.
- **List of Cubes.** This report is a drill-down from the report List of Catalogs. It uses the BAPI_MDPROVIDER_GET_CUBES to list all the query cubes for the selected InfoProvider.

Drilling down from List of Cubes brings the report What do you want to Explore. Using this report, it is possible to drill-down to:

- **Dimensions/Hierarchies/Levels/Members.** The first report shows the dimensions of the query. Further drill-down displays the hierarchies for the selected dimension. Drill-down from the hierarchies shows the levels of the selected hierarchy and finally the drill-down form the hierarchy level displays the individual values for the selected hierarchy level.
- **Measures.** Displays information about the Measures (Key Figures) of the query.
- **Properties.** Shows the properties (attributes) associated with each dimension of the query.
- **Variables.** Shows the variables of the query, if any exist. If the report returns no records, it indicates that there are no variables assigned to the query.

Procedure: How to Create Synonyms for BW Database Tables

Note: It is possible to read the database tables from BW R/3. This method should be used only for master data reporting against very large volume of data, but the synonym would have to be created manually. The syntax is provided at the end of this section.

To create the synonym, the name of the BW database table should be determined beforehand. To determine the underlying database table for the InfoObject:

1. Logon to BW using the SAP front-end.
2. Launch the transaction RSA1. (Administrator Workbench: Modeling)

   Make sure that InfoObject is selected on the left pane. Find the InfoObject that you wish to access. To search, click the button with the binocular picture, and type the name of the InfoObject. Once the search result is displayed, double-click on the name of the InfoObject to navigate in the main window.
3. On the main window, double-click the name of the InfoProvider.
4. Select the Master data/texts tab. You should see the master data table under *Master Data Tables*. For the syntax below, the table name /BI0/PMATERIAL is used as an example.
5. Start the server using edastart -t. The syntax for creating the synonym is as follows:

   ```
   CREATE SYNONYM <app.directory>/<tablename> FOR <tablename> TABLE
   DBMS SQLSAP AT <systemname>
   END
   ```

   Example:

   ```
   CREATE SYNONYM baseapp//BI0/PMATERIAL FOR /BI0/PMATERIAL TABLE
   DBMS SQLSAP AT <systemname>
   END
   ```

Customization Settings

The following topics describe advanced set commands that allow you to customize your Adapter for SAP BW.

Tip: You can change several of these setting from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu.

Syntax: How to Specify Field Names as Friendly or Technical

```
ENGINE BWBAPI SET FIELDNAMES fieldtype {TECHNICAL|FRIENDLY}
```

where:

fieldtype

Is the type of Master File field you wish to affect. Possible values are ALL, MEASURES, LEVELS, PROPERTIES, or VARIABLES.

TECHNICAL

Indicates that the adapter will generate Master File field names from the given metadata element's technical name.

In some cases, it may be necessary to use the technical field names when generating synonyms to ensure field name uniqueness.

`FRIENDLY`

Indicates that the adapter will generate Master File field names for the given multidimensional metadata element from its *caption*. FRIENDLY is the default value.

Depending on the field type, additional keywords such as "LEVEL_01", "KEY", and so on may be appended to the end of the friendly field name.

Syntax: How to Specify Empty Report Cell Settings

```
ENGINE BWBAPI SET EMPTY {ON|OFF}
```

where:

`ON`

Indicates that empty cells are included in a report and all data is shown whether it is associated with a dimension member or not.

`OFF`

Indicates that empty cells are excluded from a report (if measures are referenced) and dimension data is not shown if no fact data is associated with the dimension member (if measures are not referenced). OFF is the default value.

Tip: You can change this setting from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Specify Member Sampling Settings

During the synonym creation, member sampling is used to create Master File fields for dimension levels that precisely fit dimension member captions.

```
ENGINE BWBAPI SET MEMBER_SAMPLING {ON|OFF}
```

where:

`ON`

Indicates that the adapter will select all members of a hierarchy level to determine the optimum length for the field, and also the cardinality. The default value is ON.

Note: Depending on the data volume, this may take a long time. If synonym creation takes too long, you may wish to turn the MEMBER _SAMPLING OFF.

`OFF`

Indicates that all field lengths for the hierarchy levels are set to alphanumeric 60 and the cardinality information will be missing.

Tip: You can change this setting from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Specify Block Size

SAP recommends fetching large row and cell sets by portions in order to avoid crashes in the SAP server. The BLOCK_SIZE setting sets the size of a portion.

```
ENGINE BWBAPI SET BLOCK_SIZE n
```

where:

n

Sets the size of a portion. 0 is the default value, which means all members of the result set are returned in one portion.

The value of this setting should not be changed unless there is a timeout. Contact Information Builders high-level tech support for further information on how to use this option.

Syntax: How to Specify Property Restrictions

The ability to query on attribute values in the MEMBERS row set was made available with SAP BW 3.0B. (For details, see the SAP OSS Note 609314.)

```
ENGINE BWBAPI SET PROPERTY_RESTRICTIONS {ON|OFF}
```

where:

`ON`

Indicates that any tests on a dimension attribute will be sent to SAP.

It is recommended that you set PROPERTY_RESTRICTIONS to ON if your SAP BW release and patch level support the specifying restrictions on user-defined properties as selection tables. (Contact SAP support for exact list of releases and patch levels.)

`OFF`

Indicates that the filtering will be done on the WebFOCUS server. OFF is the default value.

Tip: You can change this setting from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Specify Unassigned Member Settings

```
ENGINE BWBAPI SET INCLUDE_UNASSIGNED {OFF|ACTIVE|ALL}
```

where:

`OFF`

Does not include unassigned members belonging to an active hierarchy in a target report. OFF is the default value.

`ALL`

Includes all unassigned members belonging to an active hierarchy in a target report.

`ACTIVE`

Includes unassigned members belonging to an active hierarchy in which the lowest level is referenced. Individual unassigned members are listed at the lowest level of the hierarchy.

If the query does not include the lowest level, the summary for the unassigned nodes will not be shown unless there is a specific WHERE/IF statement at the lowest level.

Tip: You can change this setting from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Support for BEx Structures

Structures consist of a group of characteristics and/or key figures. They can be saved and reused in other queries. For example, if plan/actual comparisons are being used extensively, it is possible to save a key figure group that consists of 10 key figures arranged for plan/actual comparison. To be able to use the same set of key figures in a different query, instead of selecting 10 key figures individually, adding the structure to the BEx query automatically brings those key figures into the query.

Note that SAP supports a maximum of two structures in a BEx query and only one of the structures can be of type Key Figure.

Syntax: How to Include Structures as Dimensions in a Master File

```
ENGINE BWBAPI SET STRUCTURE_DIMENSIONS {ON|OFF}
```

where:

`ON`

Includes structures as dimensions in a Master File.

`OFF`

Does not include structures as dimensions in a Master File. OFF is the default value.

Tip: You can change this setting from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Example: Representing Structures as Dimensions

You can represent structures as dimensions or create a cartesian product of the fields in the structures. Consider a BEx query that uses 2 structures:

- One structure is called “Curr. Year vs. Prev. year Cumulated” and it includes two time dimensions: Current Year and Prev.Year.
- The second structure is called “Orders-Sales-Returns” and it includes four key figures: Incoming Orders, Open Orders, Sales Volume, and Returns.

If the STRUCTURE_DIMENSIONS is set to OFF, the created Master File includes 8 fields representing the cartesian product of the two structures. All of these fields are be placed under the MEASURES section of the Master File and treated as separate key figures. The Master File looks something like this:

```
FILENAME=BW31C/Z2STRUCTURES_1KF_ROW_1COL, SUFFIX=BWBAPI  , $
  SEGMENT=Z2STRUCT, SEGTYPE=S0, $
$  -MEASURES-
    FIELDNAME=CURRENT_YEAR_CUMULATED_INCOMING_ORDERS,
ALIAS=3466HXATG6RG1BG9GSTCN65IR3RV6E, USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Current year cumulated Incoming Orders', $
    FIELDNAME=CURRENT_YEAR_CUMULATED_OPEN_ORDERS,
ALIAS=3466HXATG6RG1BG9GSTCN65IR3RV6E, USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Current year cumulated Open orders', $
    FIELDNAME=CURRENT_YEAR_CUMULATED_SALES_VOLUME,
ALIAS=3466HXATG6RG1BG9GSTCN65IR3RV6E, USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Current year cumulated Sales Volume', $
    FIELDNAME=CURRENT_YEAR_CUMULATED_RETURNS,
ALIAS=3466HXATG6RG1BG9GSTCN65IR3RV6E, USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Current year cumulated Returns', $
    FIELDNAME=PREV__YEAR_CUMULATED_INCOMING_ORDERS,
ALIAS=36AWN99ETKI46HMTOGIVHHSOJ3RV6E, USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Prev. year cumulated Incoming Orders', $
    FIELDNAME=PREV__YEAR_CUMULATED_OPEN_ORDERS,
ALIAS=36AWN99ETKI46HMTOGIVHHSOJ3RV6E, USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Prev. year cumulated Open orders', $
    FIELDNAME=PREV__YEAR_CUMULATED_SALES_VOLUME,
ALIAS=36AWN99ETKI46HMTOGIVHHSOJ3RV6E, USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Prev. year cumulated Sales Volume', $
    FIELDNAME=PREV__YEAR_CUMULATED_RETURNS,
ALIAS=36AWN99ETKI46HMTOGIVHHSOJ3RV6E, USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Prev. year cumulated Returns', $
$  DIMENSION 0DISTR_CHAN (Distribution Channel):
$  HIERARCHY  (Distribution Channel):
...
...
```

The following is a sample WebFOCUS request against this Master File:

```
TABLE FILE Z2STRUCTURES_1KF_ROW_1COL
SUM CURRENT_YEAR_CUMULATED_INCOMING_ORDERS
    CURRENT_YEAR_CUMULATED_OPEN_ORDERS
    CURRENT_YEAR_CUMULATED_SALES_VOLUME
    CURRENT_YEAR_CUMULATED_RETURNS
    PREV__YEAR_CUMULATED_INCOMING_ORDERS
    PREV__YEAR_CUMULATED_OPEN_ORDERS
    PREV__YEAR_CUMULATED_SALES_VOLUME
    PREV__YEAR_CUMULATED_RETURNS
BY 0DISTR_CHAN_KEY
END
```

If the STRUCTURE_DIMENSIONS is set to ON, the created Master File includes four key figures placed under the MEASURES section of the Master File and the “Curr. Year vs. prev. year Cumulated” structure are treated as a dimension. The Master File looks something like this:

```
FILENAME=BW31C/Z2STRUCTURES_1KF_ROW_1COL_STON, SUFFIX=BWBAPI  , $
  SEGMENT=Z2STRUCT, SEGTYPE=S0, $
$  -MEASURES-
    FIELDNAME=INCOMING_ORDERS, ALIAS=[3RV6ENA572BRXFKFCO2M2ZUWS],
USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Incoming Orders', $
    FIELDNAME=OPEN_ORDERS, ALIAS=[3RV6ENHTQ0XHG23VII4YD1TMK],
USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Open orders', $
    FIELDNAME=SALES_VOLUME, ALIAS=[3RV6ENPI8ZJ6YONBOC7AN3SCC],
USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Sales Volume', $
    FIELDNAME=RETURNS, ALIAS=[3RV6ENX6RY4WHB6RU69MX5R24], USAGE=D18.2,
ACTUAL=D8, MISSING=ON,
      TITLE='Returns', $
$  DIMENSION 0DISTR_CHAN (Distribution Channel):
$  HIERARCHY  (Distribution Channel):
...
...
$  DIMENSION EAZ5PAJIWG37DI674JZI2C3UB (Curr. year vs. prev. year
(cumulated to last period)):
$  HIERARCHY  (Curr. year vs. prev. year (cumulated to last period)):
    FIELDNAME=CURR__YEAR_VS__PREV__YEAR__CUMULATED_TO_LAST_PERIOD_,
USAGE=A22, ACTUAL=A22, MISSING=ON,
      TITLE='Curr. year vs. prev. year (cumulated to last period)',
      WITHIN='*[EAZ5PAJIWG37DI674JZI2C3UB]', $
$  Cardinality: 2
$  DIMENSION EAZ5PAJIWG37DI674JZI2C3UB, PROPERTIES:
    FIELDNAME=EAZ5PAJIWG37DI674JZI2C3UB_KEY, ALIAS=MEMBER_NAME,
USAGE=A64, ACTUAL=A64, MISSING=ON,
      TITLE='Curr. year vs. prev. year (cumulated to last period) Key', $
```

The following is sample WebFOCUS request against this Master File:

```
TABLE FILE Z2STRUCTURES_1KF_ROW_1COL_STON
PRINT INCOMING_ORDERS
    OPEN_ORDERS
    SALES_VOLUME
    RETURNS
BY CURR__YEAR_VS__PREV__YEAR__CUMULATED_TO_LAST_PERIOD_
END
```

Note: When the STRUCTURE_DIMENSIONS is set to ON, you must use the key figures with the verb PRINT rather than SUM.

Important: If the BEx query includes only two structures and no dimensions as free characteristics or no dimensions in rows and columns, you should set STRUCTURE_DIMENSIONS to ON. If STRUCTURE_DIMENSIONS is set to OFF, the created synonym will have only measures and no dimensions. Any WebFOCUS requests created against such a synonym would produce the following messages:

```
(FOC11205) ERROR IN MASTER FILE DESCRIPTION:
```

```
(FOC11229) MDX SEGMENT DESCRIPTION CONTAINS NO HIERARCHIES
```

Producing SAP BW Requests

Example:

The following requests and output are examples of the capabilities of the Adapter for SAP BW using standard ANSI SQL code. These examples require that CREATE SYNONYM has been performed on the relevant tables from the Web Console. You can use the Web Console to navigate the SAP application hierarchy by searching the relevant tables or by doing a simple search by name.

Example: Retrieving Values From SAP BW

The following syntax retrieves values from an SAP BW database:

```
SELECT SUM(DISBURSEMENT),ADDRESS1_LEVEL_01 FROM ZPAY4
GROUP BY ADDRESS1_LEVEL_01
ORDER BY ADDRESS1_LEVEL_01;
```

The output is:

address1 Level 01	**disbursement**
1106 BROADWAY	132,242.33
11607 PACIFIC BLVD	9,174.96
11607 TREMONT AVENUE	14,849.56
1200 RIVERDALE	9,983.99
187 WEST 39TH ST	9,209.97
2601 8TH AVENUE	26,274.65
31-18 SOUTHERN BLVD	1,165.64
719 SEVENTH STREET	10,800.00
DRAWER 4, COOPER STA	21,406.32
P.O. BOX 1206	53,636.74
P.O. BOX 17616	6,703.90

Example: Selecting Items From a Hierarchy

The following syntax selects items from a hierarchy in MARA:

```
SELECT SUM(SALES_VOLUME),SUM(INCOMING_ORDERS),
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02 FROM ZBIGQ
GROUP BY PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02
ORDER BY PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02
```

The output is:

Product Hierarchy for material MARA Level 01	**Product Hierarchy for material MARA Level 02**	**Sales Volume**	**Incoming Orders**
Foods	Bakery product; configured to order	.00	400.00
Hardware	PCs	85,159,860.98	86,754,997.71
	Printer	.00	60,011.10
Lighting	Bulbs	62,235,898.00	62,674,320.00
Machines	Pumps	44,328,485.55	46,231,307.55
Paints	Gloss paints	.00	125.03
Services	Maintenance	291,562.54	485,315.83
Vehicles	Cars	155,640.00	155,640.00
	Motorcycles	41,761,619.76	42,494,811.00

Example: Selecting a Complete Hierarchy Path

The following syntax retrieves a complete hierarchy path from MARA:

```
SELECT SUM(SALES_VOLUME),SUM(INCOMING_ORDERS),
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_03,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_04
FROM ZBIGQ
GROUP BY
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_03,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_04
ORDER BY
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_03,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_04;
```

The output is:

Product Hierarchy for material MARA Level 01	**Product Hierarchy for material MARA Level 02**	**Product Hierarchy for material MARA Level 03**	**Product Hierarchy for material MARA Level 04**	**Sales Volume**	**Incoming Orders**
Foods	Bakery product; configured to order			.00	400.00
Hardware	PCs	Drive	Harddisk 1080 MB / SCSI-2-Fast	4,234,580.64	4,094,612.64
			Harddisk 2113 MB / ATA-2	5,216,171.71	5,216,171.71
			Harddisk 2149 MB / SCSI-2-Fast	4,727,005.36	4,713,724.40

Product Hierarchy for material MARA Level 01	**Product Hierarchy for material MARA Level 02**	**Product Hierarchy for material MARA Level 03**	**Product Hierarchy for material MARA Level 04**	**Sales Volume**	**Incoming Orders**
			Harddisk 4294 MB / SCSI-2-Fast	5,132,812.80	5,132,812.80
		Input device	Professional keyboard - MAXITEC Model	720,022.65	714,750.89
			Professional keyboard - NATURAL Model	813,765.24	788,838.81
			Professional keyboard - PROFITEC Model	576,079.36	576,079.36
			Standard Keyboard - EURO Model	620,294.21	600,911.21
			Standard Keyboard - EURO-Special Model	633,094.19	631,433.36
		Memory	SIM-Module 16M x 32, 70 ns	808,663.10	808,663.10
			SIM-Module 4M x 36, 70 ns	348,434.93	337,138.13
			SIM-Module 8M x 32, PS/2-72 Pin EDO-RAM	562,619.24	561,208.39

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
			SIM-Module 8M x 36, 70 ns	373,794.70	373,794.70
		Monitor	Flatscreen LE 50 P	1,116,349.10	1,155,586.30
			Flatscreen LE 64P	1,749,563.60	1,749,563.60
			Flatscreen MS 1460 P	2,525,626.85	2,704,385.95
			Flatscreen MS 1575P	2,375,663.66	2,469,023.66
			Flatscreen MS 1585	3,365,928.24	3,487,309.32
			Flatscreen MS 1775P	3,809,366.74	4,074,466.82
			Flatscreen MS 1785P	4,722,113.10	4,722,113.10
			Jotachi SN4000	4,396,902.60	4,572,792.60
			Jotachi SN4500	2,473,317.57	2,647,587.95
			Jotachi SN5000	2,949,233.80	2,949,233.80
			MAG DX 15F/Fe	2,437,207.85	2,521,555.85
			MAG DX 17F	2,567,506.69	2,655,601.89
			MAG PA/DX 175	2,976,662.21	3,181,471.75

Product Hierarchy for material MARA Level 01	**Product Hierarchy for material MARA Level 02**	**Product Hierarchy for material MARA Level 03**	**Product Hierarchy for material MARA Level 04**	**Sales Volume**	**Incoming Orders**
			PAQ MONITOR, 17", Color	683,666.40	685,463.40
			PAQ Monitor, 20", Color	1,371,234.90	1,371,234.90
			SEC Multisync XV 17	3,557,594.77	3,690,080.79
			SEC Multisync XV15	2,961,072.00	2,961,072.00
			Sunny Extreme	3,215,185.42	3,339,353.42
			Sunny Tetral3	2,996,901.84	2,996,901.84
			Sunny Xa1	2,830,258.78	3,026,866.52
		PC ensemble	Maxitec-R 3100 Personal Computer	3,616,021.08	3,600,521.08
			Maxitec-R 375 personal computer	194,520.00	194,520.00
		Processor	Motherboard 3100	.00	18,558.62
			Processor 100 MHz	401,608.95	401,608.95
			Processor 133 MHz	15,949.50	11,574.50
			Processor 166 MHz	1,083,067.20	1,016,409.60
	Printer	Laser printer	High Speed Printer	.00	60,011.10

Product Hierarchy for material MARA Level 01	**Product Hierarchy for material MARA Level 02**	**Product Hierarchy for material MARA Level 03**	**Product Hierarchy for material MARA Level 04**	**Sales Volume**	**Incoming Orders**
Lighting	Bulbs	Light Bulb 40 Watt clear 220/235V		15,351,358.00	15,351,358.00
		Light Bulb 40 Watt frosted 220/235V		5,432,920.00	5,256,060.00
		Light Bulb 40 Watt red 220/235V		2,976,965.00	3,197,425.00
		Light Bulb 40 Watt yellow 220/235V		3,032,064.00	2,917,008.00
		Light Bulb 60 Watt clear 220/235V		6,312,877.00	6,731,493.00
		Light Bulb 60 Watt frosted 220/235V		6,115,488.00	5,902,704.00
		Light Bulb 60 Watt red 220/235V		2,775,919.00	2,975,767.00
		Light Bulb 60 Watt yellow 220/235V		2,958,303.00	2,850,624.00
		Light Bulb 80 Watt clear 220/235V		7,095,740.00	7,589,438.00
		Light Bulb 80 Watt frosted 220/235V		5,120,425.00	4,935,200.00

Product Hierarchy for material MARA Level 01	**Product Hierarchy for material MARA Level 02**	**Product Hierarchy for material MARA Level 03**	**Product Hierarchy for material MARA Level 04**	**Sales Volume**	**Incoming Orders**
		Light Bulb 80 Watt red 220/235V		2,416,380.00	2,416,380.00
		Light Bulb 80 Watt yellow 220/235V		2,647,459.00	2,550,863.00
Machines	Pumps	Special pump	Pump GG IDESNORM 100-200	5,798,946.55	6,191,646.55
			Pump cast steel IDESNORM 150-200	6,721,540.00	6,962,580.00
			Pump cast steel IDESNORM 170-230	9,558,837.00	9,565,269.00
			Pump chrome-steel IDESNORM 150-200	475,602.00	495,702.00
			Pump standard IDESNORM 100-402	6,893,740.00	7,336,070.00
			pump CR IDESNORM 150-200 ATO	7,822,280.00	8,343,320.00
			pump sphere-cast IDESNORM 150-200	7,057,540.00	7,336,720.00

Product Hierarchy for material MARA Level 01	**Product Hierarchy for material MARA Level 02**	**Product Hierarchy for material MARA Level 03**	**Product Hierarchy for material MARA Level 04**	**Sales Volume**	**Incoming Orders**
Paints	Gloss paints	Opaque	Coating Matt Green RAL 6014/10 Liter	.00	125.03
Services	Maintenance	HiTech maintenance	PC Service (Configurable)	171,992.54	285,995.83
			PC Service Plus	119,570.00	199,320.00
Vehicles	Cars	Car (complete)	SAPSOTA FUN DRIVE 2000GT	155,640.00	155,640.00
	Motorcycles	Accessories	Motorcycle Helmet - Standard	3,651,535.18	3,800,706.24
		Components	Deluxe Gas Tank Striping Decals	658,897.52	703,373.20
			Deluxe Headlight	1,609,718.67	1,675,515.07
			Deluxe Taillight	546,588.47	579,120.40
		Motor-cycle (compl.)	CrossFun / 350 cm3	10,842,211.75	11,243,347.19
			IDES Glad Boy configurable	929,800.00	953,830.00
			SunFun / 1200 cm3	23,522,868.17	23,538,918.90

CHAPTER 41

Using the Adapter for SAP R/3

Topics:

The Adapter for SAP R/3 allows DataMigrator, WebFOCUS for SAP, and other applications to access SAP R/3 data sources. The adapter converts data or application requests into native SAP R/3 statements and returns optimized answers sets to the requesting program.

For sample requests and output, see *Producing SAP R/3 Requests* on page 41-47.

Preparing the SAP R/3 Environment

How to:

Log on to SAP

Create Development Class or Package

Create a Function Group

Deactivate the Unicode Checks

Activate a Function Group

Verify a Function Group

You will use SAP GUI to logon to SAP R/3 and prepare the environment. You will need an SAP user ID and password to logon to SAP R/3.

A Developer's Key should be available and preferably entered for the SAP user ID. The system prompts for the Developer's Key if it has not been entered previously.

The following SAP Authorizations are required at a minimum. This list may change depending on your SAP R/3 Release and/or site specific Authorization Profiles.

The adapter allows using two separate user IDs at the console, IBI_USER and USER. USER is used to submit the dynamically created ABAP/4 requests. All other tasks including creation of the Development Class and Function Group, installing the SAP Components, and creating the dynamic ABAP programs for the corresponding WebFOCUS requests are carried out using IBI_USER.

LIST OF AUTHORIZATION OBJECTS, THEIR FIELDS, AND THE REQUIRED VALUES FOR IBI_USER

S_RFC: Authorization check for RFC access

Field	**Value**
ACTVT (Activity)	16 (Execute)
RFC_NAME (Name of RFC to be protected)	AQRC, SDTX, SLST, SUTL, SYST, Z***
RFC_TYPE (Type of RFC object to be protected)	FUGR (function group)

S_TCODE: Authorization check for Transaction Start

Field	Value
TCD (Transaction code)	S001, SE09, SE11, SE11_OLD, SE13, SE37, SE38, SE80, SU53

S_ADMI_FCD: System Authorizations

Field	Value
S_ADMI_FCD (System administration functions)	MEMO

S_C_FUNCT: C Calls in ABAP Programs

Field	Value
ACTVT (Activity)	16 (Execute)
CFUNCNAME (Name of a CALLable C routine)	SYSTEM
PROGRAM (ABAP program name)	SAPLSTRF, SAPLSTRI

S_DATASET: Authorization for File Access

Field	Value
ACTVT (Activity)	33, 34, A6
FILENAME (Physical file name)	*
PROGRAM (ABAP program name)	*

S_TABU_DIS: Table Maintenance (via standard tools such as SM30)

Field	Value
ACTVT (Activity)	03
DICBERCLS (Authorization group)	SS, &NC&

S_DEVELOP: ABAP Workbench

Field	**Value**
ACTVT (Activity)	01, 02, 03, 07, 16, 40
DEVCLASS: Development class for transport system	*
OBJNAME: Object name	*
OBJTYPE: Object type	DEVC, FUGR, PROG, TABL, TABT
P_GROUP: Authorization group with ABAP programs	*

S_TRANSPORT: Transport Organizer

Field	**Value**
ACTVT (Activity)	01
TTYPE (Request type (Change and Transport System))	DTRA, TASK

LIST OF AUTHORIZATION OBJECTS THEIR FIELDS AND THE REQUIRED VALUES FOR USER

S_RFC: Authorization check for RFC Access

Field	**Value**
ACTVT (Activity)	16 (Execute)
RFC_NAME (Name of RFC to be protected)	SYST, Z***
RFC_TYPE (Type of RFC object to be protected)	FUGR (function group)

S_TCODE: Authorization check for Transaction Start

Field	**Value**
TCD (Transaction code)	S001, SE37, SE38, SU53

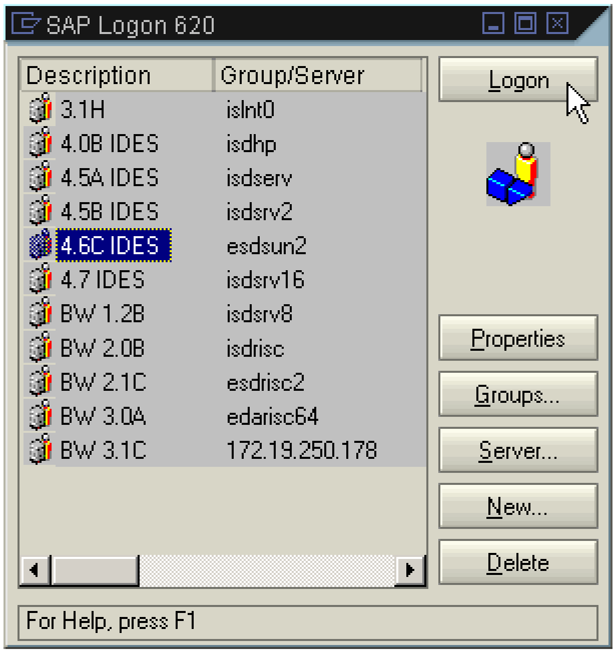
The Adapter for SAP R/3 generates ABAP/4 programs dynamically at run time. This is done using adapter components uploaded into the SAP R/3 system during the configuration of the adapter. Prior to uploading the components, the SAP R/3 system has to be prepared.

To prepare the SAP R/3 environment for the adapter components, perform the following steps:

1. Log on to SAP.
2. Create the Development Class or Package.
3. Create the Function Group.
4. Deactivate the Unicode checks - SAP R/3 4.7 overlay.
5. Activate the Function Group.
6. Verify the Function Group.

Procedure: How to Log on to SAP

1. Launch SAP logon. In a standard SAP GUI installation, the executable file resides in \Program Files\SAP\FrontEnd\SAPgui\saplogon.exe. The SAP Logon dialog box opens.

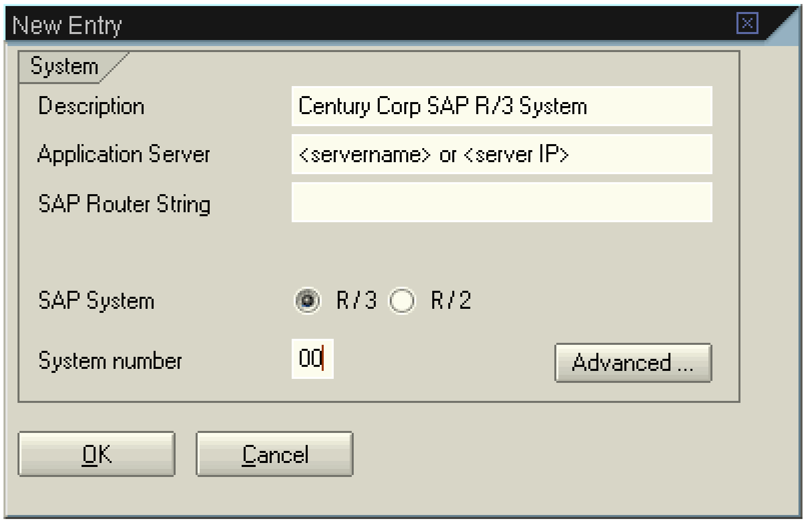


Description	Group/Server
3.1H	islnt0
4.0B IDES	isdhp
4.5A IDES	isdserv
4.5B IDES	isdsrv2
4.6C IDES	esdsun2
4.7 IDES	isdsrv16
BW 1.2B	isdsrv8
BW 2.0B	isdrisc
BW 2.1C	esdrisc2
BW 3.0A	edarisc64
BW 3.1C	172.19.250.178

For Help, press F1

If the list is empty, create a new entry.

a. Click *New*. The New Entry dialog box opens.



New Entry

System

Description: Century Corp SAP R/3 System

Application Server: <servername> or <server IP>

SAP Router String:

SAP System: [✓ R/3] R/2

System number: 00

b. Type the Description and Application Server, and select the System Number. In most cases, the SAP Router String is not needed, but check with your assigned SAP technical contact.

In the Application Server field, type either the SAP application server's network name or IP address.

Click *OK* to create a new entry on the Logon list.

2. Select the entry on the list and click *Logon*.

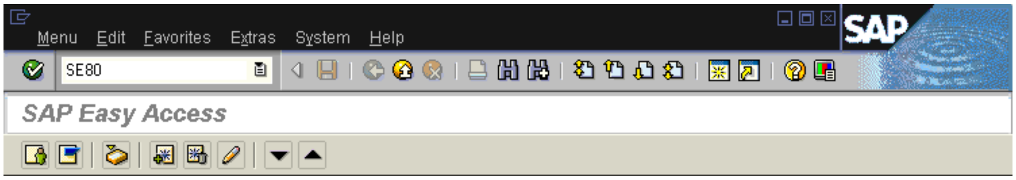
A logon page prompting for a user ID and password appears. Logon to the system. Note that if the user ID is being used for the first time, the system will ask for the password to be changed.

Procedure: How to Create Development Class or Package

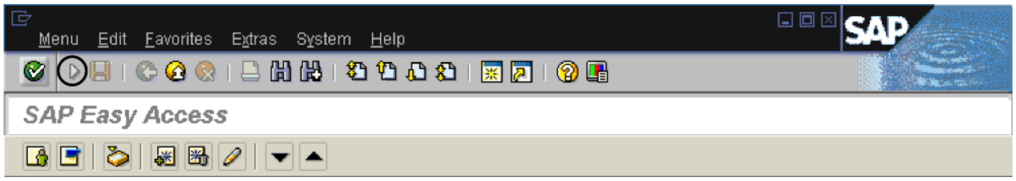
Note: SAP R/3 Release 4.7 introduced Package as an alternative to Development Class. If you are using Release 4.7, create a Package instead of a Development Class.

1. Execute transaction SE80 to start the Object Navigator.

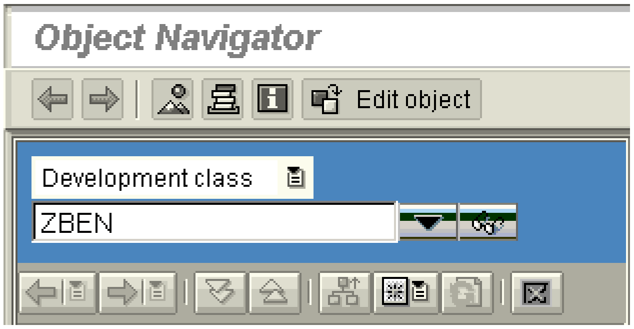
   Tip: The easiest way to execute a transaction is to type the transaction code on the command line box shown below.



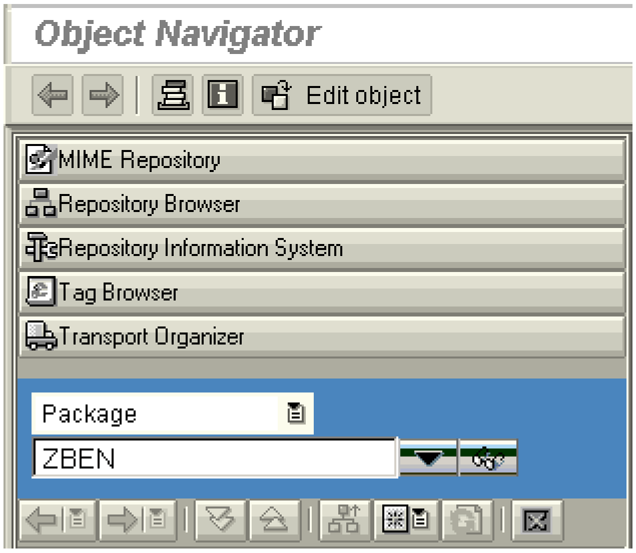
   If the command line is hidden when you logon to the system, click the arrow highlighted below.



2. For releases prior to 4.7, select *Development Class* from the pull-down menu.



For Release 4.7, select *Package* from the pull-down menu.



3. Type the Development Class or Package name for the installation of the Adapter for SAP R/3 (for example, ZBEN).

   Note: The Development Class or Package name can be a maximum of 14 characters and must follow SAP naming conventions for customer objects.

4. Click *Display* (The button with the picture of glasses).

   The system responds: Development Class or Package ZBEN does not exist.

5. Click *Yes* to create the object.
6. Type a short description for the Development Class or Package, for example, *IBI-Created*.
7. Accept all displayed defaults and click *Create*.

   The system generates a unique Change and Transport System (CTS) request #.

8. Save this CTS request # for deployment of the adapter to other SAP R/3 systems.
9. Click *Continue*.

Procedure: How to Create a Function Group

1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select *Function Group*.
3. Type the same Development Class or Package name for the installation of the Adapter for SAP R/3 that you specified in *How to Create Development Class or Package* on page 41-7 (for example, ZBEN).
4. Click *Display*.

   The system responds: Function Group ZBEN does not exist.
5. Click *Yes* to create the object.
6. Type a short description for the iWay Function Group, for example, *Information Builders Release 5.30 Function Group*.
7. Accept all displayed defaults and click *Save* (unless otherwise directed by SAP Basis Administrator).
8. Under Create Object Directory Entry, type the same characters assigned to the Development Class, and click *Save*.
9. Under Prompt for transportable change request, type the CTS request # assigned during the creation of the Development Class or Package.
10. Click *Continue*.

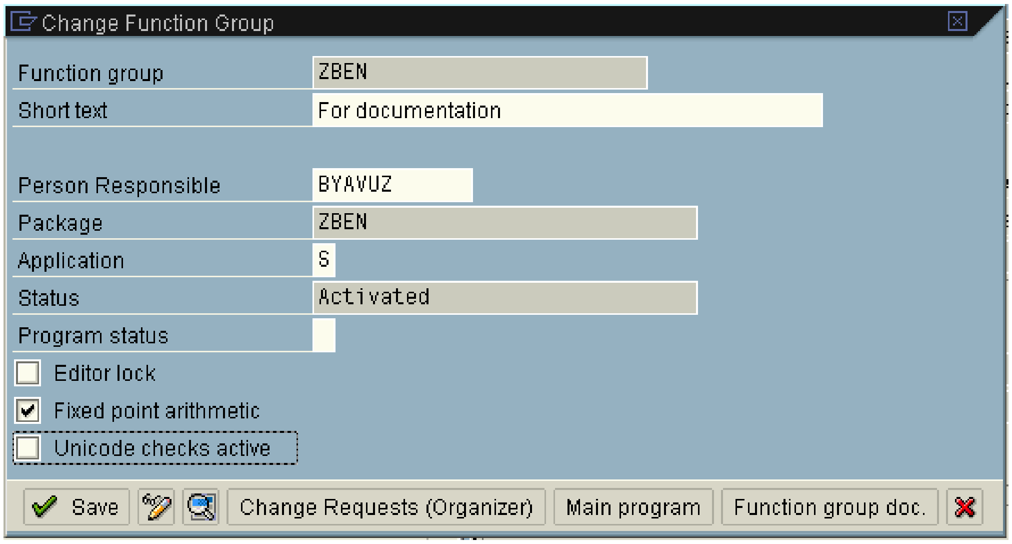
Procedure: How to Deactivate the Unicode Checks

This procedure applies only to SAP R/3 Release 4.7.

1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select *Function Group*.
3. For Function Group value, type the name of the Function Group that you created (for example, ZBEN).
4. Click *Display*.

   The object window shows an open folder with the name of the function group and a closed folder named Includes.

5. Right-click the function group name, for example, ZBEN, and select *Change* from the pop-up menu.



Function group	ZBEN
Short text	For documentation
Person Responsible	BYAVUZ
Package	ZBEN
Application	S
Status	Activated
Program status	

- [ ] Editor lock
- [x] Fixed point arithmetic
- [ ] Unicode checks active

6. Deselect the check box for Unicode checks active, and click *Save*.

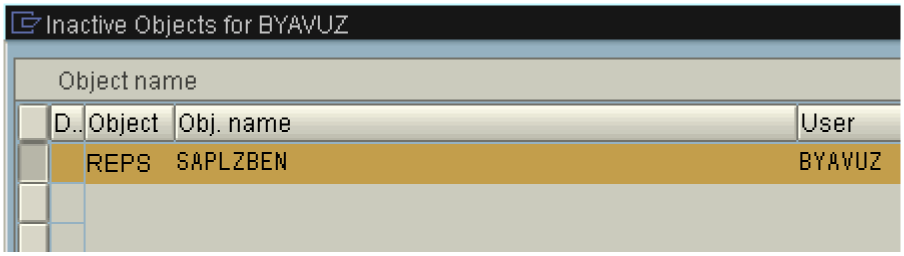
Procedure: How to Activate a Function Group

1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select *Function Group*.
3. For Function Group value, type the name of the function group you created (for example, ZBEN).
4. Click *Display*.

   The object window shows an open folder with the name of the function group and a closed folder named Includes.
5. Right-click the function group name (in this example, ZBEN) and select *Activate* from the pop-up menu.

   A new window opens which lists all inactive objects. Note that the object name now begins with SAPL.

   Make sure the object for the function group. In the example, ZBEN is selected.
6. Click *Continue*.


Inactive Objects for BYAVUZ

Object name

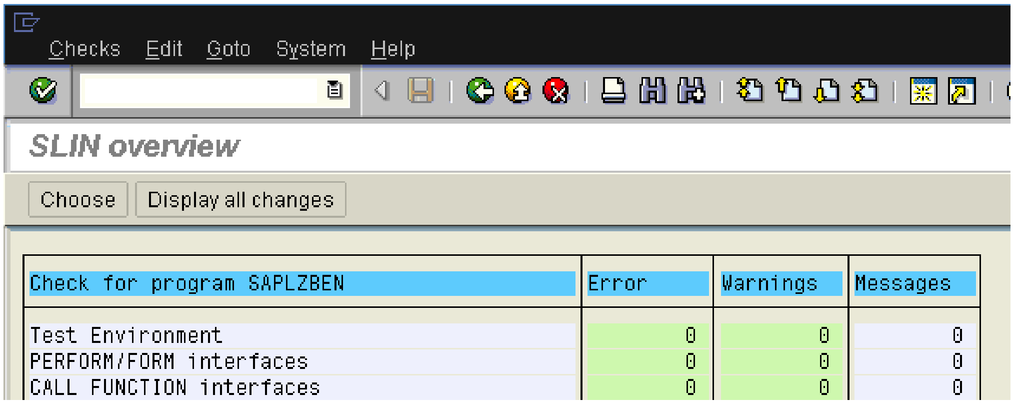
D..	Object	Obj. name	User
	REPS	SAPLZBEN	BYAVUZ

Procedure: How to Verify a Function Group

1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select *Function Group*.
3. For Function Group value, type the name of the function group created (for example, ZBEN).
4. Click *Display*.
5. Right-click the ZBEN folder and select *Check - Extended Check*.

6. Select *Perform Check*.

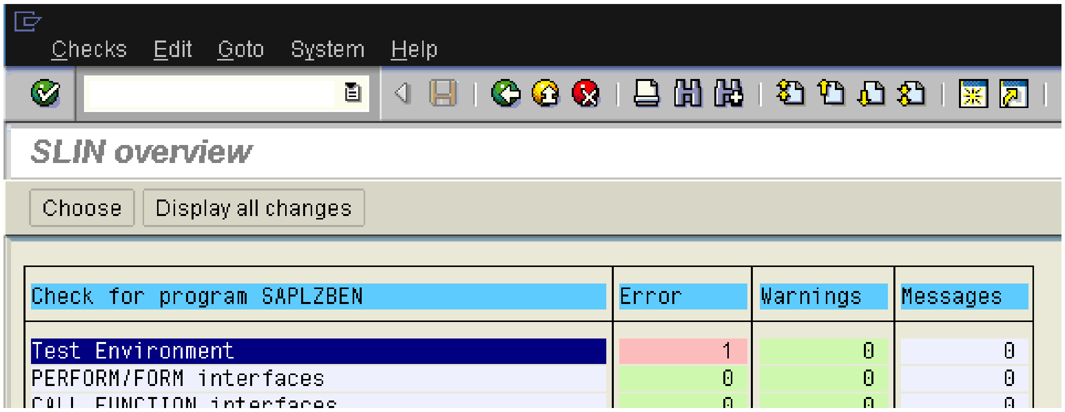
   You should get a report with 0 errors, 0 warnings, and 0 messages.



Check for program SAPLZBEN	Error	Warnings	Messages
Test Environment	0	0	0
PERFORM/FORM interfaces	0	0	0
CALL FUNCTION interfaces	0	0	0

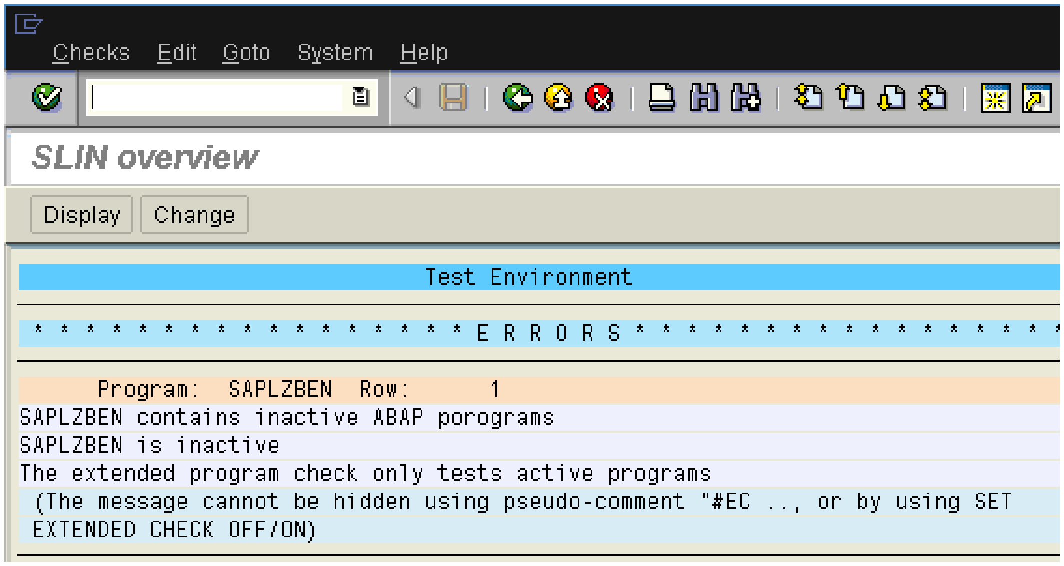
Note: If you encounter errors or warnings during this step, resolve them at the SAP R/3 layer before continuing with the installation of the Adapter for SAP R/3.

In the following image, the function group is not intentionally activated. The extended check returns an error.



Check for program SAPLZBEN	Error	Warnings	Messages
Test Environment	1	0	0
PERFORM/FORM interfaces	0	0	0
CALL FUNCTION interfaces	0	0	0

Double-click the error line for detailed information about the error.



```
Test Environment
* * * * * * * * * * * * * * * * * E R R O R S * * * * * * * * * * * * * * * * *
    Program:  SAPLZBEN  Row:      1
SAPLZBEN contains inactive ABAP porograms
SAPLZBEN is inactive
The extended program check only tests active programs
 (The message cannot be hidden using pseudo-comment "#EC .., or by using SET
 EXTENDED CHECK OFF/ON)
```

The SAP environment is now prepared for the next step of function module upload from the Web Console.

Accessing Multiple SAP R/3 Systems

The adapter can operate across multiple R/3 systems. In R/3, data from multiple companies can share the same metadata, and is identified by a client value. Normally, a given system has multiple clients, and you may access one or more of them depending on your authorization. For each system, the adapter requires one R/3 logon consisting of a client, a user, and a password. This logon must be:

- RFC-enabled.
- Authorized for all clients that you may access.

You may need additional authorizations depending on the SAP data you wish to access. Consult your SAP administrator for details.

Configuring the Adapter for SAP R/3

Configuring the Adapter for SAP R/3 consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Configure the Adapter on a MVS or OS/390 and z/OS Environment

Install SAP Components

Verify Installation of SAP Components

In order to connect to SAP R/3, the adapter requires connection and authentication information. You supply this information in the Web Console configuration page.

Procedure: How to Declare Connection Attributes From the Web Console

The configuration page prompts you for the following connection attributes. The settings are stored in etc/sapserv.cfg. The information is stored by blocks. A block is identified by the system ID (usually the three character name of the SAP instance).

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *ERP* group folder, then expand the adapter folder and click a connection. The Add SAP R/3 to Configuration window opens.
3. Provide valid values for input fields.

User Authentication Parameters

The major work of the adapter is to translate user requests into code that can be understood by SAP. For this purpose, an SAP user ID, with a given set of privileges, is required. In the following table, this user ID is referred to as IBI_USER. After the request has been generated and is ready for execution, it can be executed either under the IBI_USER ID, or under a less privileged user (referred to as USER); results are then based on the user's credentials.

Parameter	**Description**
System	SAP system ID.
IBI_CLIENT	SAP Client for the IBI logon, maximum 3 characters.
IBI_USER	SAP user ID for the IBI logon.

Parameter	Description
IBI_PASSWORD	SAP password for the IBI logon, maximum 8 characters.
CLIENT	SAP Client for the user logon, maximum 3 characters.
Security	There are two methods by which a user can be authenticated when connecting to an SAP R/3 instance: **Explicit.** The user IDs and password are specified for each connection and passed to SAP R/3 for authentication and request execution. **Password Passthru.** The user ID and password received from the client application are passed to SAP R/3 for authentication and to execute the user's request.
USER	SAP user ID for the user logon.
PASSWORD	SAP password for the user logon, maximum 8 characters.

Function Group Parameters

Parameter	Description
FPOOL	Function group where the Adapter for SAP R/3 static function modules can be cataloged. This is the function group created in *Preparing the SAP R/3 Environment* on page 41-2.
DATACLASS	SAP data class for user defined SAP tables generated by the Adapter for SAP R/3 (default appl0). Valid classes are appl0, appl1, appl2, and usr1. The maximum is five characters.
TMPDIR	Temporary directory to be used for SAP extracts (default /tmp/), maximum 64 characters. These extracts are flat files that can only be written to a read/write directory, and are readable for the end user.
RANGEBEG	Beginning range for the function group.
RANGEEND	Ending range for the function group.

Connection Parameters

Parameter	Description
APPGRP	Name of the application group. An application group defines a list of application servers, on which an RFC application can be running. R/3 transaction SMLG can be used to view or modify application groups.

Parameter	**Description**
MSGHOST	Name of the SAP message server.
HOST	Host name of the SAP application server.
GWHOST	Host name of the machine where the SAP gateway process is running. In the case where there is only one SAP application server, gwhost and host is the same.
SYSNR	SAP system number. Obtain this value from the SAP Administrator.
LANGUAGE	Language installed on SAP to be used with the Adapter for SAP R/3. Typically, this is the default SAP language.

Parameters for Mixed Character Code Set

The following parameters apply only when the server platform and the SAP instance platform do not use the same character code set (ASCII or EBCDIC). See *Supporting Mixed Code Page Environments* on page 41-41 for detailed information.

Parameter	**Description**
eda2sap	Server to SAP host conversion table, as generated in *Preparing the SAP R/3 Environment* on page 41-2. This is the conversion file where the server code page is the source and the SAP code page is the target. In most cases, this parameter is blank.
sap2eda	SAP host to server conversion table file name, as generated in *Preparing the SAP R/3 Environment* on page 41-2. This is the conversion file in which the SAP code page is the source and the server code page is the target. In most cases, this parameter is blank.

Note: If the SAP R/3 system is running on an MVS/USS environment, see *How to Configure the Adapter on a MVS or OS/390 and z/OS Environment* on page 41-17.

Once all necessary parameter values are entered, click *Configure* to save the entry in the edasprof.prf file.

After configuring the adapter, it is necessary to install the static function modules in the SAP data dictionary to complete the configuration.

Procedure: How to Configure the Adapter on a MVS or OS/390 and z/OS Environment

To configure the adapter if running on an MVS or OS/390 and z/OS environment, the SAP Code Page 0126 must be installed on the R/3 server. This code page will be provided with one of the upcoming SAP Basis Support Packages, possibly with 45. Follow the instructions from SAP to install the Code Page.

1. Create two conversion tables by following the instructions in *Supporting Mixed Code Page Environments* on page 41-41, and transfer the tables to the $EDACONF/etc directory.

   For example, if the iWay server environment uses the code page 1100, then two conversion tables, 11000126.CDP and 01261100.CDP, should be created and transferred to the $EDACONF/etc directory.

   Important: Do not export the variables SAP_CODEPAGE and PATH_TO_CODEPAGE since these variables are not used by the adapter.

2. Assign the correct code page files to the parameters of eda2sap and sap2eda by using the Web Console or manually editing. Make sure to type CDP in capital letters. For example:

   ```
   eda2sap=01261100.CDP
   ```

   ```
   sap2eda=11000126.CDP
   ```

Procedure: How to Install SAP Components

To install SAP components:

1. Launch the Web Console.
2. In the navigation pane, click *Adapters*.
3. In the left pane, expand the Configured folder, then the SAP R/3 folder and click the system created in *Declare Connection Attributes From the Web Console* on page 41-14.

   A pop-up menu opens.
4. Select *Properties*.

   The Change SAP System Connect Parameters pane opens in the right pane.
5. If you are re-installing due to an error in a previous installation, scroll to the bottom of the left pane, which shows the Install SAP Components button.
6. Select the *Overwrite existing Function Group* check box only if an error occurred in a previous installation.
7. Click *Install SAP Components*.

   A message appears confirming this request.

8. Click *OK* to continue the installation.

   A new browser window opens with a message "Processing the request".

   This process may take a few minutes. Do not close the browser window. The message Done appears in the browser window along with any errors.

The following is a sample message which you may receive if you try to reinstall the SAP components without checking the Overwrite Existing Function Group:

```
(FOC44427) YOU ARE ABOUT TO OVERWRITE THE FOLLOWING FUNCTION MODULES
(FOC44428)   ZY52_DYNAMIC_RUN
(FOC44428)   ZY52_GET_APPLICATION_TREE
(FOC44428)   ZY52_REPORT_ABORT_BATCH
(FOC44428)   ZY52_REPORT_CREATE
(FOC44428)   ZY52_REPORT_DELETE
(FOC44428)   ZY52_REPORT_GET_BATCH_DATA
(FOC44428)   ZY52_REPORT_GET_BATCH_STATUS
(FOC44428)   ZY52_REPORT_RUN
(FOC44428)   ZY52_REPORT_RUN_IN_BATCH
(FOC44428)   ZY52_TEST_REVERSE
(FOC44429) RERUN THE COMMAND WITH THE OVERWRITE OPTION
```

Procedure: How to Verify Installation of SAP Components

To confirm that all necessary function modules are successfully installed:

1. Run transaction SE80.
2. Select *Function Group* from the first drop-down list.
3. Type the name of the Function Group you created in *Declare Connection Attributes From the Web Console* on page 41-14.
4. Click *Display* next to the text box.

   Folders named Function Modules and Includes should be under the Function Group name.

5. Expand the Function Modules and Includes folders to verify that the following appears:

```
Z***_DYNAMIC_RUN
Z***_GET_APPLICATION_TREE
Z***_REPORT_ABORT_BATCH
Z***_REPORT_CREATE
Z***_REPORT_DELETE
Z***_REPORT_GET_BATCH_DATA
Z***_REPORT_GET_BATCH_STATUS
Z***_REPORT_RUN
Z***_REPORT_RUN_IN_BATCH
Z***_TEST_REVERSE
```

   Note: Z*** represents the name of the Function Group. If the Development Class and Function Group name is ZIBI, then the Function Module name begins with ZIBI.

Post-Configuration Tasks in an SAP R/3 Environment

In this section:

Customizing the Transport/Promotion of the ZXXXREPTS Temporary Object

Example:

Transporting a Request

Reference:

Transport Data File Names

Change Request Information File

SAP R/3 Reserved Names

During the installation of the Adapter for SAP R/3, the following objects are added to the R/3 repository:

- **A client-independent table object**, Z*XXX*REPTS in Dev Class $TMP. The table object is used internally by the adapter at run time to store non-persistent information. It includes the following attributes:

  Transparent Table NAME: Z*XXX*REPTS

  Status: Active

  Attributes Development class: $TMP

  The table contains 2 columns: SEQNUM (NUM4) and REPID (CHAR08)

- **A set of function modules** in the corresponding development class (Z*XXX*).

A certified SAP R/3 system administrator must manually customize the transport/promotion of the Z*XXX*REPTS temporary object in SAP system landscape. Note that *XXX* represents the 3 characters that are set by a certified SAP administrator to uniquely identify the client-independent table object and function modules. For details, see *Customizing the Transport/Promotion of the ZXXXREPTS Temporary Object* on page 41-20.

Customizing the Transport/Promotion of the ZXXXREPTS Temporary Object

SAP provides a transport control program (tp) for controlling release upgrades and transports among SAP systems. The transport control program:

- Keeps track of transports.
- Exports and imports objects in the correct order.
- Ensures that imports into a target system are performed in the same order as the exports from the source system(s). (Processing of imports out of order can result in severe inconsistencies in the target system, which are difficult to diagnose.)
- Enables you to perform exports and imports separately.

During an *export*, the objects to be transported are extracted from the database of the source system and stored in files of the operating system.

During the *import*, the objects are added to the database of the target system (according to the transport function recorded in the task).

Tip: For detailed technical background, navigate to *http://help.sap.com-> (version 4.6C)* and the menu path: *SAP Library->Basis Components->Change and Transport System (BC-CTS)-> Transport Tools (BC-CTS-TLS)>Transport Control Program t*p. From here:

- Users should proceed to the section entitled *Preparing Operating System Users.*
- System administrators should proceed to the section entitled *Preparing the SAP System*.

Reference: Transport Data File Names

All data files are located in the transport directory data. The name of a data file consists of the name of the change request, and a code letter that distinguishes between data files generated by R3trans and those generated by application programs for application-specific development environment objects (ADOs:

- R <6 digits>.<source system> R3trans
- D <6 digits>.<source system> application programs

Reference: Change Request Information File

The change request information file contains information about a change request, including the transport type and the class of the objects to be transported. It also contains information about the steps required for the change request, exit codes, and the time of execution.:

This information file is located in the cofiles directory. The name is derived from the name of the change request:

K <6 digits>.<source system>

Reference: SAP R/3 Reserved Names

The iWay Adapter for SAP R/3 is compatible with the SAP Basis 4.6C version. It reserves the following names:

- Development class Z*XXX*
- Table Z*XXX*REPTS
- Function Group Z*XXX* and function module names with Z*XXX**

where:

XXX

Represents the 3 characters that are set by a certified SAP administrator to uniquely identify the client-independent table object and function modules. Note that a conflict with an existing unique name may cause the transport to fail.

Example: Transporting a Request

This illustration uses the sample request K900022 on P46. You may use it as a template for transporting a request.

Complete the following tasks in the order appropriate for each SAP system.

- Release the request from the sending system.
- Goto */usr/sap/transport/cofiles* and copy *K900022.P46* to the target system cofile directory.
- Goto */usr/sap/transport/data* and copy *R900022.P46* to the target system data directory.
- To check the transport system status of your SAP system, enter the command

  ```
  tp checkimpdp <sapsid>
  ```

  where:

  sapsid

  Is the system id. For example, P46, which is defined during the SAP system installation process.

- Check the status of the tp import buffer with the command:

  ```
  tp showbuffer <sapsid>
  ```

  Tip: Several tp commands are used in this procedure. For detailed information about tp procedures and commands, refer to the SAP tp documentation at *http://help.sap.com*.

- Clear the buffer using the command:

  ```
  tp cleanbuffer <sapsid>
  ```

- Check the status to make sure all items have been cleared using the command:

  ```
  tp showbuffer <sapsid>
  ```

- Add the transport using the command:

  ```
  tp add tobuffer P46K900022 <sapsid>
  ```

- Check that the transport has been added to the buffer using the command:

  ```
  tp showbuffer <sapsid>
  ```

- Import the transport using the command:

  ```
  tp import P46K900022 <sapsid> u1
  ```

  Note that you may need to add other options for override. For details, refer to the tp options documentation at *http://help.sap.com*.

- Verify that the *ZXXX* development class has been imported and activated, as indicated on your system screens.

Managing SAP R/3 Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server accesses, you create a synonym that describes the structure of the data source and maps the server data types to the SAP R/3 data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each SAP data object that is accessible from the server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File, which represent the server's metadata.

Reference: Types of SAP Synonyms

The Adapter for SAP R/3 enables you to create:

- **SAP Synonyms**

  To create synonyms for SAP tables, Logical Data Base (LDBs) or Business Application Programming Interface (BAPIs), the adapter requires that base metadata be available. You must generate this base metadata before you begin the synonym creation. This task is carried out using the same window for creating the synonyms. The task is submitted as a deferred request and it takes about 15-20 minutes. It is suggested that you should not do any other activity using the Web Console while the base is being generated. If an underlying data object changes, you must regenerate base metadata for subsequent synonym creation.

- **FM Synonyms**

  To create synonyms for SAP function modules, the only thing needed is to know the exact name of the function module. Note that not all function modules are suitable for reporting.

- **IDOC Synonyms**

  To create synonyms for Intermediate Document Interface (IDOCs), the adapter requires that an IDOCs list be available. You must generate this list before you begin the synonym creation for the first time. If an underlying IDOC changes, due to SAP R/3 requirements, regenerate your IDoc synonyms after assessing the impact on any active DataMigrator flows. Keep in mind that IDOC synonyms are used for DataMigrator purposes and not for queries.

- **Query Synonyms**

  An SAP query is a predefined report designed for users with little or no knowledge of the SAP programming language ABAP. From InfoSets (data sources), which are organized in user groups, an SAP query creates a list of information not already contained in the SAP system. Being a report, it may have variants. A variant is a saved set of selection criteria for a query or the report generated by the query. If you specify a variant when starting a query, the system uses the values in the variant for the query selection criteria.

  To create a synonym for an SAP query, base metadata must be created. Note that the SAP query base metadata must be refreshed to see newly created queries.

Procedure: How to Create an SAP Synonym Using the Web Console

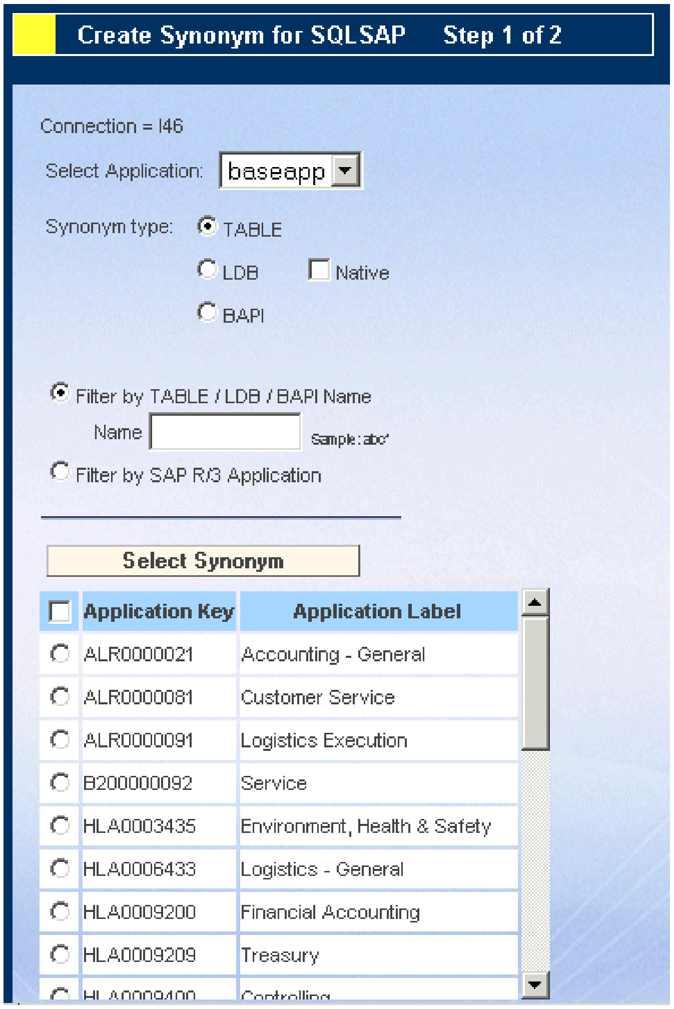
1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for the adapter.

4. From the resulting menu, select *Create SAP Synonym*. The Create Synonym pane opens.



Connection = I46

Select Application: baseapp

Synonym type: [✓ TABLE] LDB Native BAPI

[✓ Filter by TABLE / LDB / BAPI Name]

Name Sample:abc*

Filter by SAP R/3 Application

	Application Key	Application Label
○	ALR0000021	Accounting - General
○	ALR0000081	Customer Service
○	ALR0000091	Logistics Execution
○	B200000092	Service
○	HLA0003435	Environment, Health & Safety
○	HLA0006433	Logistics - General
○	HLA0009200	Financial Accounting
○	HLA0009209	Treasury
○	HLA0009400	Controlling

5. Select a synonym type, either TABLE, LDB, or BAPI.
6. To filter by name, select the *Filter by TABLE/LDB/BAPI Name* radio button and in the Name field, type a string, using the wildcard character (*) as needed.

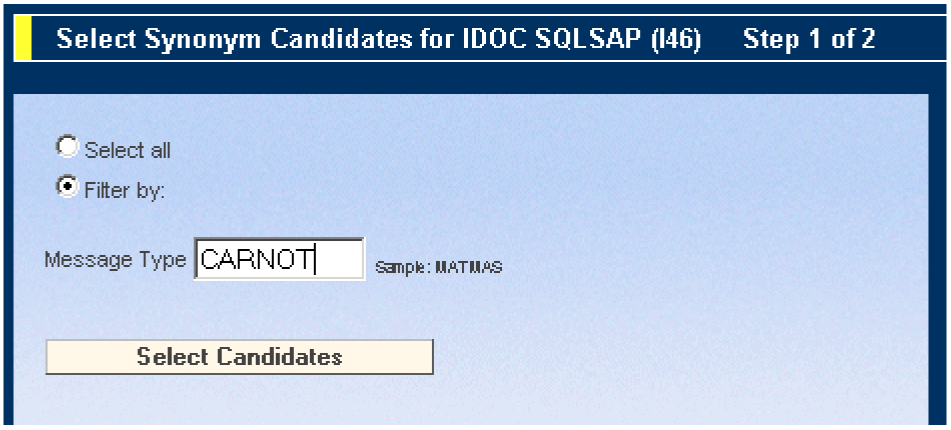
   or

   To filter by application, select the *Filter by SAP R/3 Application* radio button, and select an application from the list.
7. Select a directory from the Select Application drop-down list. baseapp is the default value.
8. Click *Select Synonym*. Synonyms are created and added under the specified application directory.

Procedure: How to Create an IDOC Synonym Using the Web Console

1. Start the Web Console. In the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter. For information on how to configure an adapter, see *Configuring the Adapter for SAP R/3* on page 41-14.
2. Expand the Add folder and the adapter folder, and click a connection.
3. From the resulting menu, select *Create IDOC Synonym*. The IDOC Metadata Management page opens.


Select Synonym Candidates for IDOC SQLSAP (I46) Step 1 of 2

Select all [✓ Filter by:]

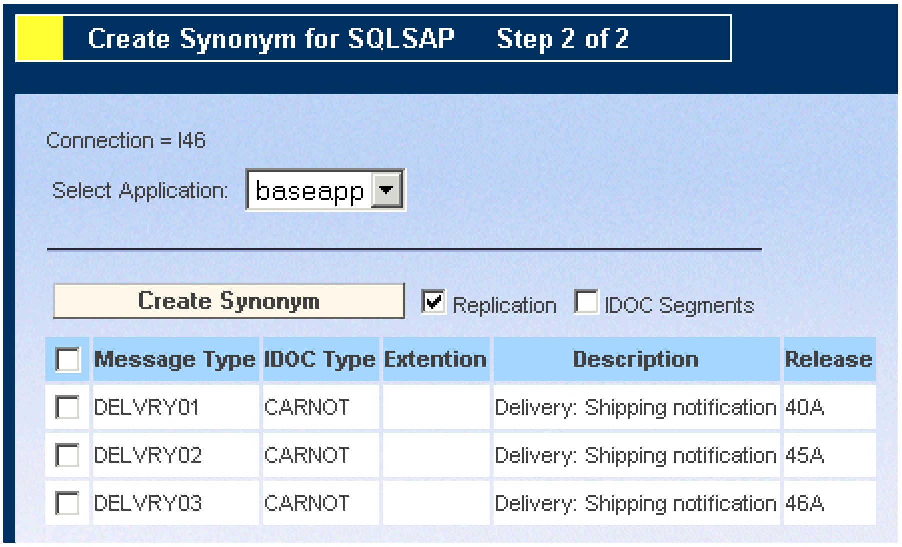
Message Type: CARNOT Sample: MATMAS

4. If necessary, create an SAP IDOCs list. For more information, see *Types of SAP Synonyms* on page 41-24.

5. To list IDOCs for a specific message type, type a message type in the Message Type field (for example, CARNOT).

   To list all IDOCs, leave the Message Type field blank.

6. Click *Select Candidates*. The IDOCs that meet your criteria are listed.



Connection = I46

Select Application: baseapp

[✓] Replication [ ] IDOC Segments

☐	Message Type	IDOC Type	Extention	Description	Release
☐	DELVRY01	CARNOT		Delivery: Shipping notification	40A
☐	DELVRY02	CARNOT		Delivery: Shipping notification	45A
☐	DELVRY03	CARNOT		Delivery: Shipping notification	46A

Note: If you receive the message "No IDOCs Found," it is likely that the IDOC list was not generated. See Step 4.

7. Select a directory from the Select Application drop-down list. baseapp is the default value.

8. Ensure that Replication is checked.

9. Select the IDOCs for which you want to create synonyms, and click *Create Synonym*. Synonyms are created and added under the specified application directory.

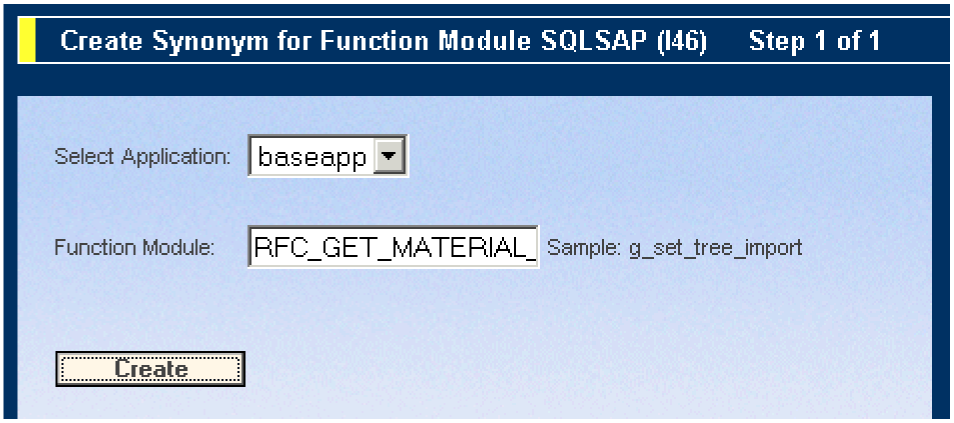
Procedure: How to Create an FM Synonym Using the Web Console

1. Start the Web Console. In the navigation pane, click *Metadata*. The Managing Metadata pane opens.

   To create a synonym, you must have configured the adapter. For information on how to configure an adapter, see *Configuring the Adapter for SAP R/3* on page 41-14.

2. Expand the Add folder and then the adapter folder, and click a connection.

3. From the resulting menu, select *Create FM Synonym*. The Function Module pane opens.



Create Synonym for Function Module SQLSAP (I46) Step 1 of 1

Select Application: baseapp

Function Module: RFC_GET_MATERIAL_ Sample: g_set_tree_import

4. Select a directory from the Select Application drop-down list. baseapp is the default value.

5. In the Function Module name field, type a module name and click *Create*.

The synonym is created and added under the specified application directory.

Procedure: How to Create a SAP Query Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter. For information on how to configure an adapter, see *Configuring the Adapter for SAP R/3* on page 41-14.

2. Expand the Add folder, expand the adapter folder, and then click a connection.

3. From the resulting menu, select *Create QUERY Synonym*. The Query Metadata Management pane opens.

4. The SAP Queries can be created under two different query areas in SAP R/3: Standard Area (client-specific) and Global Area (cross-client). By default, queries created under the Standard Area are selected. To select queries created under the Global Area, click the *Cross Client Queries* check box.

5. If necessary, select *Create SAP QUERYs List*. For more information, see *Types of SAP Synonyms* on page 41-24.

6. To list QUERYs for a specific Group, select *Filter by* and type a Group Name. You can also type the Query Name to filter the result set.

   To list all QUERYs, choose *Select all QUERYs*.

7. Click *Select QUERY Synonyms*.

   The SAP QUERYs that meet your criteria appear.

   Note: If you receive the message *No QUERYs Found*, it is likely that the QUERY list was not generated. See Step 4.

8. Select a directory from the Select Application drop-down list. baseapp is the default value.

9. Select the QUERYs for which to create synonyms and click *Create QUERY Synonym*.

   Synonyms are created and added under the specified application directory.

Note: To be able to run WebFOCUS requests against queries that were originally created under the Global Area (cross-client), add the following selection criteria to the request:

```
WHERE WORKSPACE EQ 'X';
```

Reference: Creating an SAP Query Synonym

When creating an SAP query synonym, all variants (if any) will be found at CREATE SYNONYM time. A two segment Master File is created. The top segment holds the VARIANT field for all possible variants. The second segment is the actual report fields. A variant must be specified for running the report.

For example, with the following synonym:

```
$     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$            SNAPpack generated on Thu Feb 20 11:44:07 2003
$            SAP instance: I46   SAP Release: 46C
$            MFD generated for QUERY: BASEAPP/ALM01_02
$     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

FILE=BASEAPP/ALM01_02  ,SUFFIX=SQLSAP,
  REMARKS='UG: 02 Query: ALM01',$
SEGNAME=VARIANT,SEGTYPE=S0,$
  FIELD=VARIANT, ALIAS=VARIANT,ACTUAL=A14,USAGE=A14,ACCEPT=
     'CAUS' OR
     'MY VARIANT' OR
     '             ',$
SEGNAME=REPORT,PARENT=VARIANT,SEGTYPE=S0,$
  FIELD=SKA1_SAKNR, ALIAS=SKA1_SAKNR, USAGE=A010, ACTUAL=A010,
  TITLE='G/L acct', DESC='G/L account number',$
  FIELD=SKA1_GVTYP, ALIAS=SKA1_GVTYP, USAGE=A002, ACTUAL=A002,
  TITLE='AT', DESC='P&L statement account type',$
  FIELD=SKC1C_BUKRS, ALIAS=SKC1C_BUKRS, USAGE=A004, ACTUAL=A004,
  TITLE='CoCd', DESC='Company Code',$
  FIELD=SKC1C_GJAHR, ALIAS=SKC1C_GJAHR, USAGE=A004, ACTUAL=A004,
  TITLE='Year', DESC='Fiscal year',$
  FIELD=SKC1C_GSBER, ALIAS=SKC1C_GSBER, USAGE=A004, ACTUAL=A004,
  TITLE='BA', DESC='Business Area',$
  FIELD=SKC1C_HWAER, ALIAS=SKC1C_HWAER, USAGE=A005, ACTUAL=A005,
  TITLE='Crcy', DESC='Currency Key',$
  FIELD=SKC1C_UM01O, ALIAS=SKC1C_UM01O, USAGE=P16.2, ACTUAL=A16,
  TITLE='SKC1C-UM01', DESC='Monthly balance',$
  FIELD=SKC1C_HWAER1, ALIAS=SKC1C_HWAER1, USAGE=A005, ACTUAL=A005,
  TITLE='Crcy', DESC='Currency Key',$
```

A valid request using variant 'CAUS' is:

```
TABLE FILE ALM01_02
PRINT
  SKA1_SAKNR
  SKA1_GVTYP
  SKC1C_BUKRS
  SKC1C_GJAHR
  SKC1C_GSBER
  SKC1C_HWAER
  SKC1C_UM01O
  SKC1C_HWAER1
IF VARIANT EQ 'CAUS'
IF READLIMIT EQ 50
END
```

Reference: Managing Synonyms

In the navigation pane, click the name of the synonym to access the following options:

Sample Data	Retrieves up to 20 rows from the associated data source. **Note:** This option is not available for IDOC synonyms.
Properties	Displays a graphic representation of the synonym and enables you to edit its metadata.
Edit Master File	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the Properties option, rather than manually editing the Access File.
Edit Access File	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the Properties option, rather than manually editing the Access File.
Refresh	Regenerates an SAP or FM synonym. Use this option if structural changes were made to the data source.
Archive	Regenerates an IDOC synonym. Use this option if structural changes were made to the source IDOC.
Drop	Deletes the synonym.
Copy to	Copies the synonym to another application directory. Click the target directory from the resulting list.
Move to	Moves the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR source DBMS SQLSAP AT system
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

source

Possible values are TABLE, LDB, BAPI, FM, IDOC, QUERY.

SQLSAP

Indicates the Adapter for SAP R/3.

system

Is a system name previously defined using the SET CONNECTION_ATTRIBUTES command. When the synonym is created, this system name becomes the value for the CONNECTION attribute in the Access File.

END

Indicates the end of the command, and must be on a separate line in the procedure.

Example: CREATE SYNONYM

```
CREATE SYNONYM baseapp/DD02T FOR DD02T TABLE DBMS SQLSAP AT I46
END
```

Generated Master File DD02T.mas

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       SNAPpack generated on Tue Nov 26 17:42:06 2002
$       SAP instance: I46   SAP Release: 46C
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
FILE=DD02T, SUFFIX=SQLSAP
  REMARKS='R/3 DD: SAP table texts',$

SEGNAME=DD02T,SEGTYPE=S0,$
  FIELD=DD02T_TABNAME                            ,TABNAME ,
  A30         ,A30         ,MISSING=OFF,
  TITLE='Table',   DESC='Table name',$
  FIELD=DD02T_DDLANGUAGE                         ,DDLANGUAGE ,
  A1          ,A1          ,MISSING=OFF,
  TITLE='Lang.',   DESC='Language key',$
  FIELD=DD02T_AS4LOCAL                           ,AS4LOCAL ,
  A1          ,A1          ,MISSING=OFF,
  TITLE='Activation',   DESC='Activation status of a Repository object',$
  FIELD=DD02T_AS4VERS                            ,AS4VERS ,
  A4          ,A4          ,MISSING=OFF,
  TITLE='Version of',   DESC='Version of the entry (not used)',$
  FIELD=DD02T_DDTEXT                             ,DDTEXT ,
  A60         ,A60         ,MISSING=OFF,
  TITLE='Short text',   DESC='Short text describing R/3 Repository
objects',$
```

Generated Access File DD02T.acx

```
SYSTEM=I46,$
SEGNAME=DD02T     ,
TABLENAME=DD02T, TABLETYPE=TRANSP,
MANDT=NO,
KEYS=4
,$
```

Limitations for Logical Databases

Synonym for SAP Logical Databases (LDB) can be generated in two different formats: SQL mode (set of joined SQL tables) or Native Mode (use of the underlying SAP Logical Database Program).

- **SQL Mode.** There are no limitations on accessing an LDB in SQL mode, other than limitations documented elsewhere in the server documentation concerning the number of Joins or number of fields in a synonym.

  From an end-user standpoint, SQL Mode usually runs faster, and is, therefore, the preferred method. Native Mode should be used only when there is a requirement to access so-called *structures*—for example, the segments SKC1A, SKC1C in SDF.

- **Native Mode.** Most LDBs in Native mode open a selection screen at run-time, on which some of the parameters are required. Since the Adapter for SAP/R3 works through RFC, it has no knowledge of this screen(s). Only the following LDBs are currently certified: VAV, SDF, KDF, CEK, CFK, CIK, CPK, CRK. Others need field certification.

Limitations for Function Modules and BAPIs

- A function module must have at least one Import or Export parameter.
- When filtering with an internal table of type Select Option, any combination of conditions can be applied.
- Filtering with an internal table that is not of type Select Option is not supported.

SAP R/3 Table Support

The following table describes support for R/3 tables:

R/3 Table Type	Server Support
TRANSPARENT	Supported.
CLUSTER TYPE 1	Type 1 (for example, BSEG). This type of cluster has only one metadata definition (or set of fields), and can be read using Open/SQL statements. The adapter supports this type of cluster.
CLUSTER TYPE 2	Type 2 (a repository, for example, Payroll in HR). In the same physical file, R/3 stores metadata and data. Open/SQL cannot be used to read these clusters. They must be accessed by a specific ABAP program, using R/3 internal macros. The adapter does not support this type of cluster.
CLUSTER TYPE 3	Type 3 (an in-core cluster, for example, Cost Accounting Hierarchy). An R/3-specific program must be written to access these clusters. The adapter does not support this type of cluster.
POOL	Supported.
VIEW	Supported if composed of TRANSPARENT, POOL, or accessible CLUSTER tables.

SAP R/3 Data Type Support

The following table lists the R/3 data types and notes how they are mapped to data types in the Master File.

R/3 Data type	R/3 Length	R/3 Decimals	Usage	Actual
ACCP	N	0	An	An
CHAR	N	0	An	An
CLNT	3	0	A3	A3
CUKY	N	0	An	An
CURR	n	p	P(n+2).p	P(n+1)/2
DATS*	8	0	YYMD or DMYY	A8
DEC	n	p	P(n+2).p	P(n+1)/2
FLTP	n	p	P(n+2).p	P(n+1)/2
INT1	3	0	I3 (limited to 127)	I1
INT2	5	0	I4	I2
INT3	5	0	I4	I3
INT4	10	0	P12.0	P6
LANG	1	0	A1	A1
LCHR	N	n	not supported	
LRAW	N	N	not supported	
NUMC	N	0	An	An
PREC	N	N	not supported	
QUAN	N	P	P(n+2).p	P(n+1)/2
RAW	N	n	not supported	
TIMS	6	0	A6	A6 (00:00:00)
VARC	N	0	An (<=255)	An (<=255)
UNIT	3	0	A3	A3

* The date usage (DATS) is dynamically determined based on the value of the country supplied in the general server configuration. Therefore, date presentations will vary by country.

SAP R/3 Open/SQL Support

The following table describes the conversion that takes places from the FOCUS statements to SAP’s Open/SQL statements.

FOCUS Command	**Corresponding Open/SQL statement**
READLIMIT	UP TO *n* ROWS When a multiple SELECT statement is generated, READLIMIT is translated in UP TO n ROWS for each individual SELECT.
WHERE	Generally translated into a WHERE statement. However, in some cases, due to SAP's SQL buffer limit it might be translated into a CHECK command of ABAP/4.
JOIN TO	For each table included in the JOIN command, embedded SELECT statements are created.
JOIN	Translated into SELECT ... SELECT SINGLE ...
NULL	The Adapter for SAP R/3 does not support the use of the reserved word 'NULL' as part of a WHERE clause. Including 'NULL' as part of a WHERE clause results in a syntax error.
SUM ... BY	By default, the aggregation feature is turned off. If the aggregation is turned off, individual rows are selected and aggregation and sorting is done by FOCUS. It can be turned on by issuing `ENGINE SQLSAP SET OPTIMIZATION SQL` Once the aggregation is turned on, the SUM is translated into SUM and the BY is translated into GROUP BY and ORDER BY. The aggregation feature only applies to TRANSPARENT tables.

Advanced SAP R/3 Features

In this section:

Support for User Security

BAPI Support

Joins Support

This topic describes how to use security, BAPIs, joins, and tracing.

Support for User Security

The Native interface for SAP R/3 has been enhanced to handle the user ID and password provided in

```
ENGINE SQLSAP SET CONNECTION_ATTRIBUTES system/user,password:'client'
```

This feature allows you to run a secured request that involves the following:

- One or more BAPIs, as BAPIs are secured by SAP.
- SAP delivered logical databases, as the server code is secured by SAP.
- Function modules (either SAP or customer) that are fully prototyped, including the proper AUTHORITY-CHECK statements.

BAPI Support

This release supports most read-only BAPIs, including joins, as described in the following example:

```
CREATE SYNONYM baseapp/BUS0002_GETLIST
FOR BUS0002/GETLIST
BAPI DBMS SQLSAP AT I46
END
CREATE SYNONYM baseapp/BUS0002_GETDETAIL
FOR BUS0002/GETDETAIL BAPI
DBMS SQLSAP AT I46
END
JOIN BAPI0002_COMP_CODE IN COMPANYCODE_GETLIST TO
CCGD2_COMP_CODE IN OMPANYCODE_GETDETAIL
END
TABLE FILE COMPANYCODE_GETLIST
PRINT
CCGL0_TYPE NOPRINT
BAPI0002_COMP_CODE
BAPI0002_COMP_NAME
CCGD2_CURRENCY
CCGD2_LANGU
IF BAPI0002_COMP_CODE NE '2300' OR '6000'
END
```

Joins Support

The Native Interface supports all joins from and to SAP. For performance reasons, we do not recommend joining to an SAP data source from a non-SAP data source. It is more efficient to hold the keys in a sequential file, and then use the following code:

```
TABLE FILE SAP PRINT FIELDS IF KEYS IS (HOLD) END
```

Setting Up the Report Processing Mode

SAP R/3 supports many different types of processes (for example, batch, dialog (online), dynamic, print, and so on). Each WebFOCUS for SAP query creates a process within the R/3 application server. This process can execute as either a dialog (online) or batch process.

These processing mode options allow you to adapt to the R/3 application server configuration where there are a specific number of processes allocated for each process type per R/3 application server.

You can change mode settings from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Setting Dialog and Batch Execution Modes

You can set execution processing to dialog (also called online) mode or batch mode.

- **Dialog (or online) processing** is the default mode for the Adapter for SAP R/3. An SAP dialog process is suitable for a task that needs to run immediately at a higher priority. An SAP dialog process is subject to an idle timeout limit where the default value is 15 minutes. Check with your SAP Basis consultant to see if this default value has changed. Dialog process mode should be set for reports that have processing times below the idle timeout limit. If a report's processing time exceeds the dialog process idle timeout limit, the SAP R/3 application server terminates the process and the report fails to run.

  In this mode, the report is executed online. The session is blocked until execution is completed. Execution time is bound to limits preset in the SAP system.

  The following command sets your Adapter for SAP R/3 to dialog processing mode:

  ```
  ENGINE SQLSAP SET EXECUTIONMODE ONLINE
  ```

  You would use this command if you had explicitly set batch execution mode and wanted to switch back to dialog mode within a single session on the server.

If you have a WebFOCUS for SAP report that requires longer processing time, consider Batch processing.

- **Batch processing** is an alternative mode for the Adapter for SAP R/3. An SAP batch process is suitable for a task that can afford to run in the background at a lower priority. This mode is for reports that require longer processing time (for example, reports that involve joining multiple tables or month end reports that need to process a lot of data). In this mode, the report will not time-out due to an execution time limit.

  The following command will set your Adapter for SAP R/3 to batch processing mode:

  ```
  ENGINE SQLSAP SET EXECUTIONMODE BATCH
  ```

Tip: To implement these settings from the Web Console, click Data Adapters in the navigation pane, click the name of a configured adapter, and choose *Change Settings* from the menu. The Change Settings pane opens. Choose *Online* or *Batch* from the Execution mode drop-down menu.

Note that you can also insert these commands directly into the server profile (for example, edasprof.prf) or included within the report procedure.

Setting Dynamic Mode

There are two ways to run the Interface:

- **Dynamic mode ON** executes a report generated via subroutine-pool. This is the standard SAP method for uploading and running dynamic reports, which are deleted after the request is completed. However, this technique precludes setting program attributes, such as the name of the Logical Database, which is required when reporting in Native Mode. ON is the default value.
- **Dynamic mode OFF** enables you to report from LDBs in Native Mode. With this setting, you can catalog the report and set the attributes to the desired LDB before the report is executed.

Tip: To implement this setting from the Web Console, click Data Adapters in the navigation pane, click the name of a configured adapter, and choose *Change Settings* from the menu. The Change Settings pane opens. Choose *On* or *Off* from the Dynamic mode drop-down menu.

Supporting Mixed Code Page Environments

SAP R/3, as well as adapters, can be installed on various platforms. If SAP R/3 is installed on a platform that uses a code page or character set different from the adapter platform, two character conversion tables must be generated to translate the different character sets, one for each direction of data flow.

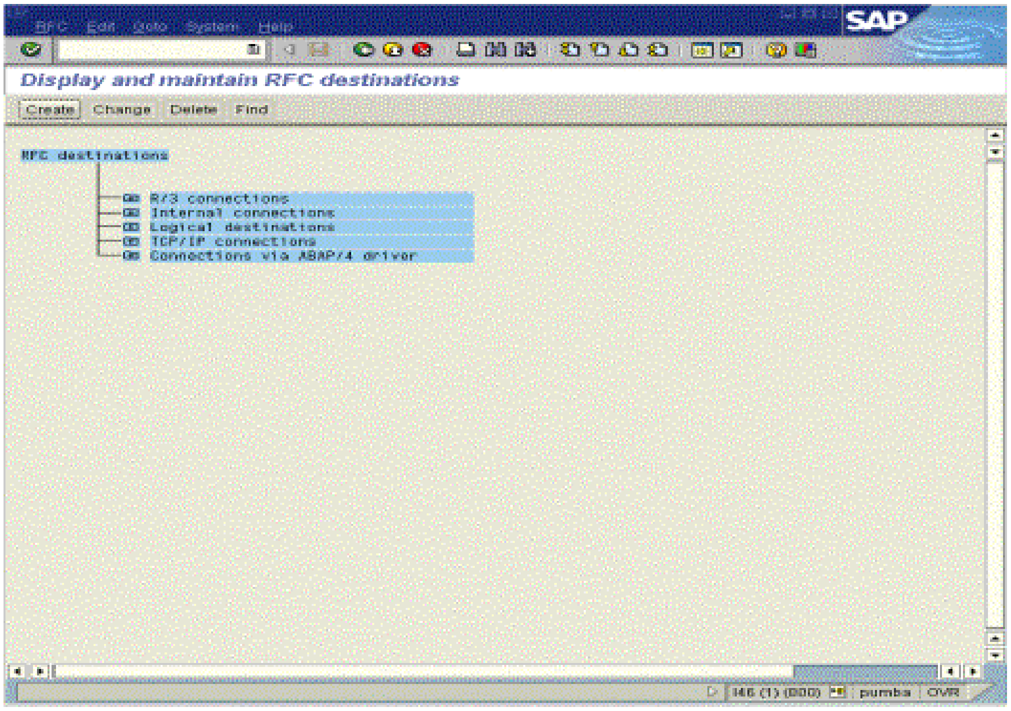
For example, SAP R/3 is installed on an OS/400 EBCDIC machine and the Adapter for SAP R/3 is installed on an Intel Windows ASCII machine. If a request is going from the server to SAP R/3, an ASCII to EBCDIC translation is required. If SAP R/3 is sending data back to the Adapter for SAP R/3, an EBCDIC to ASCII translation is required. This section describes how to create conversion tables and where they must reside.

Character Conversion Tables

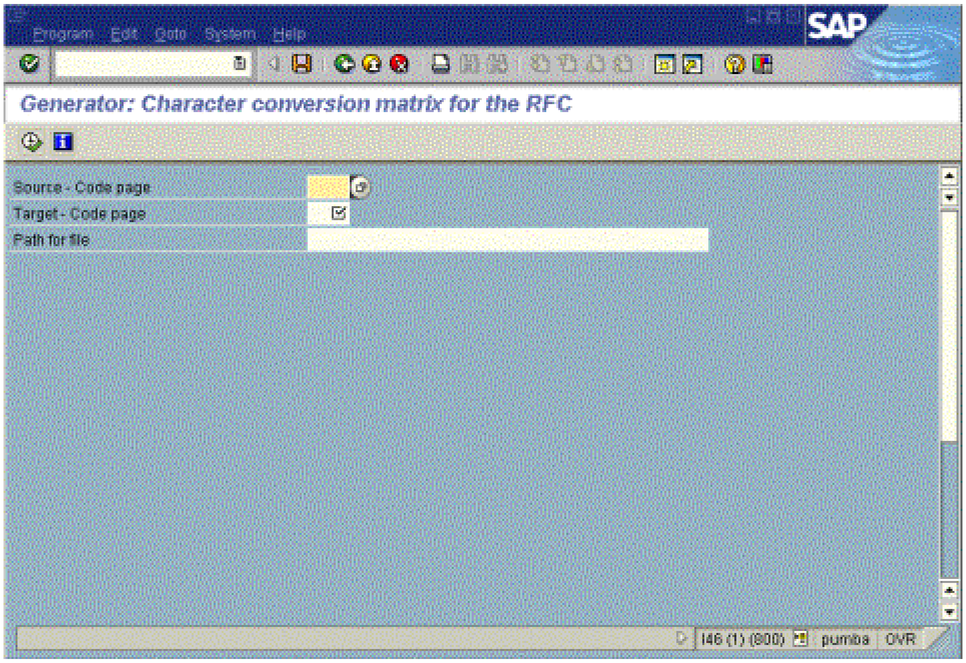
SAP R/3 provides a transaction, SM59, to generate conversion tables for RFC-based applications such as the Adapter for SAP R/3. Once these tables are created, they must be copied to the adapter.

Procedure: How to Generate Conversion Tables

1. Type transaction SM59 at the SAP GUI command line. The following window opens.


RFC Edit Goto System Help
Display and maintain RFC destinations
Create Change Delete Find
RFC destinations
R/3 connections
Internal connections
Logical destinations
TCP/IP connections
Connections via ABAP/4 driver
I46 (1) (800) pumba OVR

2. Select the *Generate Conv.* Tab option from the RFC pull-down menu. The following window opens.



Generator: Character conversion matrix for the RFC

Source - Code page	
Target - Code page	
Path for file	

3. Type a value for the Source code page, Target code page, and a valid path on the R/3 application server where the conversion table file is created. Optionally, you can supply a valid path and a unique file name.

4. Click *Execute* .

   If you did not specify a unique file name, the default file name of the newly created conversion table will contain the source and target code page values. For example, if you specified a source code page of 0102 (OS/400 for some CUAs) and a target code page of 1101 (7 bit USA ASCII pur), the default file name is 01021101.cdp. This is your OS/400 (EBCDIC) to ASCII conversion table file.

   It is now necessary to create another conversion table file for translating code pages in the opposite direction.

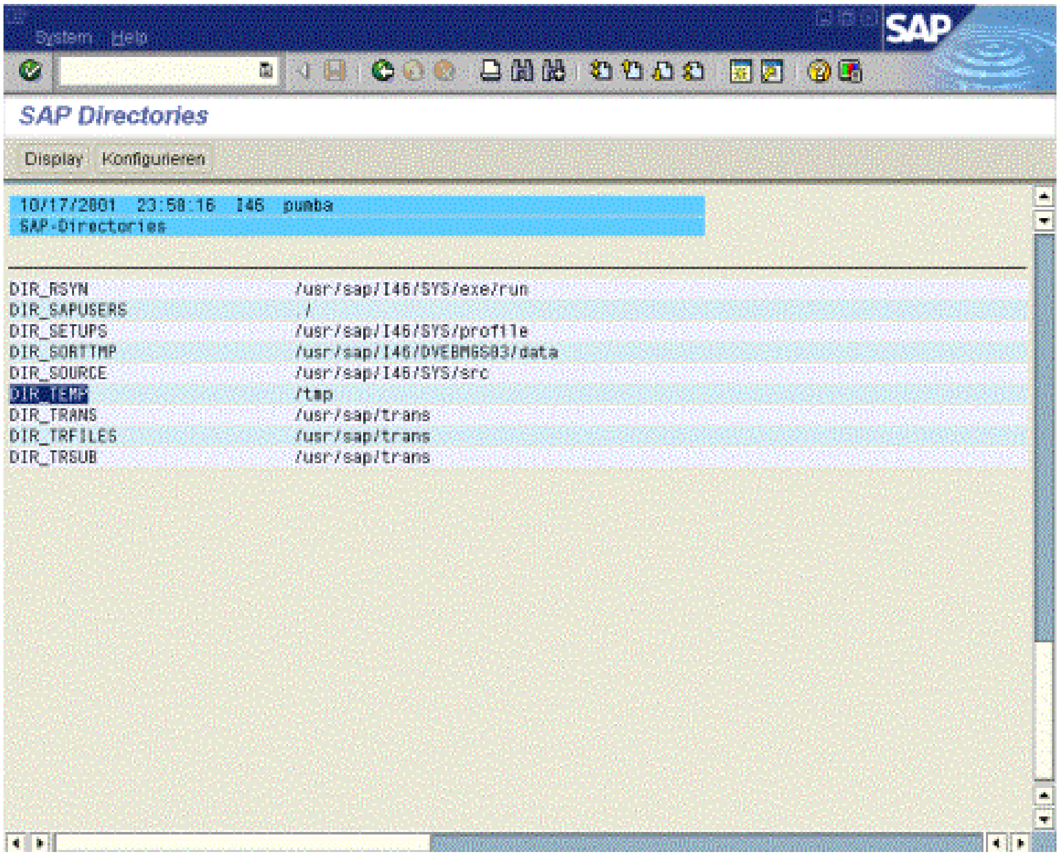
5. Switch the source and target code pages and generate the conversion table.

   Continuing with previous example, you should have a source code page of 1101 and a target code page of 0102, resulting in a default file name of 11010102.cdp. This is your ASCII to OS/400 (EBCDIC) conversion table file.

Note: If a unique file name was used for the first conversion table, be sure you specify a unique file name for the second conversion table.

Procedure: How to Download the Conversion Tables to the Server

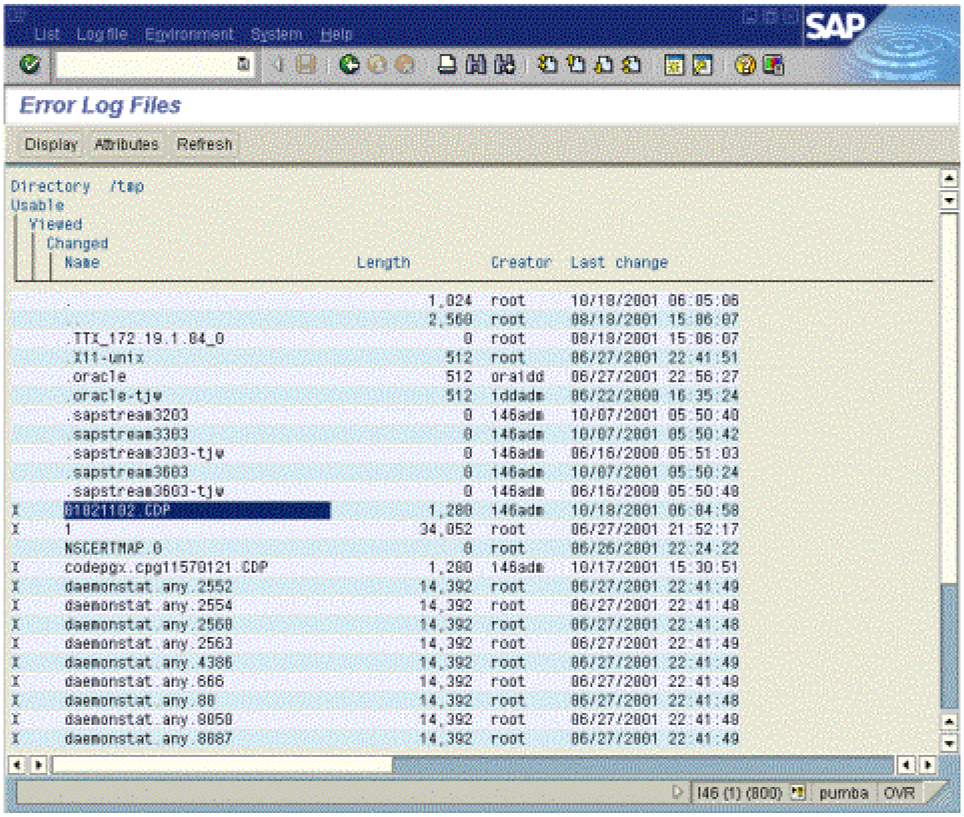
Once these conversion table files are created, you can download them from the SAP application server directory to the server configuration ETC directory. Use SAP R/3 transaction AL11 to browse and download the two conversion table files to a local file on the desktop or Windows machine where the server is installed. Note that SAP GUI is required on the machine to which you are downloading the conversion table files. The following window represents transaction AL11.



```
10/17/2001  23:58:16  I46  pumba
SAP-Directories

DIR_RSYN                      /usr/sap/I46/SYS/exe/run
DIR_SAPUSERS                   ./
DIR_SETUPS                    /usr/sap/I46/SYS/profile
DIR_SORTTMP                   /usr/sap/I46/DVEBMGS03/data
DIR_SOURCE                    /usr/sap/I46/SYS/src
DIR_TEMP                      /tmp
DIR_TRANS                     /usr/sap/trans
DIR_TRFILES                   /usr/sap/trans
DIR_TRSUB                     /usr/sap/trans
```

1. Double-click the directory where the conversion table files were created. The following window opens.

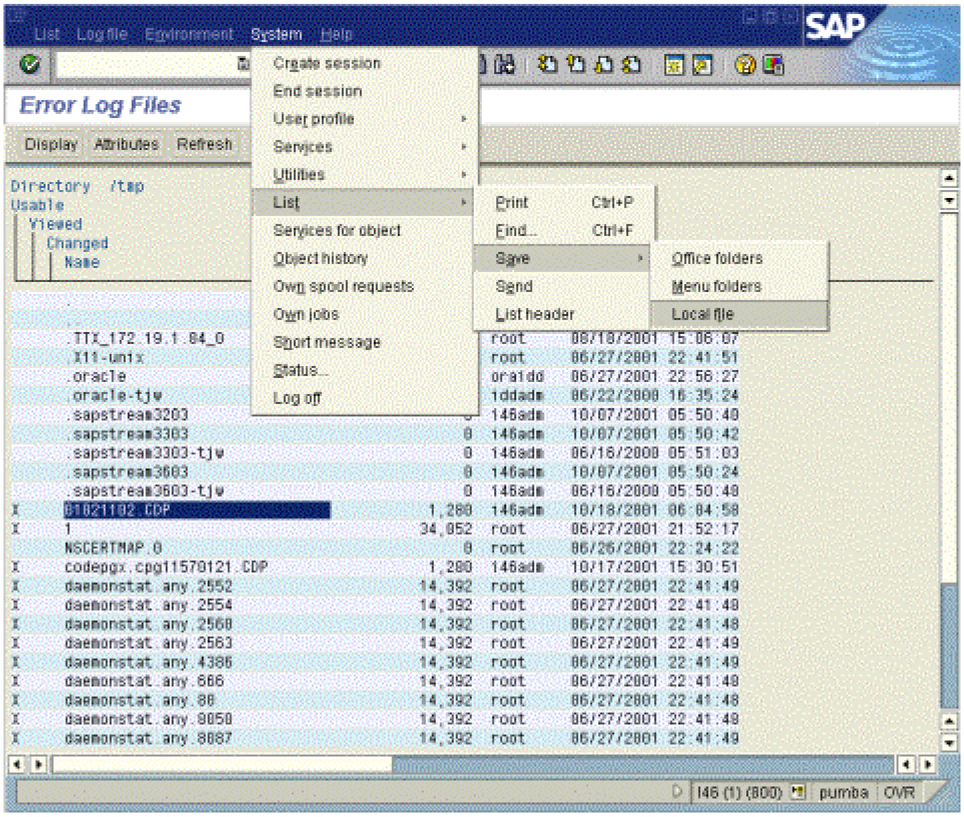


```
Directory  /tmp
Usable
| Viewed
| | Changed
| | | Name                             Length      Creator  Last change

      .                                  1,024    root     10/18/2001 06:05:06
      ..                                 2,560    root     08/18/2001 15:06:07
      .TTX_172.19.1.04_0                     0    root     08/18/2001 15:06:07
      .X11-unix                            512    root     06/27/2001 22:41:51
      .oracle                              512    oraidd   06/27/2001 22:56:27
      .oracle-tjw                          512    iddadm   06/22/2000 16:35:24
      .sapstream3203                         0    146adm   10/07/2001 05:50:40
      .sapstream3303                         0    146adm   10/07/2001 05:50:42
      .sapstream3303-tjw                     0    146adm   06/16/2000 05:51:03
      .sapstream3603                         0    146adm   10/07/2001 05:50:24
      .sapstream3603-tjw                     0    146adm   06/16/2000 05:50:48
X     01021102.CDP                       1,280    146adm   10/18/2001 06:04:58
X     1                                 34,052    root     06/27/2001 21:52:17
      NSCERTMAP.0                            0    root     06/26/2001 22:24:22
X     codepgx.cpg11570121.CDP            1,280    146adm   10/17/2001 15:30:51
X     daemonstat.any.2552               14,392    root     06/27/2001 22:41:49
X     daemonstat.any.2554               14,392    root     06/27/2001 22:41:48
X     daemonstat.any.2560               14,392    root     06/27/2001 22:41:48
X     daemonstat.any.2563               14,392    root     06/27/2001 22:41:49
X     daemonstat.any.4386               14,392    root     06/27/2001 22:41:49
X     daemonstat.any.666                14,392    root     06/27/2001 22:41:48
X     daemonstat.any.88                 14,392    root     06/27/2001 22:41:48
X     daemonstat.any.8050               14,392    root     06/27/2001 22:41:48
X     daemonstat.any.8087               14,392    root     06/27/2001 22:41:49
```

2. Double-click the conversion table file to view its contents.

3. Select the List option from the System pull-down menu and select *Save to Local File* as illustrated in the following window:



```
Directory  /tmp
Usable
 Viewed
  Changed
   Name

    .
    ..
    .TTX_172.19.1.84_0                        root     08/18/2001 15:06:07
    .X11-unix                                 root     06/27/2001 22:41:51
    .oracle                                   oraldd   06/27/2001 22:56:27
    .oracle-tjw                               lddadm   06/22/2000 16:35:24
    .sapstream3203                            146adm   10/07/2001 05:50:40
    .sapstream3303                        0   146adm   10/07/2001 05:50:42
    .sapstream3303-tjw                    0   146adm   06/16/2000 05:51:03
    .sapstream3603                        0   146adm   10/07/2001 05:50:24
    .sapstream3603-tjw                    0   146adm   06/16/2000 05:50:40
X   01021102.CDP                      1,280   146adm   10/18/2001 06:04:58
X   1                                34,052   root     06/27/2001 21:52:17
    NSCERTMAP.0                           0   root     06/26/2001 22:24:22
X   codepgx.cpg11570121.CDP           1,280   146adm   10/17/2001 15:30:51
X   daemonstat.any.2552              14,392   root     06/27/2001 22:41:49
X   daemonstat.any.2554              14,392   root     06/27/2001 22:41:48
X   daemonstat.any.2560              14,392   root     06/27/2001 22:41:48
X   daemonstat.any.2563              14,392   root     06/27/2001 22:41:49
X   daemonstat.any.4386              14,392   root     06/27/2001 22:41:49
X   daemonstat.any.666               14,392   root     06/27/2001 22:41:48
X   daemonstat.any.88                14,392   root     06/27/2001 22:41:48
X   daemonstat.any.8050              14,392   root     06/27/2001 22:41:48
X   daemonstat.any.8087              14,392   root     06/27/2001 22:41:49
```

A dialog box displays prompting on a format for the transfer file.

4. Select unconverted format and click *Continue*.

5. Type a valid file name and click *Transfer*. This transfers the file to your local desktop.

6. Use any available editor on your desktop and remove the first three lines from the conversion table file. The following is an example of the text that should be removed:

```
Directory /tmp
Name: 01021102.CDP
-----------------------------------------------
```

After the edits have been made, configure the server with the Adapter for SAP R/3. Once the adapter is configured, the conversion table files must be moved to the server's EDAHOME ETC directory. Make sure both conversion table files are moved.

Tracing Options

Traces are enabled from the Web Console. Use the Custom option and select one of the following:

- SQLCALL
- SQLSAP/1
- SQLSAP/2
- SQLSAP/3

where:

`level`

Indicates the trace level. Currently supported trace levels are:

- Generated ABAP/4 program.
- Retrieved segments information.

You may also run a diagnostic trace from the SAP system by running transaction ST05 (Performance Trace). See your SAP system Administrator for details.

Producing SAP R/3 Requests

Example:

Retrieving All Records

Retrieving Selected Fields

Joining Two Tables

The following requests and output are examples of the capabilities of the Adapter for SAP R/3 using standard ANSI SQL code. These examples require that CREATE SYNONYM has been performed on the relevant tables from the Web Console. You can use the Web Console to navigate the SAP application hierarchy by searching the relevant tables or by doing a simple search by name.

Example: Retrieving All Records

The following syntax retrieves all records from T014:

```
SELECT * FROM T014;
```

WebFOCUS equivalent of this SQL statement is:

```
TABLE FILE T014
PRINT *
END
```

The output is:

Client	Credit Area	Currency	Update	FYI variant	Risk Categ.	Cred. Limit	Rep. group	All CoCdes
800	0001	EUR	000012			.00		
800	1000	EUR	000012	K4		.00		
800	3000	USD	000012	K4		.00		
800	5000	JPY	000012	K4		.00		
800	6000	MXN	000012	K4		.00		
800	R100	EUR	000012	K4		.00		
800	R300	USD	000012	K4		.00		

Example: Retrieving Selected Fields

The following syntax retrieves selected fields from MARA:

```
SELECT MARA_MATNR, MARA_MTART, MARA_MATKL, MARA_WRKST, MARA_BISMT FROM
MARA WHERE MARA_MATKL='999';
```

The WebFOCUS equivalent of this SQL statement is:

```
TABLE FILE MARA
PRINT MARA_MATNR MARA_MTART MARA_MATKL MARA_WRKST MARA_BISMT
WHERE MARA_MATKL EQ '999';
END
```

The output is:

Material	Matl_type	Matl_group	Basic_matl	Matl_no_
000000000000000038	HALB	999		
AM3-GT	KMAT	999		
BG100001	HALB	999		
KR090654	HALB	999		

Example: Joining Two Tables

The following syntax joins the MARA database with the MARC database:

```
SELECT MARA_MATNR, MARA_WRKST,MARC_WERKS FROM MARA,MARC WHERE
MARA_MATNR=MARC_MATNR AND MARC_WERKS='5100';
```

The WebFOCUS equivalent of this SQL statement is:

```
JOIN CLEAR *
JOIN MARA_MATNR IN MARA TO ALL MARC_MATNR IN MARC AS J0
TABLE FILE MARA
PRINT MARA_MATNR MARA_WRKST MARC_WERKS
WHERE MARC_WERKS EQ '5100';
END
```

The output is:

Material	Basic_matl	Plant
NC-100-212		5100
NC-100-213		5100
NC-100-311		5100
NC-100-312		5100
NC-103-101		5100
NC-103-211		5100

CHAPTER 42

Using the Adapter for Siebel

Topics:

The Adapter for Siebel allows applications to access Siebel data sources. The adapter converts data or application requests into native Siebel statements and returns optimized answer sets to the requesting program.

Software Requirements for the Adapter for Siebel

Verify that Java 2 SDK Version 1.2.2 or higher is installed in your environment by issuing the following command from a command prompt:

```
Java -version
```

If a version is not returned, contact your system administrator.

Preparing the Siebel Environment

To prepare the Siebel environment for adapter configuration, you must:

- Set Siebel Security.
- Access a Siebel Server.

Setting Security

The Adapter for Siebel offers two methods of user authentication:

- **Explicit.** User IDs and passwords are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.
- **Password Passthru.** User ID and password received from the client application are passed to the Siebel Server for authentication. When a client connects to the server, the user ID and password are passed to Siebel for authentication and are not authenticated by the server.

Accessing a Server

Using the standard rules for deploying the Siebel Client, the server supports connections to:

- Local Siebel database servers.
- Remote Siebel database servers.

To connect to a Siebel database server, the server must be pointing to the correct Siebel Gateway Server, Siebel Enterprise Server, Siebel Object Manager, and Siebel Server on the target machine.

Once you are connected to a Siebel database server, that server may define Siebel DATABASE LINKs that can be used to access Siebel tables on other Siebel database servers.

Preparing the Server Environment for Adapter Configuration

To prepare the server environment for adapter configuration, you must define environment variables.

Defining Environment Variables

How to:

Define Configuration Environment Variables on Windows

Define Configuration Environment Variables on UNIX

Example:

Defining Configuration Environment Variables on Windows

Defining Configuration Environment Variables on UNIX

The Adapter for Siebel requires two server environment variables.

Procedure: How to Define Configuration Environment Variables on Windows

1. Ensure that JAVA is in the application path. For example:

   ```
   PATH=c:\jdk1.3.1_05\bin
   ```

2. Point directly to the JVM.dll file in your path.
3. Add the Siebel jar files to the CLASSPATH, using the following syntax:

   ```
   CLASSPATH=jar1;jar2;%CLASSPATH%
   ```

   where:

   jar1

   Is equal to the full path specification of the SiebelJI_Common Java Class file, SiebelJI_Common.jar.

   jar2

   Is equal to the full path specification of the SiebelJI_enu Java Class file, SiebelJI_enu.jar.

Note: Contact your Siebel Administrator for access to these Siebel Java Class files. These files must reside on the same machine as the iWay Server. The files may be copied from a machine that has a Siebel installation.

Example: Defining Configuration Environment Variables on Windows

```
CLASSPATH=D:\IBI\SiebelJl_enu.jar;D:\IBI\SiebelJl_Common.jar
```

Procedure: How to Define Configuration Environment Variables on UNIX

1. Export the JDK_HOME variable to point to the JAVA\bin directory. For example:

   ```
   export JDK_HOME=/usr/java131
   ```

2. Export the Shared Library variable to point to JAVA Home and the directory to LIBJVM shared library. For example:

   ```
   export LD_LIBRARY_PATH=/usr/java131/jre/bin/classic:/usr/java131/jre/bin
   ```

3. Export the Siebel jar files to your CLASSPATH, using the following syntax:

   ```
   export CLASSPATH=jar1:jar2:$CLASSPATH
   ```

   where:

   jar1

   Is equal to the full path specification of the SiebelJI_Common Java Class file, SiebelJl_Common.jar.

   jar2

   Is equal to the full path specification of the SiebelJI_enu Java Class file, SiebelJl_enu.jar.

Note: Contact your Siebel Administrator for access to these Siebel Java Class files. These files must reside on the same machine as the iWay Server. The files may be copied from a machine that has a Siebel installation.

Example: Defining Configuration Environment Variables on UNIX

```
export
CLASSPATH=/u1/ibi/SiebelJl_enu.jar:/u1/ibi/SiebelJl_Common.jar:$CLASSPATH
```

Configuring the Adapter for Siebel

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to Siebel, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Siebel data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Siebel takes place when the first query that references that connection is issued.

If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *ERP* group folder, then expand the *Siebel* adapter folder and click a connection. The Add Siebel to Configuration pane opens.
3. Enter the following parameters:

Field	Description
Connection name	Logical name used to identify this particular set of connection attributes.
Gateway server	Siebel Gateway Server name.
Enterprise server	Siebel Enterprise Server name.
Object Manager	Siebel Object Manager name.
Siebel server	Siebel Server name.
Security	There are two methods by which a user can be authenticated when connecting to an Siebel database server: **Explicit.** The user ID and password are explicitly specified for each connection and passed to Siebel, at connection time, for authentication. **Password Passthru.** The user ID and password received from the client application are passed to Siebel, at connection time, for authentication. This option requires that the server be started with security off.
User	Username by which you are known to Siebel. This field only displays when the security mode of the server is non-trusted.
Password	Password associated with the username. This field only displays when the security mode of the server is non-trusted.

Field	Description
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Tip: Once the adapter is configured, the connection name appears in the navigation pane under *Configured*. If you left-click either the adapter name or the connection name, an appropriate set of options becomes available.

Syntax: How to Declare Connection Attributes Manually

Explicit. The user ID and password are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES connection_name
/userid,password;:'Gateway/Enterprise/Obj Mngr/Siebel'
```

Password Passthru. The user ID and password received from the client application are passed to the Siebel Server for authentication. When a client connects to the server, the user ID and password are passed to Siebel for authentication and are not authenticated by the server.

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES connection_name
:'Gateway/Enterprise/Obj Mngr/Siebel'
```

where:

`SIBIN`

Indicates the Adapter for Siebel.

`connection_name`

Is a logical name used to identify this particular set of connection attributes. In the Access File, it becomes the CONNECTION= attribute.

`userid`

Is the primary authorization ID by which you are known to Siebel.

`password`

Is the password associated with the primary authorization ID.

```
'Gateway/Enterprise/Obj Mngr/Siebel'
```

Is the Siebel connection string. This information must be supplied.

```
Gateway
```

Is the Siebel Gateway Server name.

```
Enterprise
```

Is the Siebel Enterprise Server name.

```
Obj Mngr
```

Is the Siebel Object Manager name.

```
Siebel
```

Is the Siebel Server name.

Example: Declaring Connection Attributes

Explicit authentication:

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES CON1
/USER_ID,PASSWORD;:'ARIBA01/ARIBA01/SCCObjMgr/SiebelSrv'
```

Password Passthru authentication:

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES CON2
:'devportal2/siebel/EAIObjMgr/devportal2'
```

Overriding the Default Connection

Once the connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET CONNECTION_ATTRIBUTES command.

You can also include the CONNECTION attribute in the Access File of the Business Component specified in the current query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION attribute is automatically included in the Access File. This attribute supercedes the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SIBIN SET DEFAULT_CONNECTION connection_name
```

where:

```
SIBIN
```

Indicates the Adapter for Siebel.

`connection_name`

Is the connection name specified in a previously issued SET CONNECTION_ATTRIBUTES command. If this connection name has not been previously declared, a FOC44221 message is issued.

Note: If you use the SET DEFAULT_CONNECTION command more than once, the connection_name specified in the last command will be the active one.

Example: Selecting SIB1 as the Default Connection

The following example selects the previously defined connection named CON1 as the default Siebel connection:

```
ENGINE SIBIN SET DEFAULT_CONNECTION CON1
```

Managing Siebel Metadata

In this section:

Creating Synonyms

Data Type Support

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Siebel data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Business Component (BC) that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata pane opens.

   To create a synonym, you must have configured the adapter. See *Configuring the Adapter for Siebel* on page 42-5 for more information.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter.

4. The right pane displays selection options for the adapter:
   - **Select All Business Objects.** Select this option to create synonyms for all Business objects. This value is the default.
   - **Filter by Name.** Select this option to filter the Business Objects for which you wish to create synonyms.

     When you filter by name, the Business Object Name input box is displayed. Enter a string for filtering the Business Objects, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter ABC% to select Business Objects which begin with the letters ABC; %ABC to select Business Objects which end with the letters ABC; %ABC% to select Business Objects which contain the letters ABC at the beginning, middle, or end.
5. Click *Select Business Objects*. All Business Objects that meet the specified criteria appear.
6. Next, you need to choose the business components to use:
   - **Select All Business Components.** Select this option to create synonyms for all Business Components for every selected Business Object. This value is the default.
   - **Filter by Name.** Select this option to filter the Business Components for which you wish to create synonyms. When you choose filter by name the Business Component Name input box is displayed.

     Enter a string for filtering the Business Component, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter ABC% to select Business Component which begin with the letters ABC; %ABC to select Business Component which end with the letters ABC; %ABC% to select Business Component which contain the letters ABC at the beginning, middle, or end.
7. Click the *Select Business Components* button. All Business Component that meet the specified criteria appear.
8. Select the Business Objects/Business Components for which you would like to create synonyms.
9. If you have Business Objects/Business Components with identical names, assign a prefix to distinguish them. Note that the resulting synonym name cannot exceed 64 characters. If all Business Objects/Business Components have unique names, leave the field blank.
10. From the Select Application drop-down list, select a directory. baseapp is the default value.

11. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

    When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

12. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

13. Complete your Business Object/Business Component selection:

    To select all Business Objects/Business Components in the list, click the check box to the left of the Default Synonym Name column heading.

    To select specific Business Objects/Business Components, click the corresponding check boxes.

14. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

15. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

16. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '| ', '&', ' (', ') ', ' <', '> ', '' ', ' =', ''''

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR business_object/business_component
  DBMS SIBIN [AT connection_name][CHECKNAMES][UNIQUENAMES]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the Siebel Business Component (maximum 64 characters for Windows and UNIX server platforms).

`business_object/business_component`

Is the Siebel Business Object and the Siebel Business Component.

`AT connection_name`

Is the connection_name as previously specified in a SET CONNECTION_ATTRIBUTES command. Default connection is used if this parameter is omitted. When the server creates the synonym, this connection_name becomes the value for CONNECTION= in the Access File.

`CHECKNAMES`

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

`UNIQUENAMES`

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note:

- CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.
- CREATE SYNONYM creates a Master File and Access File which represents the server metadata.

Example: CREATE SYNONYM

The following example creates an ACCOUNT synonym for the ACCOUNT Siebel Business Object and Siebel Business Component.

```
CREATE SYNONYM baseapp/account FOR ACCOUNT/ACCOUNT
DBMS SIBIN AT SIB1
```

Generated Master File account.mas

```
FILENAME=ACCOUNT, SUFFIX=SIBIN    , $
  SEGMENT=ACCOUNT, SEGTYPE=S0, $
    FIELDNAME=VISIBILITY_MODE, ALIAS=VISIBILITY_MODE, USAGE=A15,
ACTUAL=A15,
      ACCEPT='SalesRep' OR 'Manager' OR 'Personal' OR 'All' OR
'Organization' OR 'Contact' OR 'Group' OR 'Catalog' OR 'SubOrganization',
$
    FIELDNAME=ACCOUNT_COMPETITORS, ALIAS='Account Competitors',
USAGE=A500, ACTUAL=A500, MISSING=ON, $
    FIELDNAME=ACCOUNT_CONDITION, ALIAS='Account Condition', USAGE=A500,
ACTUAL=A500, MISSING=ON, $
    FIELDNAME=ACCOUNT_MARKETS, ALIAS='Account Markets', USAGE=A500,
ACTUAL=A500, MISSING=ON, $
    FIELDNAME=ACCOUNT_ORGANIZATION_INTEGRATION_ID, ALIAS='Account
Organization Integration Id', USAGE=A30, ACTUAL=A30, MISSING=ON, $
    FIELDNAME=ACCOUNT_PRODUCTS, ALIAS='Account Products', USAGE=A500,
ACTUAL=A500, MISSING=ON, $
    FIELDNAME=ACCOUNT_ROLE, ALIAS='Account Role', USAGE=A30, ACTUAL=A30,
MISSING=ON, $
.
.
.
    FIELDNAME=MISSION, ALIAS=Mission, USAGE=A500, ACTUAL=A500,
MISSING=ON, $
    FIELDNAME=NAME, ALIAS=Name, USAGE=A100, ACTUAL=A100, MISSING=ON, $
    FIELDNAME=NAME_AND_LOCATION, ALIAS='Name and Location', USAGE=A255,
ACTUAL=A255, MISSING=ON, $
    FIELDNAME=NOT_MANAGER_FLAG, ALIAS='Not Manager Flag', USAGE=A255,
ACTUAL=A255, MISSING=ON, $
.
.
.
```

Generated Access File account.acx

```
SEGNAME=ACCOUNT, CONNECTION=SIB1, SEGNAME_MVG=Account,
  REPOSITORY=Siebel Repository, BS_OBJECT=Account, BS_COMPONENT=Account,$
SEGNAME=CREDIT_CONTROL_AREA_CODE, SEGNAME_MVG=Credit Control Area Code, $
SEGNAME=SYNONYM, SEGNAME_MVG=Synonym, $
SEGNAME=BILL_TO_POSTAL_CODE, SEGNAME_MVG=Bill To Postal Code, $
SEGNAME=BILL_TO_FIRST_NAME, SEGNAME_MVG=Bill To First Name, $
SEGNAME=BILL_ADDRESS_FLAG, SEGNAME_MVG=Bill Address Flag, $
SEGNAME=DEDUP_KEY_UPDATE, SEGNAME_MVG=DeDup Key Update, $
SEGNAME=INDUSTRY, SEGNAME_MVG=Industry, $
SEGNAME=BACK_OFFICE_DISTRIBUTION_CHANNEL,
  SEGNAME_MVG=Back Office Distribution Channel, $
SEGNAME=TYPE_MVF, SEGNAME_MVG=Type MVF, $
SEGNAME=ROW_STATUS, SEGNAME_MVG=Row Status, $
SEGNAME=TERRITORY, SEGNAME_MVG=Territory, $
SEGNAME=AGREEMENT_END_DATE, SEGNAME_MVG=Agreement End Date, $
SEGNAME=SHIP_TO_POSTAL_CODE, SEGNAME_MVG=Ship To Postal Code, $
SEGNAME=SHIP_TO_FIRST_NAME, SEGNAME_MVG=Ship To First Name, $
SEGNAME=TERRITORY_0015, SEGNAME_MVG=Territory, $
```

The Master File root segment describes the Siebel Business Component. All scalar fields of the Business Component are described as fields in the root segment. All Multi Value fields (MVG) of the Business Component are described as descendant segments of the root. The name of the segment representing an MVG is derived from the Business Component Multi Value field name. All other Multi Value fields related to the same Multi Value Link are not shown in the Master File. The MVG segment contains references to the appropriate fields in the linked Business Component instead. All scalar fields of the MVG are described as fields in the MVG segment. All MVG fields of an MVG become descendant segments of the segment representing this parent MVG.

Master File field aliases represent real Siebel Business Component field names. Long names (between 67 and 75 characters) are stored in the Access File. Corresponding Master File fields must have no aliases. Similarly, long segment names are stored in the Access File as well.

A special virtual field VISIBILITY_MODE is added to the root segment to enable the client to set the visibility type for the Business Component. Available visibility modes are enumerated in the ACCEPT parameter for this field. Siebel Versions 6 and 7 support different sets of visibility modes, which is reflected in the ACCEPT list.

The visibility modes are:

Siebel v7: SalesRep, Manager, Personal, All, Organization, Contact, Group, Catalog, and SubOrganization.

Syntax: How to Set Visibility Mode

```
IF VISIBILITY_MODE EQ 'All'
```

This syntax tells Siebel to look at the user ID referenced in the .prf file and according to that user's ID authorization privileges, Siebel returns a view of ALL for the requested output.

If no visibility mode is set in the report, it defaults to the personal visibility mode.

Example: Using Visibility Mode in a Report Request

The following request prints the NAME field where the VISIBILITY MODE is set to All.

```
TABLE FILE ACCOUNT
 PRINT NAME VISIBILITY_MODE
 WHERE ( VISIBILITY_MODE EQ 'All' )
 AND ( NAME LIKE '%WLI%' ) ;
END
```

The output is:

```
 NAME                                                VISIBILITY_MODE
 ----                                                ---------------
 Yelena WLI 81 0129 iway55                           All
 Yelena WLI 81 0129 iway55 JCA2                      All
 Yelena WLI 81 IDE                                   All
```

Example: Using Visibility Mode in a SQL Request

The follow selects NAME and VISIBILITY_MODE based on VISIBILITY MODE and wildcards in NAME.

```
SQL
 SELECT NAME, VISIBILITY_MODE FROM ACCOUNT
 WHERE VISIBILITY_MODE = 'All' AND NAME LIKE '%WLI%'
END
```

The output is:

```
NAME                                     VISIBILITY_MODE
 ----                                     ---------------
 Yelena WLI 81 0129 iway55                All
 Yelena WLI 81 0129 iway55 JCA2           All
 Yelena WLI 81 IDE                        All
```

Reference: Access File Attributes

Keyword	Description
`CONNECTION=`*`connection_name`*	Indicates access to specified Siebel Server. Defaulted to the default connection.
`SEGNAME`	Name of the corresponding segment in the Master File.
`SEGNAME_LONG`	Long (1 to 75 characters) Siebel segment name.
`BS_OBJECT`	Siebel Business Object name.
`BS_COMPONENT`	Siebel Business Component name.
`REPOSITORY`	Siebel Repository.

Data Type Support

All Siebel scalar, alphanumeric, and currency fields are treated as alphanumeric fields. All Siebel date and integer fields are supported.

CHAPTER 43

Using the Adapter for Supra

Topics:

The Adapter for Supra allows applications to access Supra data sources. The adapter converts application requests into native Supra statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

The following topics discuss how the Adapter for Supra is configured and how data is mapped to its corresponding server counterparts in the OS/390 and z/OS environment.

Preparing the Supra Environment

How to:

Customize the LINKEDIT JCL

Customize the Server EDASTART JCL

Allocate CSIPARM

Set the CSISYSIN Parameters

Configure the Adapter Batch Autostart Facility

Define the GATEKEPR Service Block

Install the Adapter for Supra Security Exit (optional)

Utilize the Security Exit

Prior to starting the server, it is necessary to perform the following preconfiguration procedures when preparing the environment on OS/390 and z/OS.

Procedure: How to Customize the LINKEDIT JCL

To customize the LINKEDIT JCL:

1. After executing ISETUP, the JCL will be generated to LINKEDIT the supplied Adapter for Supra module with the Supra stub called CSVILUV. The CSVILUV module is supplied by the Supra vendor, Cincom, and usually resides in the Supra LINKLIB library.

   This required step allows the Adapter for Supra module to communicate with the Supra Central PDM.

2. Modify the following LINKEDIT JCL to conform to site standards and submit it for execution. The JCL listed below displays the GENEFSP JCL that is generated.

```
//LINK1     EXEC PGM=IEWL,PARM='XREF,LET,LIST,MAP,SIZE=1024K'
//SYSLIB    DD   DUMMY
//SYSPRINT DD   SYSOUT=*
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(10,1))
//SUPRA     DD   DSN=cincom.LINKLIB,DISP=SHR
//SYSLMOD   DD   DSN=qualif.TOTAL.LOAD,DISP=SHR
//MAINTAIN DD   DSN=qualif.TOTAL.DATA,DISP=SHR
//SYSLIN    DD *
  INCLUDE SUPRA(CSVILUV)
  INCLUDE SYSLMOD(FSP000)
  INCLUDE MAINTAIN(FSP000)
  NAME FSP000(R)
/*
//
```

where:

`qualif`

Is the high-level qualifier for the server production libraries.

`cincom.LINKLIB`

Is the name of the Supra load library supplied by Cincom that contains the CSVILUV module.

`qualif.TOTAL.LOAD`

Is the name of the adapter load library.

`qualif.TOTAL.DATA`

Is the name of the adapter data library.

Upon successful completion of the LINKEDIT JCL, the adapter module will be ready for use with the server.

Procedure: How to Customize the Server EDASTART JCL

To customize the server EDASTART JCL, add the Supra load libraries to STEPLIB in the server's EDASTART JCL for Supra access.

Note: The Supra DBMS has three load libraries: LINKLIB, INTERFLM, and ENVLIB. These load libraries must be concatenated to ddname STEPLIB.

Procedure: How to Allocate CSIPARM

You must allocate ddname CSIPARM to the data set that contains the CSIPARM definition, which in turn points to the Central PDM you are accessing. Contact your Supra DBA for the name of the file

```
//CSIPARM) DD('cincom.CSIPARM(CONNECT)'),DISP=SHR
```

where:

`cincom`

Is the high-level qualifier for the Supra libraries supplied by Cincom.

Procedure: How to Set the CSISYSIN Parameters

The Adapter for Supra's multi-session facility provides multi-session capability for server tasks to access the Supra Physical Data Manager (PDM) using the Call Database facility or the Relational Data Manager (RDM). It connects with the PDM on the first access by a task. The connecting task is used solely as the connection task and is not usable for other operations. The number of task and concurrent operations supported during the session is controlled by input parameters specified in the data set allocated to ddname CSISYSIN.

There is no security provided by the multi-session adapter. However, the multi-session facility does provide a security exit that can be used to control user access to the database resource. This security exit is identified with the "SECURITY =" keyword described in the next section.

The parameters used for connecting the multi-session adapter to the Central PDM is allocated to ddname CSISYSIN in the server JCL as follows:

```
//CSISYSIN DD DSN=qualif.TOTAL.DATA(SUPRCNFG),DISP=SHR
```

The options available are listed below; xx denotes a value to be supplied by the user. Keywords must start in column 1 and each must be specified on a separate input statement. An asterisk (*) in column 1 indicates a commented line.

Keyword	**Description**
`THREADS=`*`xx`*	Number of concurrent requests that can be issued to the PDM. If this number is too small, a TFUL status may result. See the *Supra PDM Administrator Guide* for more information.
`TASK=`*`xx`*	Maximum number of tasks that can be signed on to the PDM at any one time, including two tasks used for the Autostart facility and for adapter tracking. Therefore, if TASK=10, up to 8 users can be signed on. If this value is exceeded, a CFUL or MFUL status may result. The MFUL status is from the multi-session adapter and indicates that the task tables are full. A CFUL is from the PDM and indicates the same for the PDM.
`SECURITY=`*`xxxxxxx`*	Name of the user written security module that will validate access of users to the PDM data resources. If provided, it must reside in a library accessible from the multi-session adapter.
`DEBUG=`*`xxx`*	Diagnostic facility that provides console messages for problem tracing. Refer to *Adapter Tracing* on page 43-19 for details.
`END`	Ending statement for the parameters. The period is required.

Procedure: How to Configure the Adapter Batch Autostart Facility

When using the Multi-Session Facility for Supra with the server, the first task that issues a request to the Central PDM will be used by the multi-session adapter to establish a permanent connection between the Central PDM and the server. The initial request, if executed from a server client, would tie up that client until the server was shut down. To avoid this scenario, use the Batch Autostart Facility to initialize and establish the connection between the server and Central PDM regions.

Certain parameters in the EDAPROF profile can interfere with the operation of the Batch Autostart Facility. If the service block associated with the Autostart facility (GATEKEPR) executes properly, but no connection is made to Supra, there may be a conflict between your profile and the Autostart facility. To bypass the profile, use the MSOINFO subroutine at the top of your profile to avoid executing the rest of the profile.

The following sample code uses Dialogue Manager to request information about SERVICE and branch out of the profile if appropriate:

```
-SET &REQ = 'SERVICE ';
-SET &ANS = '         ';
-TSO RUN MSOINFO,&REQ,&ANS
-IF &ANS = 'GATEKEPR' GOTO END
```

Note: If you bring up the server before the Supra PDM is up, the Autostart connection will not be established; also, if the PDM is recycled, the Autostart connection will be broken. To connect under these circumstances, use the server console to restart the service block associated with the Autostart task (SERVICE=GATEKEPR).

Procedure: How to Define the GATEKEPR Service Block

The server configuration file must be modified to include an additional service block. This service block is used by the server to execute an exec at startup time which will run as a background task and establish the link between the server and the PDM. The syntax for adding the additional service block is

```
SERVICE = GATEKEPR
BEGIN
   PROGRAM = TSCOM3
   NUMBER_READY = 1
   MAXIMUM = 1
   RUNCOUNT = 5
   AUTOSTART = BATCH
PROFILE = baseapp/supprof
END
```

where:

`SERVICE`

Identifies and names a service. The name that you choose should be unique and from one to eight characters in length. The keywords and values that follow the SERVICE keyword in each service block define the type of service to be provided. GATEKEPR is the recommended name.

`PROGRAM`

This parameter should always be set to TSCOM3.

`NUMBER_READY`

Is the number of instances of this service to be preloaded by the server. This parameter should always be set to 1.

`MAXIMUM`

Is the number of instances of this service allowed by the server. This parameter should always be set to 1.

`RUNCOUNT`

Specifies the number of attempts to connect to the central PDM. There is no default value, therefore you must specify a value to prevent the server from continually attempting to access Supra data.

`PROFILE`

Identifies the file containing a simple request against the Central PDM. The result of this request is the establishment of a multi-session communications link between the server and the Central PDM. Any simple request against a Supra data source can be used for establishing the link.

Note: All parameters must start in column 1 in the configuration file. Any line starting with an asterisk (*) will be considered a comment.

To summarize, the following items are required to implement the Batch Autostart Facility:

- CSISYSIN
- GATEKEPR service block in EDASERVE
- Simple request against Supra data source

Procedure: How to Install the Adapter for Supra Security Exit (optional)

The Adapter for Supra has a security exit, FSPEXT, that allows the user to obtain the account name and account password from a user defined source. The account name and account password are then passed to the adapter and used for login to the Central PDM.

When you issue a request against the Supra database, the adapter calls the FSPEXT function with standard IBM calling conventions, and passes it these parameters:

USR	Passed to function (Server Logon User ID)	8 Bytes Alpha
ACCNT	Returned from function	16 Bytes Alpha
ACCNTP	Returned from function	16 Bytes Alpha

The function returns ACCNT and ACCNTP to the adapter which uses them as parameters to the signon call it issues. The adapter calls the user exit if the password is not available from other sources (for example, the SET command or an Access File). The exit is loaded as a module using a LOAD macro from ddname USERLIB, TASKLIB, or STEPLIB in that order. It is called in AMODE 31. The syntax for the call to the FSPEXT function is:

```
CALL FSPEXT(USR,ACCNT,ACCNTP)
```

Procedure: How to Utilize the Security Exit

To utilize the security exit you must:

1. Write the function using COBOL or Assembler. The function name must be FSPEXT. An example of the FSPEXT function written in Assembler follows.

```
* SAMPLE EXIT FOR Supra ADAPTER

      EJECT
FSPEXT  CSECT
      USING FSPEXT,R15        TEMPORARY ADDRESSABILITY
      B     PROC              JUMP AROUND THE ID INFORMATION
* ID INFORMATION
      DC    CL8'FSPEXT '      PROG NAME
      PRINT NOGEN
PROC STM    R14,R12,12(R13)
      LR    R12,R15
      DROP  R15
      USING FSPEXT,R12
      ST    R13,SAVAR+4
      MVC   8(4,R13),=A(SAVAR)
      LA    R13,SAVAR
*
*     INSERT THE CODE TO FETCH THE ACCOUNT /PASSWORD
       L     R2,4(R1)
       MVC   0(16,R2),=CL16'EXITACCOUNT     '
       L     R2,8(R1)
       MVC   0(16,R2),=CL16'EXITPASS        '
*
       L     R13,SAVAR+4
       LM    R14,R12,12(R13)
       BR    R14
************************************************************
       SAVAR  DS    18F
        R1    EQU   1
        R2    EQU   2
        R9    EQU   9
        R10   EQU   10
        R11   EQU   11
        R12   EQU   12
        R13   EQU   13
        R14   EQU   14
        R15   EQU   15
*
             END
```

2. LINKEDIT the object module into the `*qualif*.TOTAL.LOAD' library with AMODE(31). For example

   ```
   //LINK     EXEC PGM=IEWL,PARM='LET,NCAL,SIZE=(1024K),LIST'
   //OBJLIB   DD   DSN=objlib,DISP=SHR
   //SYSLMOD  DD   DSN=qualif.TOTAL.LOAD,DISP=SHR
   //SYSUT1   DD   UNIT=SYSDA,SPACE=(CYL,(10,1))
   //SYSPRINT DD   SYSOUT=*
   //SYSLIN   DD   *
     INCLUDE OBJLIB(FSPEXT)
     ENTRY FSPEXT
     MODE AMODE(31)
     NAME FSPEXT(R)
   /*
   //
   ```

   where:

   objlib

   Is the name of the library containing FSPEXT object code.

   qualif.TOTAL.LOAD

   Is the adapter load library.

3. If you LINKEDIT the object module into a load library other than `*qualif*.TOTAL.LOAD', concatenate the load library from Step 2 to the STEPLIB DD in your server JCL.

Configuring the Adapter for Supra

Configure the adapter using the Web Console Adapter Add screen and click *Configure*.

Declaring Connection Attributes

In order to access data on the Supra Central Physical Data Manager (PDM), you must have a valid user name and password. You can issue the user name and password to the Central PDM in several ways:

- Hard code the user name and password in the Access File with the USER= and PASSWORD= attributes.
- Issue the Adapter for Supra SET commands from any supported server profile. The syntax is:

  ```
  ENGINE SUPRA SET USER userid
  ```

  ```
  ENGINE SUPRA SET PASSWORD password
  ```

- Invoke the security exit. The security exit, described in *Install the Adapter for Supra Security Exit (optional)* on page 43-7, allows you to code a procedure that returns the user name and password to the Adapter for Supra which are used when the signon call is issued to the Central PDM.

The adapter ascertains the user name and password with the following steps:

1. It checks the Access File.
2. If the user name and password are not coded in the Access File, the adapter checks whether the user issued the SET commands to set the user name and password.
3. If the adapter has not resolved the user name and password using the Access File or SET commands, it invokes the security exit.

Supra Overview and Mapping Considerations

In this section:

This topic explains how the Adapter for Supra allows you to describe and report from Supra database views. You should become familiar with these topics, because most of the concepts affect the way Supra files are described to the server.

Mapping Concepts

Each Supra database can be described as a segment in a Master File. You can describe multiple views in a single Master File if the key of one view is a field in the other view. This embedded cross reference is described with the KEYFLD and IXFLD parameters in the *Access File* on page 43-16.

Supra RDM

A Supra database is a collection of one or more views. The Supra Relational Data Manager (RDM) provides access to physical fields from one or more files in a flat, two-dimensional format. A set of field values is a row. A Supra database consists of one or more views, each of which consists of one or more rows. You can define a subset of rows or reorder the columns according to your needs. This subset of data is called a user view. A column is the smallest logical unit of data a program or user can request.

After creating the view, the database administrator (DBA) defines it in the directory. The DBA can also designate one or more columns as keys to the view. The keys can be used to locate a specific set of rows.

In summary, the terms essential for understanding the data model are as follows:

Term	Definition
View	Set of one or more rows as defined by the Database Administrator (DBA).
User view	Subset of a view which may consist of all or part of the view.
Row	Set of one or more related data items stored in computer memory.
Column	In a row, a specified area used for a particular category of data.
Value	Quantity assigned to a constant, variable, parameter, or symbol.

Term	**Definition**
Key	One or more data items, the contents of which identify the type or location of a row or the ordering of data.

Keys can appear anywhere in the view. The DBA can define different types of keys:

- **Simple Unique Key.** The simplest view has one unique key. This key allows you to select and retrieve data. A unique key must have a valid, non-null value.
- **Compound Unique Key.** The DBA designates several columns as unique logical keys, and the combination of the key values is unique; that is, an "and" connection between the columns is implied. For example, to check customer orders for a certain part number, you would use a view with both customer number and part number as key values. RDM retrieves the specific customer number and part number combination if it is present.
- **Simple Non-unique Key.** This kind of key allows more than one row to contain the same value in a key column. An example is a customer file where a list of notes or comments about each customer is stored, and they are not dated or distinct. In this case, the customer number could be a simple non-unique key. When the program does its first GET using a customer number, it retrieves the first comment for this customer. A subsequent GET retrieves the second, and so on. After RDM retrieves the last comment for that customer, it will reach a boundary condition and return a NOT FOUND to the program.
- **Compound Non-unique Key.** This is an extension of the simple non-unique key. Here, more than one column is defined as a non-unique key. However, all the non-unique keys together still do not completely describe the record occurrence as unique. More than one record with the same compound non-unique key may exist.

You can access a set of rows by assigning values to the keys of the view (if there are any). The DBA determines which columns are keys and defines them on the directory. You can use the keys to locate a specific record or to perform a generic read. You can also access a set of rows sequentially by not supplying any values for the keys.

Mapping of Supra Views and Fields

To map a Supra structure to the server you must set up two files, the Master File and the Access File.

Supra concepts correspond to the following Master File elements:

- One view or user view can be defined as a single segment.
- A column from a Supra view or user view becomes a field.

  The Master File should include only needed columns from the Supra view. Performance may be improved if you do not specify all the columns defined in the view. Even though it is possible to define the columns in an arbitrary order, rearranging key columns may adversely affect performance.
- For each field, the ALIAS must be defined and must be the Supra view column name.

The *Access File* on page 43-16 contains Supra DBMS specific information.

Managing Supra Metadata

In this section:

This topic explains how the Adapter for Supra allows you to describe and report from Supra database views. You should become familiar with these topics, because most of the concepts affect the way Supra files are described to the server.

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Supra data types.

Master File

Reference:

File Keywords

Segment Keywords

Field Keywords

The syntax for describing a Supra database in a Master File is like that of any other Master File and uses the same keywords. A Master File can contain a File record, Segment records, and Field records. The keywords are described in the following sections.

Reference: File Keywords

Keyword	Format or Value
FILENAME	Arbitrary name up to eight characters in length.
SUFFIX	Is SUPRA when accessing RDM or PDM data using the Supra RDM. If RDM is not available, the SUFFIX=TOTIN is used for PDM data access. For more information, see *Supra PDM* on page 43-17.

Reference: Segment Keywords

Keyword	Format or Value
SEGNAME	Segment name.
SEGTYPE	Segment type. S*n* means that the first *n* fields are components of the key, in the order listed. This information is used only if the Access file is missing, or if no value is specified for the KEYNAME attribute of the Access File. If the two values disagree, the value specified in the Access File is retained; a message is displayed only if FSTRACE is allocated to the screen or to a file.
PARENT	Segment name of the parent. If you describe several Supra views or user views as a hierarchical structure in one Master File, then you must define one view as the root segment and all other views as descendant segments. Each descendant segment must include the PARENT attribute to identify its parent in the hierarchy. You must also identify the shared fields for each pair of segments by specifying the KEYFLD and IXFLD attributes in the *Access File* on page 43-16.

Reference: Field Keywords

Keyword	Format or Value
`FIELD`	Any name; 66 characters is the maximum length.
`ALIAS`	Name of the view's column as defined to Supra. The total length must not exceed 66 characters.
`USAGE`	USAGE format values are A (Alpha), I (Integer), F (Floating point), D (Decimal), or P (Packed).
`ACTUAL`	Specifies the actual Supra format (that is, the format defined by the DBA in the view definition that an application program would use for accessing the data). Some examples of Supra formats and their corresponding ACTUAL formats are: • Supra B 1, 2, 4, 8 (with decimal) fields may be defined as F or D. • Supra C 1-32,767 fields may be defined as A (the maximum length of an alphanumeric field is 256 characters). • Supra F 4, 8, 16 fields may be defined as I. • Supra P 1-16 (with decimal) fields may be defined as P. The ACTUAL format does not specify the number of decimal places in the number. The USAGE format defines the number of decimal places; it must be large enough to accommodate all of the digits in the number, the decimal point, and a possible minus sign. For example, if the Supra format is P5 with 2 decimal places, you can use the following specifications: `ACTUAL=P5, USAGE=P8.2` • Supra Z 1–18 (with decimal) fields may be defined as Z.

Note: You can place different views into one Master File. If they have shared fields (fields with the same value in both views), then one view is described as the parent segment, and the other views are described as the child segments.

Access File

The Access File provides the information necessary for selecting the appropriate Supra database, view, and access strategy (that is, the linking of items between different tables).

A logical record in the Access File consists of a list of attributes (keyword = value pairs) separated by commas and terminated by a comma and dollar sign (,$). The list is freeform and can span several lines. The keywords can be specified in any order.

The Access File contains two types of logical records: HEADER and SEGMENT.

Reference: HEADER Logical Record

Keyword	Value
`USER`	Supra user name. If omitted, it defaults to FOCUS.
`PASSWORD`	Supra user password. If omitted, it defaults to FOCUS.

Reference: SEGMENT Logical Record

Keyword	Value
`SEGNAME`	Segment name.
`VIEW`	Supra view (defaults to SEGNAME).
`USERVIEW`	Supra user view corresponding to the segment identified by segname (defaults to VIEW).
`KEYNAME`	Composite of at most 9 field names that constitute the key of the view; the field names are concatenated with the symbol "/", without intervening blanks. The keyword value can span more than one line. If omitted, the first *n* fields, as specified by the SEGTYPE keyword in the Master File, are assumed.
`KEYFLD`	Field name in the parent segment or a defined field that establishes the embedded cross reference. It is mandatory in all but the root segment. Its value is a composite of a maximum of 9 field names concatenated with the symbol "/", without blank spaces. The keyword value can span more than one line.
`IXFLD`	Field name in the descendant segment that establishes the embedded cross reference. It is mandatory for all but the root segment. The values of this field should match the values of the KEYFLD. The keyword value consists of a maximum of 9 field names concatenated with the symbol "/", without blanks. The keyword value can span more than one line.

Embedded Cross-Reference

If your request references fields from two segments defined in the same Master File, the two views are automatically cross-referenced. The fields that implement the cross-reference are specified in the Access File with the KEYFLD and IXFLD attributes. The selection of the shared fields is transparent to you.

The following rules apply a:

- As KEYFLD, you can specify:
  1. Simple field.
  2. List of fields.
- As IXFLD, you can specify a:
  1. Simple field.
  2. List of fields.

In all cases, the length and the format of KEYFLD and IXFLD must be consistent (a list of fields may be joined to a simple field and vice versa, as long as their formats are alphanumeric).

Supra PDM

Accessing PDM data requires using SUFFIX=TOTIN in the Master File.

Supra Modules

In this section:

This topic discusses the pertinent modules for the Adapter for Supra.

Module CFDP4001

The CFDP4001 module is a load module developed for support of the adapter's multi-session facility. This module is LINKEDITed with the name CSTEDBMI. Since this is the same name as the current Cincom single task and central region module, you must place '*qualif*.TOTAL.LOAD', the data set that contains this module, ahead of the LINKLIB and ENVLIB load libraries supplied by Cincom. This module is self contained and is not a composite module like the other CSTEDBMI modules. Make sure not to overlay another Cincom adapter.

Module CFDP4002

The CFDP4002 module is a reusable module that provides the link to the adapter's multi-session facility. It must be linked as reusable so that only one copy is in the address space. The CFDP4001 module loads it when the first PDM request is processed. This module loads the multi-session adapter module (CSTEDBMI) and establishes the connection with the Central PDM. It reads the input parameters to determine the number of task and concurrent operations to be supported during the session. Any errors found will generate a console message. Depending on the error, the connection may be terminated.

Module CFDP4003

The CFDP4003 module is a security module that provides the exit for controlling access to the PDM data resources. It is loaded by module CFDP4002 when the input parameters specify the SECURITY= keyword. It loads the named user module, calls the module using standard IBM calling conventions (register 1 points to the parameter list and register 13 points to the register save area), and passes a security block to the user module.

On return from the called security module, the status field is examined. The operation is accepted if the status field is "****". Any other value results in termination of the operation with a status of "SECR". The user security module may not issue any calls to the PDM. The security block passed to the module is:

Security Block Passed

Field Name	Field Format and Values	Field Length	Starting Position
`USERID`	`Character`	`8`	`1`
`ACCESS`	`Character (R=Read, W=Write)`	`1`	`9`
`DDNAME`	`Character`	`4`	`10`
`DSNNAME`	`Character`	`44`	`14`
`STATUS`	`Character (****=ACCESS ALLOWED)`	`4`	`58`

The library containing these modules should be concatenated to the STEPLIB DD of the server JCL ahead of the LINKLIB and ENVLIB supplied by Cincom. For example,

```
//STEPLIB DD DSN=qualif.TOTAL.LOAD,DISP=SHR
//        DD DSN=cincom.ENVLIB,DISP=SHR
//        DD DSN=cincom.LINKLIB,DISP=SHR
```

where:

`qualif`

Is the high-level qualifier for the server production libraries.

`qualif.TOTAL.LOAD`

Is the name of the adapter load library.

`cincom`

Is the high-level qualifier for the Supra libraries supplied by Cincom.

Adapter Tracing

To activate the Adapter for Supra Tracing Facility, use the Web Console. From the Web console main page, select *Diagnostics*, and then *Traces*. Click *Enable Traces*.

The Default traces will include the Adapter for Supra information.

CHAPTER 44

Using the Adapter for Sybase ASE

Topics:

The Adapter for Sybase ASE allows applications to access Sybase ASE data sources. The adapter converts application requests into native Sybase ASE statements and returns optimized answer sets to the requesting application.

Preparing the Sybase ASE Environment

In order to use the Adapter for Sybase, you must set Sybase and platform-specific environment variables prior to starting the server. Check with your system administrator to see if these values have already been set for you.

Identifying the Location of the Interfaces File

If you are using Sybase inherent distributed access, you must set the SYBASE environment variable to identify the directory where the *interfaces* file resides. The Windows equivalent of the UNIX *interfaces* file is the sql.ini file.

The *interfaces* file is required for both local and remote access. If you do not set the SYBASE environment variable, Sybase searches /etc/passwd for the login directory of the user named Sybase and accesses the *interfaces* file in that directory.

Note that the UNIX PATH must include the location of the Sybase executables. For example, specify the PATH entry as follows for Sybase:

```
SYBASE=/usr/sybase
PATH=$PATH:$SYBASE/bin
export SYBASE PATH
```

You can export the SYBASE environment variable from the UNIX shell or in the UNIX profile of the user's machine that starts the server.

For information about other environment variables needed for Sybase executables and components, see the Sybase Installation and Configuration manual.

Specifying the Sybase Server Name

Use the DSQUERY environment variable to specify the Sybase Server name. The server uses this value only if:

- You do not issue the SET CONNECTION_ATTRIBUTES command in the global server profile (edasprof.prf).
- You do not specify the CONNECTION= attribute in the Access File.

Procedure: How to Specify the Sybase Server Name

```
DSQUERY=server
export DSQUERY
```

where:

`server`

Is the name of the Sybase database server.

Accessing a Remote Sybase Server

Using the standard rules for deploying the Sybase Client, the server supports connections to:

- Local Sybase servers.
- Remote Sybase servers. To connect to a remote Sybase server, the *interfaces* file (*sql.ini* on Windows) must contain an entry pointing to the target machine and the listening process must be running on the target machine.

New entries in the *interfaces* file may be made using the Sybase tool DSEDIT. Entries may also be made automatically using the sybinstall facility. For details, see the Sybase documentation.

XA Support

Read/write applications accessing Sybase ASE data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see Appendix A, *XA Support*.

Configuring the Adapter for Sybase ASE

In this section:

Declaring Connection Attributes

Overriding the Default Connection

Controlling the Connection Scope

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

Example:

Declaring Connection Attributes

In order to connect to the Sybase ASE database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Sybase ASE database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Sybase ASE Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	**Description**
Server	Name of the Sybase server to access. It must match an entry in the Sybase interfaces file.
Security	There are two methods by which a user can be authenticated when connecting to Sybase: **Explicit.** The user ID and password are explicitly specified for each connection and passed to Sybase, at connection time, for authentication. **Password Passthru.** The user ID and password received from the client application are passed to Sybase, at connection time, for authentication. This option requires that the server be started with security off.
User	Primary authorization ID by which the user is known to Sybase.
Password	Password associated with the primary authorization ID. The password is stored in encrypted form.
Database name	Name of the default database for the connection. This value is used when a data object is not qualified with the database name. This parameter is optional. If not specified, it defaults to the database associated with the authorization ID.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Sybase ASE, at connection time, for authentication.

```
ENGINE [SQLSYB] SET CONNECTION_ATTRIBUTES server/userid,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Sybase ASE, at connection time, for authentication. This option requires that the server be started with security off.

```
ENGINE [SQLSYB] SET CONNECTION_ATTRIBUTES  server/
```

where:

`SQLSYB`

Indicates the Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

`server`

Is the name of the Sybase ASE server you wish to access.

`userid`

Is the primary authorization ID by which you are known to Sybase ASE.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Sybase ASE database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Sybase ASE database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

Overriding the Default Connection

How to:

Change the Default Connection

Example:

Selecting the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLSYB] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLSYB`

Indicates the Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Sybase ASE database server named SAMPLENAME as the default Sybase ASE database server:

```
ENGINE SQLSYB SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLSYB] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLSYB`

Indicates the Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Sybase ASE Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Sybase ASE data types.

Creating Synonyms

How to:

Create a Synonym Using the Web Console

Create a Synonym Manually

Example:

Using CREATE SYNONYM

Reference:

Managing Synonyms

Access File Keywords

Synonyms define unique names (or aliases) for each Sybase ASE table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value. See *Controlling the Mapping of Large Character Data Types* for more information.
11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.
12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```
14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.

Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLSYB [AT connection|AT '']
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLSYB

Indicates the Data Adapter for Sybase ASE.

`AT connection`

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`AT ' '`

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local database server.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.nf29004 DBMS SQLSYB AT connsyb NOCOLS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLSYB ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4,     I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.nf29004,
CONNECTION=connsyb,KEYS=1, WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the Sybase ASE table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: `TABLENAME=[owner.]table`
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=connection` CONNECTION=' ' indicates access to the local Sybase ASE database server. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Accessing Different Databases on the Same Sybase Server

When the server connects to Sybase, it uses a primary authorization ID. This ID is associated with a default database on the server. To access tables in another database, you must specify a fully qualified name as follows:

```
database.owner.table
```

This fully qualified name can be used in the SQL request. It *must* be used in the CREATE SYNONYM command for a table in a database other than the default associated with the connected primary authorization ID. If you are allowing access to multiple databases on the same Sybase Server, the user ID specified in the SET CONNECTION_ATTRIBUTES command must have authority to access the database as well as the tables within the database. This can be achieved by using the Sybase stored procedure sp_adduser and the SQL GRANT statement. For more information, see the Sybase SQL Reference manual.

Data Type Support

The following table lists how the server maps Sybase data types. Note that you can:

- Control the mapping of large character data types.
- Change the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Sybase Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
CHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 255
NCHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 255
VARCHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 255
NVARCHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 255
UNICHAR			Unsupported
UNIVARCHAR			Unsupported
TEXT	A32767	A32767	
BINARY (*n*)	A*m*	A*m*	*n* is an integer between 1 and 255 *m* = 2 * *n*
VARBINARY (*n*)	A*m*	A*m*	*n* is an integer between 1 and 255 *m* = 2 * *n*
DATETIME	HYYMDs	HYYMDs	range: 1/1/1753 to 12/31/9999
SMALLDATETIME	HYYMDI	HYYMDI	range: 1/1/1900 to 6/6/2079
TIMESTAMP	A16	A16	Supported as Read-only
INT	I11	I4	range: -2^{31} to $2^{31} - 1$
SMALLINT	I6	I4	range: -2^{15} to $2^{15} - 1$
TINYINT	I6	I4	range: 0 to 255
BIT	I11	I4	"True" or "False"

Sybase Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
DECIMAL (*p, s*)	P*n.m*	P*k*	*p* is an integer between 1 and 38 *s* is an integer between 0 and p If *s* is 0 and *p* is between 1 and 31, $n = p + 1$ If *s* is 0 and *p* is between 32 and 38, $n = 32$ If *s* is greater than 0 and *p* is between 1 and 31, $n = p + 2$ and $m = s$ If *s* is greater than 0 and *p* is between 32 and 38, $n = 33$ and $m = 31$ If *p* is between 1 and 31, $k = (p / 2) + 1$ If *p* is between 32 and 38, $k = 16$ **Note:** If the column is nullable, *p* is greater than or equal to 8.
NUMERIC (*p, s*)	P33.31	P16	*p* is an integer between 1 and 38 *s* is an integer between 0 and *p*
MONEY	D20.2	D8	range: -2^{63} to $2^{63} - 1$
SMALLMONEY	I11	I4	range: -214,748.3647 to 214,748.3648
FLOAT	D20.2	D8	
REAL	D20.2	D8	
IMAGE	BLOB	BLOB	length: $2^{31} - 1$ Supported through iWay API.

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Sybase ASE data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Sybase Data Type	**Remarks**	**LONGCHAR ALPHA or BLOB**		**LONGCHAR TEXT**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
TEXT		A32767	A32767	TX50	TX
BINARY (*n*)	*n* is an integer between 1 and 255 *m* = 2 * *n*	A*m*	A*m*	TX50	TX
VARBINARY (*n*)	*n* is an integer between 1 and 255 *m* = 2 * *n*	A*m*	A*m*	TX50	TX

Syntax: How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLSYB] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

`SQLSYB`

Indicates the Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

`ALPHA`

Maps the Sybase ASE data types TEXT, BINARY, and VARBINARY as alphanumeric (A). ALPHA is the default value.

`TEXT`

Maps the Sybase ASE data types TEXT, BINARY, and VARBINARY as text (TX). Use this value for WebFOCUS applications.

`BLOB`

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Sybase ASE data types TEXT, BINARY, and VARBINARY as alphanumeric (A).

For OS/390 and z/OS, maps the Sybase ASE data types TEXT, BINARY, and VARBINARY as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Sybase ASE data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Sybase Data Type	**Remarks**	**VARCHAR ON**		**VARCHAR OFF**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
VARCHAR (*n*)	*n* is an integer between 1 and 255	A*n*V	A*n*V	A*n*	A*n*
NVARCHAR (*n*)	*n* is an integer between 1 and 255	A*n*V	A*n*V	A*n*	A*n*
VARBINARY (*n*)	*n* is an integer between 1 and 255 *m* = 2 * *n*	A*m*V	A*m*V	A*m*	A*m*

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLSYB] SET VARCHAR {ON|OFF}
```

where:

`SQLSYB`

Indicates the Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Sybase ASE data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (A*n*V).

`OFF`

Maps the Sybase ASE data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). OFF is the default value.

Changing the Precision and Scale of Numeric Columns

How to:

Override Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLSYB] SET CONVERSION RESET
ENGINE [SQLSYB] SET CONVERSION format RESET
ENGINE [SQLSYB] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLSYB] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLSYB`

Indicates the Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`REAL` which indicates that the command applies only to single precision floating point columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT and REAL data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLSYB SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLSYB SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLSYB SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLSYB SET CONVERSION RESET
```

Customizing the Sybase ASE Environment

In this section:

The Adapter for Sybase ASE provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Activating NONBLOCK Mode

The Adapter for Sybase ASE has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE [SQLSYB] SET NONBLOCK {0|n}
```

where:

`SQLSYB`

Indicates the Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLSYB] SET PASSRECS {ON|OFF}
```

where:

`SQLSYB`

Indicates the Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLSYB] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLSYB`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

OFF instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

ON instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

FOCUS passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

SQL passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLSYB set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLSYB set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLSYB] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLSYB`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLSYB SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLSYB SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLSYB SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Specifying Block Size for Retrieval Processing

How to:

Specify Block Size for Array Retrieval

Specify Block Size for Insert Processing

Suppress the Bulk Insert API

Reference:

Bulk Insert API Behavior

The Adapter for Sybase ASE supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Specify Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE [SQLSYB] SET FETCHSIZE n
```

where:

`SQLSYB`

Indicates the Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Syntax: How to Specify Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE [SQLSYB] SET INSERTSIZE n
```

where:

`SQLSYB`

Indicates the Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Syntax: How to Suppress the Bulk Insert API

```
ENGINE [SQLSYB] SET FASTLOAD [ON|OFF]
```

where:

`SQLSYB`

Indicates the Adapter for Sybase ASE. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Uses the Bulk Insert API. ON is the default.

`OFF`

Suppresses the use of the Bulk Insert API.

Reference: Bulk Insert API Behavior

You can use DataMigrator with the Bulk Insert API for Sybase ASE.

With the Adapter for Sybase, the Bulk Insert API is used in LOADONLY mode when INSERTSIZE is greater than 1. INSERTSIZE determines how often the intermediate data flush is performed. Measurements show that intermediate flushes are necessary for optimal Sybase server performance. As a rule of thumb, the INSERTSIZE value should be from several thousand to several tens of thousand. Intermediate flushes cannot be rolled back.

When you suppress the Bulk Load API, the Adapter for Microsoft SQL Server resorts to array insert according to the specified INSERTSIZE.

Errors that occur during the load (such as duplication) can cause the batch of rows to be rejected as a whole.

Calling a Sybase ASE Stored Procedure Using SQL Passthru

Sybase stored procedures are supported using SQL Passthru. These procedures need to be developed within Sybase using the CREATE PROCEDURE command.

Syntax: How to Call a Sybase Stored Procedure

The supported syntax to call a stored procedure is shown below. It is recommended that you use the syntax below instead of the previously supported CALLSYB syntax.

```
ENGINE SQLSYB
EX SAMPLE PARM1,PARM2,PARM3...;
TABLE ON TABLE PCHOLD
END
```

where:

`SQLSYB`

Indicates the Adapter for Sybase. You can omit this value if you previously issued the SET SQLENGINE command.

Example: Sybase Stored Procedure

```
CREATE PROCEDURE SAMPLE
AS
SELECT SSN5, LAST_NAME5, FIRST_NAME5, BIRTHDATE5, SEX5 FROM EDAQA.NF29005
go
exec sp_procxmode 'PROC1','anymode'
go
```

CHAPTER 45

Using the Adapter for Sybase IQ

Topics:

The Adapter for Sybase IQ allows applications to access Sybase IQ data sources. The adapter converts application requests into native Sybase IQ statements and returns optimized answer sets to the requesting application.

Preparing the Sybase IQ Environment

In order to use the Adapter for Sybase IQ, you must set Sybase and platform-specific environment variables prior to starting the server. Check with your system administrator to see if these values have already been set for you.

Identifying the Location of the Interfaces File

If you are using Sybase inherent distributed access, you must set the SYBASE environment variable to identify the directory where the interfaces file resides. The Windows equivalent of the UNIX interfaces file is the sql.ini file.

The interfaces file is required for both local and remote access. If you do not set the SYBASE environment variable, Sybase searches /etc/passwd for the login directory of the user named Sybase and accesses the interfaces file in that directory.

Note that the UNIX PATH must include the location of the Sybase executables. For example, specify the PATH entry as follows for Sybase:

```
SYBASE=/usr/sybase
PATH=$PATH:$SYBASE/bin
export SYBASE PATH
```

You can export the SYBASE environment variable from the UNIX shell or in the UNIX profile of the user's machine that starts the server.

For information about other environment variables needed for Sybase executables and components, see the Sybase Installation and Configuration manual.

Specifying the Sybase Server Name

Use the DSQUERY environment variable to specify the Sybase Server name. The server uses this value only if you do not:

- Issue the SET CONNECTION_ATTRIBUTES command in the global server profile (edasprof.prf).
- Specify the CONNECTION= attribute in the Access File.

Procedure: How to Specify the Sybase Server Name

```
DSQUERY=server
export DSQUERY
```

where:

server

Is the name of the Sybase database server.

Accessing a Remote Sybase Server

Using the standard rules for deploying the Sybase Client, the server supports connections to:

- Local Sybase servers.
- Remote Sybase servers. To connect to a remote Sybase server, the interfaces file (sql.ini on Windows) must contain an entry pointing to the target machine and the listening process must be running on the target machine.

New entries in the interfaces file may be made using the Sybase tool DSEDIT. Entries may also be made automatically using the sybinstall facility. For details, see the Sybase documentation.

Configuring the Adapter for Sybase IQ

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to the Sybase IQ database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Sybase IQ database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Sybase IQ Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	**Description**
Server	Name of the Sybase server to access. It must match an entry in the Sybase interfaces file.
Security	There are two methods by which a user can be authenticated when connecting to Sybase: **Explicit.** The user ID and password are explicitly specified for each connection and passed to Sybase, at connection time, for authentication. **Password Passthru.** The user ID and password received from the client application are passed to Sybase, at connection time, for authentication. This option requires that the server be started with security off.
User	Primary authorization ID by which the user is known to Sybase.
Password	Password associated with the primary authorization ID. The password is stored in encrypted form.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Sybase IQ, at connection time, for authentication.

```
ENGINE [SQLSYB] SET CONNECTION_ATTRIBUTES server/userid,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Sybase IQ, at connection time, for authentication. This option requires that the server be started with security off.

```
ENGINE [SQLSYB] SET CONNECTION_ATTRIBUTES  server/
```

where:

`SQLSYB`

Indicates the Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

`server`

Is the name of the Sybase IQ server you wish to access.

`userid`

Is the primary authorization ID by which you are known to Sybase IQ.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Sybase IQ database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Sybase IQ database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLSYB] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLSYB`

Indicates the Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Sybase IQ database server named SAMPLENAME as the default Sybase IQ database server:

```
ENGINE SQLSYB SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLSYB] SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLSYB`

Indicates the Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Sybase IQ Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Sybase IQ data types.

Creating Synonyms

How to:

Create a Synonym Using the Web Console

Create a Synonym Manually

Example:

Using CREATE SYNONYM

Reference:

Managing Synonyms

Access File Keywords

Synonyms define unique names (or aliases) for each Sybase IQ table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value. See *Controlling the Mapping of Large Character Data Types* for more information.

11. Complete your table or view selection:

To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

Synonyms are created and added under the specified application directory.

A status window displays the message:

```
All Synonyms Created Successfully
```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.

Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLSYB [AT connection|AT '']
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

`SQLSYB`

Indicates the Data Adapter for Sybase IQ.

`AT` *`connection`*

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`AT ' '`

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local database server.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.nf29004 DBMS SQLSYB AT connsyb NOCOLS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLSYB ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4,     I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.nf29004,
CONNECTION=connsyb,KEYS=1, WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Identifies the Sybase IQ table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: `TABLENAME=[owner.]table`
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=connection` CONNECTION=' ' indicates access to the local Sybase IQ database server. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Accessing Different Databases on the Same Sybase Server

When the server connects to Sybase, it uses a primary authorization ID. This ID is associated with a default database on the server. To access tables in another database, you must specify a fully qualified name as follows:

```
database.owner.table
```

This fully qualified name can be used in the SQL request. It *must* be used in the CREATE SYNONYM command for a table in a database other than the default associated with the connected primary authorization ID. If you are allowing access to multiple databases on the same Sybase Server, the user ID specified in the SET CONNECTION_ATTRIBUTES command must have authority to access the database as well as the tables within the database. This can be achieved by using the Sybase stored procedure sp_adduser and the SQL GRANT statement. For more information, see the Sybase SQL Reference manual.

Data Type Support

The following table lists how the server maps Sybase data types. Note that you can:

- Control the mapping of large character data types.
- Change the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Sybase Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
CHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 255
VARCHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 255
BINARY (*n*)	A*m*	A*m*	*n* is an integer between 1 and 255 *m* = 2 * *n*
VARBINARY (*n*)	A*m*	A*m*	*n* is an integer between 1 and 255 *m* = 2 * *n*
DATE	HYYMDs	HYYMDs	
TIME	HYYMDs	HYYMDs	
TIMESTAMP	A16	A16	range: 0001-01-01 00:00:00.000000 to 9999-12-31 23:59:59.999999 Supported as Read-only.
INTEGER	I11	I4	range: -2^{31} to $2^{31} - 1$
UNSIGNED INT	P11	P8	range: 0 to $2^{32} - 1$
BIGINT	P21	P11	range: 2^{63} to $2^{63} - 1$
UNSIGNED BIGINT	P22	P11	range: 0 to $2^{64} - 1$
SMALLINT	I6	I4	range: -2^{15} to $2^{15} - 1$
TINYINT	I6	I4	range: 0 to 255
BIT	I11	I4	"True" or "False"

Sybase Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
DECIMAL (*p*, *s*)	P33.3	P16	*p* is an integer between 1 and 126 *s* is an integer between 0 and p
NUMERIC (*p*, *s*)	P33.31	P16	*p* is an integer between 1 and 126 *s* is an integer between 0 and p
FLOAT	D20.2	D8	range: 1.175494351e-38 thru 3.40282366e+38
REAL	D20.2	D8	range: 1.175494351e-38 thru 3.40282366e+38
DOUBLE	D20.2	D8	range: 2.22507385850720l4e-308 thru 1.797693134862315708e+308

Note: User-defined data types are not supported. For specifications of Sybase IQ data types, see the Sybase IQ Administrator's manual.

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Sybase IQ data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Sybase Data Type	**Remarks**	**LONGCHAR ALPHA or BLOB**		**LONGCHAR TEXT**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
BINARY (*n*)	*n* is an integer between 1 and 255 *m* = 2 * *n*	A*m*	A*m*	TX50	TX
VARBINARY (*n*)	*n* is an integer between 1 and 255 *m* = 2 * *n*	A*m*	A*m*	TX50	TX

Syntax: How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLSYB] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

`SQLSYB`

Indicates the Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

`ALPHA`

Maps the Sybase IQ data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A). ALPHA is the default value.

`TEXT`

Maps the Sybase IQ data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as text (TX). Use this value for WebFOCUS applications.

`BLOB`

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Sybase IQ data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as alphanumeric (A).

For OS/390 and z/OS, maps the Sybase IQ data types CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, and RAW as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Sybase IQ data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Sybase Data Type	**Remarks**	**VARCHAR ON**		**VARCHAR OFF**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
VARCHAR (*n*)	*n* is an integer between 1 and 255	A*n*V	A*n*V	A*n*	A*n*
VARBINARY (*n*)	*n* is an integer between 1 and 255 *m* = 2 * *n*	A*m*V	A*m*V	A*m*	A*m*

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLSYB] SET VARCHAR {ON|OFF}
```

where:

`SQLSYB`

Indicates the Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Sybase IQ data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (A*n*V).

`OFF`

Maps the Sybase IQ data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). OFF is the default value.

Changing the Precision and Scale of Numeric Columns

How to:

Override Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override Default Precision and Scale

```
ENGINE [SQLSYB] SET CONVERSION RESET
ENGINE [SQLSYB] SET CONVERSION format RESET
ENGINE [SQLSYB] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLSYB] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLSYB`

Indicates the Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`REAL` which indicates that the command applies only to single precision floating point columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT and REAL data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLSYB SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLSYB SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLSYB SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLSYB SET CONVERSION RESET
```

Customizing the Sybase IQ Environment

In this section:

The Adapter for Sybase IQ provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Activating NONBLOCK Mode

The Adapter for Sybase IQ has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE [SQLSYB] SET NONBLOCK {0|n}
```

where:

`SQLSYB`

Indicates the Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Note that since, by definition, the successful execution of an INSERT command always affects one record, INSERT does not generate this information.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLSYB] SET PASSRECS {ON|OFF}
```

where:

`SQLSYB`

Indicates the Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

In this section:

Optimizing Requests

Optimizing Requests Containing Virtual Fields

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

How to:

Optimize Requests

Example:

SQL Requests Passed to the RDBMS With Optimization OFF

SQL Requests Passed to the RDBMS With Optimization ON

Reference:

SQL Generation in Optimization Examples

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLSYB] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLSYB`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

`setting`

Is the optimization setting. Valid values are as follows:

`OFF` instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

`ON` instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

`FOCUS` passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

`SQL` passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLSYB set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLSYB set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

How to:

Optimize Requests Containing Virtual Fields

Example:

Using IF-THEN_ELSE Optimization Without Aggregation

Using IF-THEN_ELSE Optimization With Aggregation

Using IF-THEN_ELSE Optimization With a Condition That is Always False

Reference:

SQL Limitations on Optimization of DEFINE Expressions

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLSYB] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLSYB`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLSYB SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLSYB SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLSYB SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Calling a Sybase IQ Stored Procedure Using SQL Passthru

Sybase stored procedures are supported using SQL Passthru. These procedures need to be developed within Sybase using the CREATE PROCEDURE command.

Syntax: How to Call a Sybase Stored Procedure

The supported syntax to call a stored procedure is shown below. It is recommended that you use the syntax below instead of the previously supported CALLSYB syntax.

```
ENGINE SQLSYB
EX SAMPLE PARM1,PARM2,PARM3...;
TABLE ON TABLE PCHOLD
END
```

where:

`SQLSYB`

Indicates the Adapter for Sybase IQ. You can omit this value if you previously issued the SET SQLENGINE command.

Example: Sybase Stored Procedure

```
CREATE PROCEDURE SAMPLE
AS
SELECT SSN5, LAST_NAME5, FIRST_NAME5, BIRTHDATE5, SEX5 FROM EDAQA.NF29005
go
exec sp_procxmode 'PROC1','anymode'
go
```

CHAPTER 46

Using the Adapter for System 2000

Topics:

The Adapter for System 2000 allows applications to access System 2000 data sources.

The adapter converts application requests into native System 2000 statements and returns optimized answer sets to the requesting application.

Preparing the System 2000 Environment

The Adapter for System 2000 operates in the OS/390 and z/OS environment and issues standard System 2000 calls for record retrieval. The adapter uses System 2000's Boolean Selection capabilities, enabling it to retrieve only records which satisfy a request. This reduces the number of I/Os involved in data retrieval.

Prior to configuring the adapter using the Web Console, you must allocate the System 2000 load library data set to DDNAME STEPLIB of the server's ISTART JCL.

The adapter is compatible with Release 10.1 of System 2000 and higher, as long as compatibility at the object and source levels are guaranteed for System 2000 applications.

The module S2KPL must be link-edited into the adapter at installation time.

Procedure: How to Generate the Adapter

Use the following code to generate the Adapter for System 2000.

```
* * * Top of File * * *
/*********************************************************************
//* Name:     GENES2K
//*
//* Function: Linkedit System2000 stubs into the S2K Interface
//*
//* Substitutions:-Change "system2000.LOAD" to the name of your
//*                 System 2000 load library.
//*                -Change "qualif" to the high level qualifier of
//*
//*********************************************************************
//*
//LKED     EXEC PGM=IEWL,PARM='LIST,NOXREF,LET,MAP'
//S2K      DD   DISP=SHR,DSN=system2000.LOAD
//SYSLMOD  DD   DISP=SHR,DSN=qualif.EDALIB.LOAD
//MAINTAIN DD   DISP=SHR,DSN=qualif.EDALIB.DATA
//SYSUT1   DD   UNIT=SYSDA,SPACE=(100,(50,50))
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   *
  INCLUDE S2K(S2KPL)
  INCLUDE SYSLMOD(S2K)
  INCLUDE MAINTAIN(S2K)
  NAME S2K(R)
/*
```

where:

`system2000.LOAD`

Is the name of the System 2000 load library.

`qualif`

Is the high-level qualifier for the data sets.

Configuring the Adapter for System 2000

To configure the adapter, you simply identify it for configuration from the Web Console.

Declaring Connection Attributes

The Adapter for System 2000 supports element-level security by allowing specification of System 2000 passwords in the Access File. If an element is defined in the System 2000 control file with a security code, you must specify the two-character hex value of the code in the Access File.

The password for a System 2000 data source must be located in the Access File. The password must have R (read) authority for each field described in the Master File, and W (where) authority for each field used in an IF/WHERE statement. You can encrypt the Access File if desired.

Procedure: How to Configure the Adapter for System 2000 From the Web Console

1. Select the *Adapter Add* option from the main screen and select *System 2000* from the list provided under Add.
2. Click *Configure*.

Mapping Considerations

This topic explains how the Adapter for System 2000 allows you to describe and report from a System 2000 data source structure. These concepts affect the way System 2000 files are described to the server.

Rules for Mapping

Since System 2000 is a hierarchical database management system, the mapping from a System 2000 structure into a Master File on the server is simple. The rules are:

- A System 2000 subschema record (SSR) becomes a Master File segment.
- Any subtree may be represented as a Master File structure in one of two ways:
  - A System 2000 parent-child relationship may be defined as a parent-child relationship in the Master File.
  - A System 2000 parent-child relationship may be defined as a child-parent relationship (that is, inversion or alternate view) in the Master File.
- System 2000 items become fields in a Master File. Only required fields must be described. Within each Master File segment, the fields may be described in any order.

Retrieval Subtree

The features of System 2000 and the available navigation strategies make it possible to define any subtree. You can even define subtrees with missing intermediate levels, as long as such subtrees are meaningful.

You can create alternate views, or inversions of parent-child relationships, in the Master File or request them at run time. Note, however, that inversions are permitted only if a segment is identified as the root of a System 2000 data source tree (or subtree).

You can identify roots either as the segment named ENTRY or as the segment named in the Access File as the ENTRYNAME parameter value. If no root segment is identified in the Master File, the adapter assumes a subtree of a System 2000 data source. If an inversion is then attempted, errors may occur during retrieval.

Managing System 2000 Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a Master File and an Access File that describes the structure of the data source and the server mapping of the System 2000 data types.

Master File

Once you have decided how to map a System 2000 structure on the server, your next task is to set up two new files—the Master File and the Access File.

The Master File can contain a File record, Segment records, and Field records. The keywords are described in the reference sections that follow. For an illustration, see *Sample Data Descriptions for System 2000* on page 46-9.

Reference: File Keywords

Keyword	Format or Value
`FILENAME`	Server name for the System 2000 data source and the Master File that describes it. The same as the member name of the Master File in the MASTER PDS. Eight characters maximum.
`SUFFIX`	S2K (mandatory). This is the name of the adapter program that the server loads to read System 2000 data sources.

Reference: Segment Keywords

Keyword	Format or Value
`SEGNAME`	Segment name, which must correspond to the System 2000 subschema record (SSR) name, if the name does not exceed eight characters, or the SSR C-num. C-num indicates the System 2000 item number (C followed by up to four digits which may include leading zeros (for example, C3 is equivalent to C03, CO03, or COO03). SEGNAME may include embedded blanks. You identify the root segment of the System 2000 data source (or the topmost segment of the subtree) to the adapter by coding its segment name as the value of the parameter ENTRYNAME in the ACCESS File. See *Access File Attributes* on page 46-7.
`PARENT`	Segment name of the parent.

Reference: Field Keywords

Keyword	Format or Value
`FIELD`	Any name; 66 characters is the maximum length.
`ALIAS`	System 2000 item C-num, or its name (if its length does not exceed 12 characters). Embedded blanks are allowed.
`USAGE`	Data format of the field as processed by the server. Usage formats are: A, I, F, D, P (alphanumeric, integer, floating, decimal, packed). The usage format can be different from the actual format depending on the actual format used. If the field is numeric with decimal places, the usage format must specify the exact number of decimal places. See *Data Type Support* on page 46-8.
`ACTUAL`	Data format of the field as it is in the System 2000 data source. Specify A (for character or date fields) or P (for numeric fields). See *Data Type Support* on page 46-8.

Access File

The Access File provides the information necessary for selecting the appropriate System 2000 data source and access strategy (that is, the linking of items between different tables). It provides the password and specifies whether to use sequential scan or selective access to read records that satisfy the IF screening conditions. (See *Navigation Strategies* on page 46-14 for details.) You may also specify a TRACE parameter which is useful for debugging purposes.

For an illustration, see *Sample Data Descriptions for System 2000* on page 46-9.

Reference: Access File Attributes

A logical record in the Access File consists of a list of keyword and value pairs, separated by commas and terminated by a comma and a dollar sign (,$). The list is freeform and may span several lines. The keywords may be specified in any order.

One of the logical records in the Access File must specify the System 2000 data source name and its password, which can be encrypted.

Keyword	Value
`FILENAME`	Name of the corresponding Master File.
`DBNAME`	Name of the S2K data source (mandatory).
`PASSWORD`	Password required to access a S2K data source (mandatory, may be encrypted).
`ACCESS`	Specifies an access strategy: SEQ (sequential) or SEL (selective). • Sequential access means that the System 2000 data source is accessed using a sequence of GET1 FIRST, GETD, and GET1 NEXT commands. The screening conditions are managed by the server. • Selective access means that the System 2000 data source is accessed using a LOCATE command and a sequence of GETD and GET NEXT commands. If translatable screening conditions are not specified, the adapter performs sequential access. If this value is omitted, the access method defaults to SEQ.
`TRACE`	YES or NO. If YES is specified, the server provides a complete trace of all executed System 2000 commands. This option is recommended only for debugging purposes. If this value is omitted, NO is the default value.

Keyword	Value
`ENTRYNAME`	Name of the topmost segment of the referenced System 2000 subtree. If this value is omitted, ENTRY is the default value. Note that alternate views, or inversions, are allowed only if a segment with the name specified by this parameter (whether explicitly or by default) is found in the Master File.
`MODE`	Specifies whether System 2000 is being used in multi-user or single-user mode. May be specified as M for multi-user or S for single-user.

Data Type Support

The usage format of a field, specified by the USAGE keyword in the Master File, indicates the format of the field as processed by the server.

The actual format of a field, specified by the ACTUAL keyword in the Master File, indicates the format of a field as returned by the System 2000 data source.

Numeric Fields

All numeric fields are transferred from the System 2000 data source to the server in packed (P) format. The server's actual format for a numeric field is as follows:

- If the field has the System 2000 format INTEGER NUMBER 9(*n*), where *n* is a number, the ACTUAL format is calculated as follows:

  P*n*+1/2

  If the result is not a whole number, take the next whole number. For example, the field AMOUNT has the System 2000 format INTEGER NUMBER 9(4). 4+1 is 5, 5 divided by 2 is 2 1/2. 2 1/2 is not a whole number, so we take the next whole number, which is 3. So AMOUNT has an ACTUAL format of P3.

- If a field has the System 2000 format DECIMAL NUMBER 9(*n*).9(*m*), where *n* and *m* are numbers, the ACTUAL format is calculated as follows:

  P*n* + *m* + 1/2

  If the result is not a whole number, take the next whole number. For example, the field PRICE has the System 2000 format DECIMAL NUMBER 9(5).9(2). 5+2+1 is 8, 8 divided by 2 is 4. So PRICE has an ACTUAL format of P4.

Alphanumeric Fields

Fields defined in the System 2000 data source as CHAR or TEXT types must be defined as character fields. The maximum length for System 2000 character fields is 250 bytes.

Character fields in the server Master File may have different field lengths from those specified in the System 2000 data source definition. If the value stored in the System 2000 data source is:

- Longer than the one in the server Master File, it will be truncated to the defined length.
- Shorter than the one in the server Master File, it will be padded with blanks.

In defining the field length, consider the System 2000 overflow feature.

Date Fields

Date fields must have ACTUAL formats of A6 or A8. For date fields with ACTUAL formats of:

- A6, the allowable USAGE formats are 16MDY, 16YMD, 16DMY, I6MTDY, A6MDY, A6YMD, A6DMY and A6MTDY.
- A8, the allowable USAGE formats are I8DMYY and A8DMYY.

Sample Data Descriptions for System 2000

Example:

LIBRARY and PUBLISHERS Data Sources

LIB and PUB Master and Access Files

This topic includes two sample data sources—LIBRARY and PUBLISHERS—that are supplied when System 2000 is installed, and the corresponding Master Files and Access Files on the server.

Example: LIBRARY and PUBLISHERS Data Sources

The structures of the two System 2000 data sources are as follows:

```
SYSTEM RELEASE NUMBER 10.1
DATA BASE NAME IS      LIBRARY
DEFINITION NUMBER            2
DATA BASE CYCLE NUMBER      50
     1* TITLE _CHAR X(20) WITH FEW FUTURE OCCURRENCES)
     2* PUBLICATION TYPE (CHAR X(10))
     3* VOLUME (CHAR X(7))
     4* CALL NUMBER (DECIMAL NUMBER 999.9(5) WITH FEW FUTURE
        OCCURRENCES)
     5* AUTHOR TAG (CHAR X(7)) WITH FEW FUTURE OCCURRENCES)
     6* DOCUMENT CODE (TEXT X(10))
```

```
   100* SUBJECT (RECORD)
     101* CATEGORY (CHAR X(7) IN 100)
     200* SUBCATEGORIES (RECORD IN 100)
       201* SUBCATEGORY (CHAR X(7)) IN 200)
   300* AUTHORS (RECORD)
     301* SURNAME (CHAR X(10)) IN 300 WITH FEW FUTURE OCCURRENCES)
     303* PSEUDONYM (CHAR X(20) IN 300)
     304* YEAR OF BIRTH (NON-KEY INTEGER NUMBER 9999 IN 300)
     305* YEAR OF DEATH (NON-KEY INTEGER NUMBER 9999 IN 300)
   400* COPIES (RECORD)
     401* COpy NUMBER (INTEGER NUMBER 999 IN 400)
     402* PUBLISHER (CHAR X(14) IN 400)
     403* COPYRIGHT (INTEGER NUMBER 9999 IN 400)
     404* PUBLICATION DATE (INTEGER NUMBER 9999 IN 400 WITH FEW FUTURE
          OCCURRENCES)
     405* REPLACEMENT COST (NON-KEY MONEY $999.99 IN 400)
     406* PAGES (NON-KEY INTEGER NUMBER 9(5) IN 400)
     407* FINE PER DAY (NON-KEY MONEY $99.99 IN 400)
     408* ACTIVITY STATUS (CHAR X(7)) IN 400)
     500* LENDING RECORD (RECORD IN 400)
       501* BORROWER (CHAR X(20) IN 500 WITH MANY FUTURE OCCURRENCES)
       502* CARD NUMBER (INTEGER NUMBER 9(6) IN 500 WITH SOME FUTURE
            OCCURRENCES)
       503* DATE BORROWED (DATE IN 500 WITH MANY FUTURE OCCURRENCES)
       504* DATE DUE (DATE IN 500 WITH MANY FUTURE OCCURRENCES)
       505* DATE RETURNED (DATE IN 500 WITH MANY FUTURE OCCURRENCES)
       506* OUTSTANDING FINE (MONEY $999.99 IN 500 WITH SOME FUTURE
            OCCURRENCES)
     600* BINDING RECORD (RECORD IN 400)
       601* BINDERY (CHAR X(14) IN 600)
       602* BINDING (NON-KEY CHAR X(7) IN 600)
       603* DATE SENT (NON-KEY DATE IN 600)
       604* DATE RECEIVED (NON-KEY DATE IN 600)
       605* BINDING COST (NON-KEY MONEY $99.99 IN 600)
---
SYSTEM RELEASE NUMBER 10.1
DATA BASE NAME IS      PUBLISHERS
DEFINITION NUMBER             1
DATA BASE CYCLE NUMBER       23
     1* PUBLISHER (CHAR X(14))
    10* ADDRESS (RECORD)
      11* LINE (NON-KEY TEXT X(12) IN 10)
    20* TITLES (RECORD)
      21* TITLE (CHAR X(20) IN 20)
      22* COPYRIGHT (INTEGER NUMBER 9999 IN 20)
```

Example: LIB and PUB Master and Access Files

This topic shows the corresponding Master Files and Access Files.

- The files LIB and PUB Master Files describe the server structures that map the System 2000 structures. Note that the fields for ROOT segment in the LIB file have been specified in an arbitrary order.
- The file LIB1 Master File describes a connected subtree. As the ROOT segment is not present, any attempt to perform an inversion would produce errors at the System 2000 level.
- The file LIB2 Master File describes an alternate view of the System 2000 data source. This view may be inverted again to produce the original System 2000 data source structure.

LIB Master File

```
FILENAME=S1,SUFFIX=S2K
SEGNAME=ROOT
 FIELD=AUTHOR TAG,AUTHOR TAG,A7,A7,$
 FIELD=DOC,C6,A10,A10,$
 FIELD=TITLE,C1,A40,A40,$
 FIELD=TYPE,C0002,A10,A10,$
 FIELD=VOLUME,C3,A7,A7,$
 FIELD=NUMBER,C4,ACTUAL=P5,USAGE=P9.5,$
SEGNAME=SUBJECT,PARENT=ROOT
 FIELD=CATEGORY,C101,A7,A7,$
SEGNAME=C200,PARENT=SUBJECT
 FIELD=SUB CATEGORY,C201,A7,A7,$
SEGNAME=AUTHORS,PARENT=ROOT
 FIELD=SURNAME,C301,A10,A10,$
 FIELD=FIRST-NAME,C302,A10,A10,$
 FIELD=PSEUDONYM,C303,A20,A20,$
 FIELD=YBIRTH,C304,ACTUAL=P3.USAGE=P4,$
 FIELD=YDEATH,C305,ACTUAL=P3,USAGE=P4,$
SEGNAME=C400,PARENT=ROOT
 FIELD=COPY-NUMBER,C401,ACTUAL=P2,USAGE=P4,$
 FIELD=PUBLISHER,C402,A14,A14,$
 FIELD=COPYRIGHT,C403,ACTUAL=P3,USAGE=P4,$
 FIELD=PUB-DATE,C404,ACTUAL=P3,USAGE=P4,$
 FIELD=COST,C405,ACTUAL=P3,USAGE=P6.2,$
 FIELD=PAGES,C406,ACTUAL=P3,USAGE=P5,$
 FIELD=FINE,C407,ACTUAL=P3,USAGE=P5.2,$
 FIELD=STATUS,C408,ACTUAL=A7,USAGE=A7,$
SEGNAME=C500,PARENT=C400
 FIELD=BORROWER,C501,ACTUAL=A20,USAGE=A20,$
 FIELD=CARON,C502,ACTUAL=P4,USAGE=P6,$
 FIELD=DATEB,C503,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=DATED,C504,ACTUAL=A8,USAGE=I8DMYY,$
 FIELD=DATER,C505,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=FINE,C506,ACTUAL=P3,USAGE=P6.2,$
SEGNAME=C600,PARENT=C400
 FIELD=BINDERY,C601,ACTUAL=A14,USAGE=A14,$
 FIELD=BINDING,C602,ACTUAL=A7,USAGE=A7,$
 FIELD=DTSENT,C603,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=DTREC,C604,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=COST,C605,ACTUAL=P3,USAGE=P5.2,$
```

LIB Access File

```
DBNAME=LIBRARY.PASSWORD=LIB,
TRACE=NO,
ACCESS=XXX,$
>
```

LIB1 Master File

This file describes a connected subtree.

```
FILENAME=S1,SUFFIX=S2K
SEGNAME=CO400
 FIELD=COPY-NUMBER,C401,ACTUAL=P2,USAGE=P4,$
 FIELD=PUBLISHER,C402,A14,A14,$
 FIELD=COPYRIGHT,C403,ACTUAL=P3,USAGE=P4,$
 FIELD=PUB-DATE,C404,ACTUAL=P3,USAGE=P4,$
 FIELD=COST,C405,ACTUAL=P3,USAGE=P6.2,$
 FIELD=PAGES,C406,ACTUAL=P3,USAGE=P5,$
 FIELD=FINE,C407,ACTUAL=P3,USAGE=P5.2,$
 FIELD=STATUS,C408,ACTUAL=A7,USAGE=A7,$
SEGNAME=C500,PARENT=CO400
 FIELD=BORROWER,C501,ACTUAL=A20,USAGE=A20,$
 FIELD=CARDN,C502,ACTUAL=P4,USAGE=P6,$
 FIELD=DATEB,C503,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=DATED,C504,ACTUAL=A8,USAGE=I8DMYY,$
 FIELD=DATER,C505,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=FINE,C506,ACTUAL=P3,USAGE=P6.2,$
SEGNAME=C600,PARENT=CO400
 FIELD=BINDERY,C601,ACTUAL=A14,USAGE=A14,$
 FIELD=BINDING,C602,ACTUAL=_7,USAGE=A7,$
 FIELD=DTSENT,C603,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=DTREC,C604,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=COST,C605,ACTUAL=P3,USAGE=P5.2,$
```

LIB2 Master File

This file provides an alternate view of the System 2000 data source.

```
FILENAME=S1,SUFFIX=S2K
SEGNAME=C400
 FIELD=COPY-NUMBER,C401,ACTUAL=P2,USAGE=P4,$
 FIELD=PUBLISHER,C402,A14,A14,$
 FIELD=COPYRIGHT,C403,ACTUAL=P3,USAGE=P4,$
 FIELD=PUB-DATE,C404,ACTUAL=P3,USAGE=P4,$
 FIELD=COST,C405,ACTUAL=P3,USAGE=P6.2,$
 FIELD=PAGES,C406,ACTUAL=P3,USAGE=P5,$
 FIELD=FINE,C407,ACTUAL=P3,USAGE=P5.2,$
 FIELD=STATUS,C408,ACTUAL=A7,USAGE=A7,$
SEGNAME=C500,PARENT=C400
 FIELD=BORROWER,C501,ACTUAL=A20,USAGE=A20,$
 FIELD=CARDN,C502,ACTUAL=P4,USAGE=P6,$
 FIELD=DATEB,C503,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=DATED,C504,ACTUAL=A8,USAGE=I8DMYY,$
 FIELD=DATER,C505,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=FINE,C506,ACTUAL=P3,USAGE=P6.2,$
SEGNAME=C600,PARENT=C400
 FIELD=BINDERY,C601,ACTUAL=A14,USAGE=A14,$
 FIELD=BINDING,C602,ACTUAL=A7,USAGE=A7,$
 FIELD=DTSENT,C603,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=DTREC,C604,ACTUAL=A6,USAGE=I6MTDY,$
 FIELD=COST,C605,ACTUAL=P3,USAGE=P5.2,$
SEGNAME=ROOT,PARENT=C400
 FIELD=TITLE,C1,A40,A40,$
 FIELD=TYPE,COOO2,A10,A10,$
 FIELD=VOLUME,C3,A7,A7,$
 FIELD=NUMBER,C4,ACTUAL=P5,USAGE=P9.5,$
 FIELD=AUTHOR TAG,AUTHOR TAG,A7,A7,$
 FIELD=DOC,C6,A10,A10,$
SEGNAME=SUBJECT,PARENT=ROOT
 FIELD=CATEGORY,C101,A7,A7,$
SEGNAME=C200,PARENT=SUBJECT
 FIELD=SUB CATEGORY,C201,A7,A7,$
SEGNAME=AUTHORS,PARENT=ROOT
 FIELD=SURNAME,C301,A10,A10,$
 FIELD=FIRST-NAME,C302,A10,A10,$
 FIELD=PSEUDONYM,C303,A20,A20,$
 FIELD=YBIRTH,C304,ACTUAL=P3,USAGE=P4,$
 FIELD=YDEATH,C_O5,ACTUAL=P3,USAGE=P4,$
```

PUB Master File

```
FILENAME=S1,SUFFIX=S2K
SEGNAME=ROOT
 FIELD=PUBLISHER,C1,A40,A40,$
SEGNAME=C10,PARENT=ROOT
 FIELD=LINE,LINE,A40,A40,$
 SEGNAME=C20,PARENT=ROOT
 FIELD=TITLE,TITLE,A40,A40,$
 FIELD=COPYRIGHT,COPYRIGHT,ACTUAL=P3,USAGE=P4,$
```

PUB Access File

```
DBNAME=LIBRARY.PASSWORD=PUB,
TRACE=NO,
ACCESS=XXX,$
>
```

Navigation Strategies

In this section:

The server can access System 2000 data sources using two strategies:

- **Sequential.** The adapter uses this strategy when a request to the server does not include any screening conditions. In addition, you may want to choose this strategy if the screening conditions are not required for optimal retrieval.

- **Selective.** The adapter uses the strategy when a request to the server includes screening conditions. To retrieve all System 2000 entries that satisfy the screening conditions, the adapter issues a LOCATE command and retrieves the records using a sequence of GETD and GET NEXT commands.

  The adapter translates:

  - Values in the form

    ```
    IF field EQ value
    ```

    into the System 2000 format:

    ```
    WHERE <ENTRYNAME> HAS field EQ value
    ```

  - Range tests in the form

    ```
    IF field FROM value1 TO value2
    ```

    into the System 2000 format:

    ```
    WHERE <ENTRYNAME> HAS field SPANS value1*value2
    ```

Note that each access strategy can be used with and without file inversion.

You indicate which strategy you wish to use with the ACCESS attribute in the Access File.

Example: Translating Server Selection Criteria to System 2000

The following selection criteria

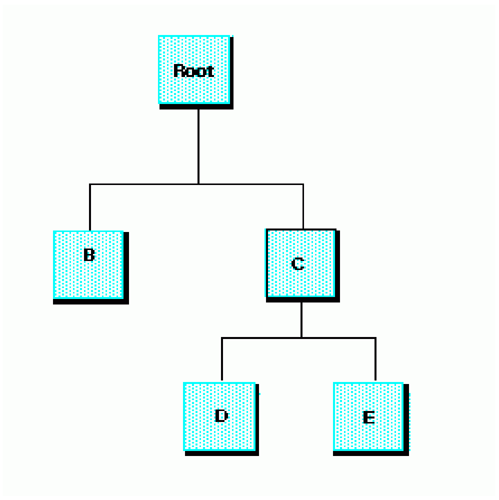
```
IF CAR EQ TRIUMPH
IF MODEL EQ TR7
IF MPG FROM 10 TO 15
```

translate into System 2000 format as:

```
WHERE <ENTRYNAME> HAS CAR EQ TRIUMPH AND
      <ENTRYNAME> HAS MODEL EQ TR7   AND
      <ENTRYNAME> HAS MPG SPANS 10*15
```

Sequential Access Without Inversion

Assume that you are working with a System 2000 data source with the following structure, and a Master File on the server that describes the data source the same way:



The sequence of System 2000 commands is as follows:

```
GET1 ROOT FIRST
GETD B
GETD B(repeated until no more Bs exist)
GETD C
GETD D(repeated until no more Ds exist)
GETD E(repeated until no more Es exist)
GETD C
GETD D(repeated until no more Ds exist)
GETD E(repeated until no more Es exist)
GETD C
```

... and so on until no more Cs exist under the current ROOT.

```
GET1 ROOT NEXT
```

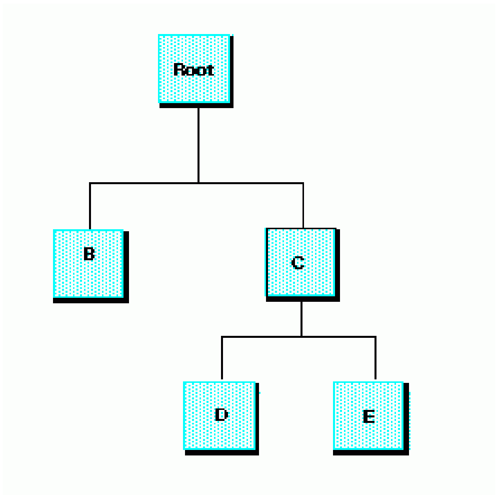
... and so on until no more ROOTs exist.

Sequential Access With Inversion

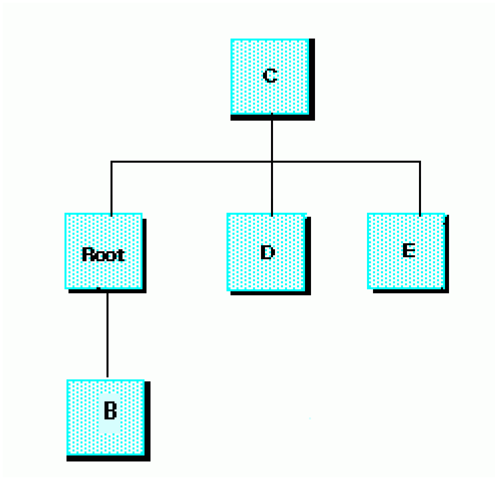
If you access a System 2000 file with an inverted server structure, the adapter uses the command:

- STACK 0 to access the root segment of the inverted structure and segments descended from this segment in the original System 2000 structure.
- STACK 1 to access the segments that lie along the root path and the segments descended from these segments in the original System 2000 structure.

For example, suppose that a System 2000 data source has the following structure



and is inverted on the server into the following structure:



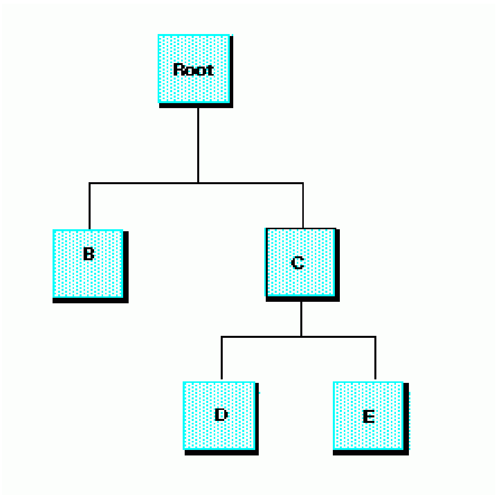
The sequence of System 2000 commands is:

```
GET1(O) C FIRST
COPYS (O) TO 1
GET1(1) ROOT PRESENT
GETD(1) B
GETD(1) B           (repeated until no more Bs exist)
GETD(O) 0           (repeated until no more Ds exist)
GETD(O) E           (repeated until no more Es exist)
GET1(O) C NEXT
COPYS (0) TO 1
GET1(1) ROOT PRESENT
GETD(1) B
GETD(1) B           (repeated until no more Bs exist)
GETO(O) 0           (repeated until no more Ds exist)
GETO(O) E           (repeated until no more Es exist)
COPYS (O) TO 1
GET1(O) C NEXT
... and so on until no more Cs exist.
```

Note that you can invert a file only if the root segment is defined in the Master File. However, you need not refer to fields in the segment in requests to the server.

Selective Access Without Inversion

Assume that you are working with a System 2000 data source that has the following structure, and is described the same way in the Master File on the server:



The sequence of System 2000 commands is as follows:

```
LOCATE (O) ROOT WHERE (SYSTEM 2000 WHERE conditions are equivalent
                    to screening conditions used by the adapter)
GET(O) ROOT NEXT
GETD(O) B
GETD(O) B               (repeated until no more Bs exist)
GETD(O) C
GETD(O) D               (repeated until no more Ds exist)
GETD(O) E               (repeated until no more Es exist)
GETD(O) C
GETD(O) D               (repeated until no more Ds exist)
GETD(O) E               (repeated until no more Es exist)
GETD(O) C
... and so on until no more Cs exist under the current ROOT.
GET (O) ROOT NEXT
... and so on until the end of the LOCATE file.
```

Selective Access With Inversion

There are two ways to invert a System 2000 file structure:

- In the Master File on the server.
- At run time, using the command

  ```
  TABLE FILE filename.fieldname
  ```

  where:

  filename

  Is the name of the file you are accessing.

  fieldname

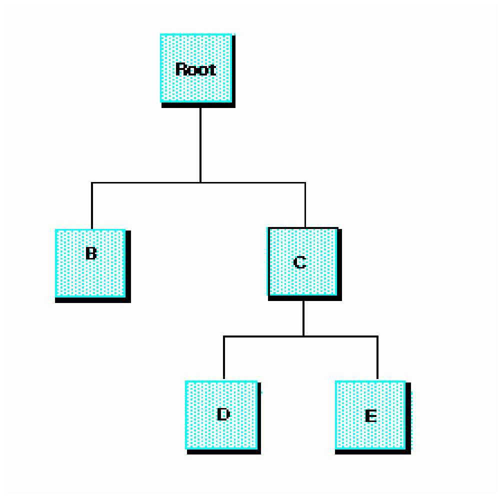
  Is the name of a field in the segment that is to become the root segment of the inverted structure.

The path that links the root of the original structure to the root of the inverted structure is called the root path.

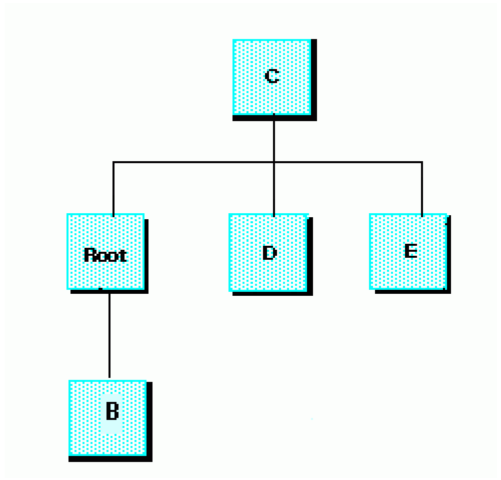
If you access a System 2000 file with an inverted server structure, the adapter uses the command:

- STACK 0 to access the root segment of the inverted structure and segments descended from this segment in the original System 2000 structure.
- STACK 1 to access the segments that lie along the root path and the segments descended from these segments in the original System 2000 structure.

For example, suppose that a System 2000 data source has the following structure



and is inverted on the server into the following structure:



The sequence of System 2000 commands is as follows:

```
LOCATE (O) C WHERE (SYSTEM 2000 WHERE conditions are equivalent
                    to screening conditions used by the adapter)

GET1(O) C NEXT
COPYS (O) TO 1
GETl(l) ROOT PRESENT
GETD(1) B
GETD(1) B           (repeated until no more Bs exist)
GETD(O) D           (repeated until no more Ds exist)
GETD(O) E           (repeated until no more Es exist)
GET(O) C NEXT
COPYS (O) TO 1
GETl(l) ROOT PRESENT
GETD(1) B
GETD(l) B           (repeated until no more Bs exist)
GETD(O) D           (repeated until no more Ds exist)
GETD(O) E           (repeated until no more Es exist)
COPYS (O) TO 1
GET1(O) C NEXT
... and so on until the end of the LOCATE file.
```

Note that you can invert a file only if the root segment is defined in the Master File. However, you need not reference fields in the segment in your requests to the server.

CHAPTER 47

Using the Adapter for Tamino

Topics:

The Adapter for Tamino allows Tamino client applications to access Tamino providers. The adapter converts:

- Internal data manipulation language (DML) requests or SQL requests into Tamino requests.
- Tamino responses into answer sets.

Preparing the Tamino Environment

The Adapter for Tamino minimally requires a Java Virtual Machine (JVM), Version 1.3 or higher, running on the same machine as the adapter.

To prepare the Tamino environment, you must set your CLASSPATH to include the following iWay and Tamino jar files:

		Directory in which files are located
iWay file	jtmnin.jar	`$EDAHOME\etc`
Tamino files	jdom.jar log4j.jar TaminoAPI4J.jar TaminoJCA.jar xercesImpl.jar xmlParserAPIs.jar	`c:\Program Files\Software AG\Tamino\Tamino 4.1.4.1\SDK\TaminoAPI4J\lib`

You must also modify your PATH to include the following directory:

```
C:\Program Files\Software AG\Tamino\Tamino 4.1.4.1\SDK\TaminoAPI4C\lib
```

Connecting to a Remote Tamino Server

If you are connecting to a remote Tamino Server, you must modify the *xtsurl.cfg* file, which is located in the following directory:

```
C:\Documents and Settings\All Users\Application Data\Software AG
```

Syntax: How to Modify Configuration for a Remote Tamino Server

Locate the *xtsurl.cfg* and include the following lines of code in the xfsurl.cfg file:

```
XTSlisten.Tamino:database_name[0]=tcpip://host:port
XTSaccess.Tamino:database_name[0]=tcpip://host:port
```

where:

database_name

Is the name of the database that exists on the Tamino Server.

host

Is the machine name or IP address where the Tamino server is installed.

port

Is the port number on which the Tamino Server is listening.

Configuring the Adapter for Tamino

In this section:

Declare Connection Attributes Manually

Overriding the Default Connection

Configuring the adapter consists of specifying connection and authentication information for each of the Web Application Servers you want to access.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

Example:

Declaring Connection Attributes

In order to locate the Web application server hosting the target Web service, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Web application server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Web application server takes place when the first request that references the connection is issued.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *XML-based* group folder, then expand the *Tamino* adapter folder and click a connection. The Add Connection for Tamino pane opens.
3. Enter the following parameters:

Parameter	Description
Connection Name	Logical name used to identify this particular set of Web application attributes.
Tamino database	Name of the Tamino database.
Tamino Server URL	URL of the Tamino server.
Security	There are two methods by which a user can be authenticated when connecting to a Tamino database server: **Explicit.** The user ID and password are explicitly specified for each connection and passed to Tamino, at connection time, for authentication. **Password Passthru.** The user ID and password received from the client application are passed to Tamino, at connection time, for authentication. This option requires that the server be started with security off.
User	Authorization ID by which the user is known to Tamino.
Password	Password associated with the user ID. The password is stored in encrypted form.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click the *Configure* button.

   Tip: Once the adapter is configured, the connection name appears in the navigation pane under *Configured*. If you left-click either the adapter name or the connection name, an appropriate set of options becomes available.

Syntax: How to Declare Connection Attributes Manually

The user ID and password are explicitly specified for each connection and passed to Tamino, at connection time, for authentication.

```
ENGINE TMNIN SET CONNECTION_ATTRIBUTES connection/uid,pwd;:'database
 database_url'
```

where:

`connection`

Is a logical name used to identify this particular set of Web application attributes.

`database`

Is the Tamino database.

`database_url`

Is the URL of the Tamino database.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allows the application to access the Web application named SAMPLEAPP with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE TMNIN SET CONNECTION_ATTRIBUTES DEMO
 /edabxv,C341A037BABA5D81;:'baruch http://localhost/tamino/baruch'
```

Overriding the Default Connection

How to:

Change the Default Connection

Example:

Selecting the Default Connection

Once connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [TMNIN] SET DEFAULT_CONNECTION [connection]
```

where:

`TMNIN`

Indicates the Adapter for Tamino. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Tamino database server named SAMPLENAME as the default Tamino database server:

```
ENGINE TMNIN SET DEFAULT_CONNECTION SAMPLENAME
```

Managing Tamino Metadata

In this section:

When the server accesses a Tamino provider, it needs to know how to interpret the WSDL description of the Web service operations that it finds. For each operation the server will access, you create a synonym that describes the operation and the server mapping of the Tamino data types. You can describe related operations using a single synonym.

Creating Synonyms

A synonym defines a unique logical name (also known as an alias) for each Tamino operation. Synonyms:

- Insulate client applications from changes to a request's location and identity. You can move or rename a request without modifying the client applications that use it; you need make only one change, redefining the request's synonym on the server.
- Provide support for the server's extended metadata features, such as virtual fields and security mechanisms.

Creating a synonym generates a Master File and an Access File. These are metadata that describe the Tamino request to the server.

When you create a synonym using the:

- **Web Console,** where you can use the synonym to describe one operation within a Web service.
- **CREATE SYNONYM command,** where you can use the synonym to describe multiple operations within the same Web service.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter. See *Configuring the Adapter for Tamino* on page 47-3 for more information.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The Collection Name Selection pane (Step 1 of 3) opens.

4. Click the Filter Collection Names check box to filter the collections for which you wish to create synonyms. To create synonyms for all collections leave this box blank.

5. Click *Select Collections*. All collections that meet the specified criteria are displayed in the Select Collection Name pane (Step 2 of 3).

6. Click the drop-down arrow to choose a collection, then click *Next*.

   The Select Schema Name pane (Step 3 of 3) opens.

7. Click the drop-down list to choose the schema name (for example, HospitalSchema) that will be assigned to the synonym.

8. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

   When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

9. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

10. Enter or select the following additional parameters:

Parameter	**Description**
Select Application	Select an application directory. baseapp is the default value.
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

11. Complete your selection:

To select all schemas in the list, select the check box to the left of the *Default Synonym Name* column heading.

To select specific schemas, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

Synonyms are created and added under the specified application directory.

A status window displays the message:

```
All Synonyms Created Successfully
```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '| ', '&', ' (', ') ', ' <', '> ', '' ', ' =', ''''

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM [app/]synonym FOR collection/schema DBMS TMNIN [AT connection]
[CHECKNAMES] [UNIQUENAMES]
END
```

where:

app

Is the application namespace in which you want to create the synonym.

If your server is APP-enabled, you can specify this; otherwise, you must omit it.

synonym

Is an alias for a request (maximum 64 characters).

`collection`

Is the largest unit of information within a database. It serves as a container for related information.

Note: The adapter does not support queries from schemas that contain two different DOCTYPES. Attempting to report from a synonym that contains different DOCTYPES will crash the agent. If you are testing from the Web Console, a *Session Lost* message is displayed.

`schema`

A Tamino schema complies with a subset of the W3C XML schema standard with Tamino-specific information defined in annotations. Within a schema, we distinguish between a physical schema and a logical schema.

`TMNIN`

Indicates the Adapter for Tamino.

`connection`

Is the name of the connection to the Web application server, as previously declared in the adapter's configuration. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

If multiple connections have been declared for this adapter, and you do not want to accept the default connection, you must specify the AT *connection* option here. Otherwise, if there is only one connection, or if you wish to accept the default connection, you can omit this option. When specified, the CONNECTION attribute is added to the Access File.

Note: When the connection's attributes were declared, the declaration must have specified the URL that describes the operation. For more information about declaring connection attributes, see *Declare Connection Attributes Manually* on page 47-5.

`CHECKNAMES`

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

`UNIQUENAMES`

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Creating a Synonym

The following example shows how to create a synonym for hospitalschema in the collection. The synonym will be saved under the baseapp application directory. The corresponding Master File and Access file follow.

```
CREATE SYNONYM baseapp/hospitalschema FOR hospital/patient DBMS TMNIN
 AT DEMO
END
```

Generated Master File for Hostpitalschema

```
FILENAME=HOSPITALSCHEMA, SUFFIX=TMNIN   , $
  SEGMENT=PATIENT, SEGTYPE=S0, $
    FIELDNAME=PATIENT, ALIAS=patient, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=NAME, ALIAS=name, USAGE=A1, ACTUAL=A1, MISSING=ON,
      REFERENCE=PATIENT, PROPERTY=ELEMENT,  $
    FIELDNAME=SURNAME, ALIAS=surname, USAGE=A20V, ACTUAL=A20V,
      REFERENCE=PATIENT.NAME, PROPERTY=ELEMENT,  $
    FIELDNAME=FIRSTNAME, ALIAS=firstname, USAGE=A20V, ACTUAL=A20V,
      REFERENCE=PATIENT.NAME, PROPERTY=ELEMENT,  $
    FIELDNAME=TITLE, ALIAS=title, USAGE=A20V, ACTUAL=A20V, MISSING=ON,
      REFERENCE=PATIENT.NAME, PROPERTY=ELEMENT,  $
    FIELDNAME=SEX, ALIAS=sex, USAGE=A20V, ACTUAL=A20V,
      REFERENCE=PATIENT, PROPERTY=ELEMENT,  $
    FIELDNAME=BORN, ALIAS=born, USAGE=P33, ACTUAL=A33, MISSING=ON,
      REFERENCE=PATIENT, PROPERTY=ELEMENT,  $
    FIELDNAME=ADDRESS, ALIAS=address, USAGE=A1, ACTUAL=A1, MISSING=ON,
      REFERENCE=PATIENT, PROPERTY=ELEMENT,  $
    FIELDNAME=STREET, ALIAS=street, USAGE=A20V, ACTUAL=A20V, MISSING=ON,
      REFERENCE=PATIENT.ADDRESS, PROPERTY=ELEMENT,  $
        .
        .
        .
```

Generated Access File for Hospitalschema

```
SEGNAME=PATIENT, CONNECTION=DEMO, COLLECTION=Hospital, SCHEMA=patient,$
```

Example: Reporting from the Hospitalschema Synonym

You can issue an SQL or TABLE request.

Sample SQL Request

```
>>sql
>select surname, firstname, title, sex, born
>from hospitalschema
>end
```

The output is:

```
 SURNAME            FIRSTNAME         TITLE      SEX        BORN
 -------            ---------         -----      ---        ----
 Bloggs             Fred              .          Male       .

 NUMBER OF RECORDS IN TABLE=        1  LINES=      1
```

Sample SQL Request

```
>>table file hospitalschema
>print surname  firstname  title  sex born
>end
```

The output is:

```
 NUMBER OF RECORDS IN TABLE=        1  LINES=      1

 PAGE     1

 SURNAME           FIRSTNAME         TITLE       SEX         BORN

 -------           ---------         -----       ---         ----
 Bloggs            Fred              .           Male        .
```

Reference: Master File Attributes

Attribute	Description
`PROPERTY`	Indicates whether the field corresponds to an XML attribute or an XML element. PROPERTY is used only in Master File segments that map directly to the response document.
`REFERENCE`	Identifies this field's parent element in the XML hierarchy, as defined by the Tamino document's XML schema. The value's format is *`segmentname.fieldname`* where: *`segmentname`* Is the name of the Master File segment in which the field resides. *`fieldname`* Is the name of the field that corresponds to the parent element.

Reference: Access File Attributes

Attribute	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local Tamino database server. Absence of the CONNECTION attribute indicates access to the default database server.
`COLLECTION`	In Tamino, is the largest unit of information within a database. It serves as a container for related information. **Note:** The adapter does not support queries from schemas that contain two different DOCTYPES. Attempting to report from a synonym that contains different DOCTYPES will crash the agent. If you are testing from the Web Console, a *Session Lost* message is displayed.

Attribute	Description
SCEHMA	A Tamino schema complies with a subset of the W3C XML schema standard with Tamino-specific information defined in annotations. Within a schema, we distinguish between a physical schema and a logical schema.

Data Type Support

The following table lists how the server maps XSD data types in a Master File. You can change some of these mapping defaults, as described in *Changing the Length of Character Strings* on page 47-18 and *Changing the Precision and Scale of Numeric Fields* on page 47-19.

XSD Data Type	USAGE Attribute	ACTUAL Attribute
decimal	P33.13	A33
integer	P33	A33
nonPositiveInteger	33	A33
negativeInteger	P33	A33
long	P33	A33
int	P33	A33
short	P33	A33
byte	I4	A4
nonNegativeInteger	P32	A32
unsignedLong	P20	A20
unsignedInt	P10	A10
unsignedShort	I5	A5
unsignedByte	I4	A4
positiveInteger	P32	A32
double	D24.16E	A24
float	F15.8E	A15
duration	A10	A10

XSD Data Type	USAGE Attribute	ACTUAL Attribute
dateTime	HYYMDm	A27
time	HHISsm	A15
date	YYMD	A10
gYearMonth	HYYM	A8
gYear	HYY	A5
gMonthDay	HMD	A6
gDay	HD	A3
gMonth	HM	A4
string	A10	A10
normalizedString	A10	A10
token	A10	A10
Name	A10	A10
NMTOKEN	A10	A10
NCName	A10	A10
ID	A10	A10
hexBinary	A10	A10
language	A10	A10
anyURI	A10	A10
QName	A10	A10

Changing the Length of Character Strings

How to:

Switch Between Variable and Fixed-length Strings

Change String Length

By default, when the Adapter for Tamino creates a synonym, it maps all fields defined as string in the XML schema to 10-byte fixed-length alphanumeric in the Master File. You can change the length (the variability and the number of bytes) using the SET VARCHAR and SET LENGTH commands.

Syntax: How to Switch Between Variable and Fixed-length Strings

By default, when the Adapter for Tamino creates a synonym, it maps all fields defined as string in the XML schema to fixed-length alphanumeric in the Master File. You can change this default to variable length using the SET VARCHAR command

```
ENGINE [TMNIN] SET VARCHAR {ON|OFF}
```

where:

`TMNIN`

Indicates the Adapter for Tamino. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps all fields defined as string in the XML schema to variable-length alphanumeric (A*n*V) in the Master File's USAGE and ACTUAL attributes.

`OFF`

Maps all fields defined as string in the XML schema to fixed-length alphanumeric (A) in the Master File's USAGE and ACTUAL attributes. OFF is the default value.

Syntax: How to Change String Length

By default, when the Adapter for Tamino creates a synonym, it maps all fields defined as string in the XML schema to 10-byte alphanumeric in the Master File. You can change this default length using the SET FIELDLENGTH command

```
ENGINE [TMNIN] XML SET FIELDLENGTH length
```

where:

`TMNIN`

Indicates the Adapter for Tamino. You can omit this value if you previously issued the SET SQLENGINE command.

`length`

Is the length, in bytes, assigned to the ACTUAL and USAGE attributes of all fields defined in the XML schema as string. The maximum length is 3000.

Changing the Precision and Scale of Numeric Fields

How to:

Change the Precision and Scale of Numeric Fields

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric fields returned by a request by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL attributes of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax: How to Change the Precision and Scale of Numeric Fields

```
ENGINE [TMNIN] SET CONVERSION RESET
ENGINE [TMNIN] SET CONVERSION format RESET
ENGINE [TMNIN] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [TMNIN] SET CONVERSION format [PRECISION MAX]
```

where:

`TMNIN`

Indicates the Adapter for Tamino. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` indicates that the command applies only to INTEGER columns.

`DECIMAL` indicates that the command applies only to DECIMAL columns.

`REAL` indicates that the command applies only to single precision floating point columns.

`FLOAT` indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL, REAL, and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set mm to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	**MAX Precision**
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When you issue the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when you issue a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depend on whether a CONVERSION command is in effect.

- If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.
- If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER fields to 7:

```
ENGINE TMNIN SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE TMNIN SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER fields to the default:

```
ENGINE TMNIN SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE TMNIN SET CONVERSION RESET
```

CHAPTER 48

Using the Adapter for Teradata

Topics:

The Adapter for Teradata allows applications to access Teradata data sources. The adapter converts application requests into native Teradata statements and returns optimized answer sets to the requesting application.

Preparing the Teradata Environment

In order to use the Adapter for Teradata, NCR Teradata ODBC or CLI clients must be installed and configured and the path to the Teradata ODBC or CLI libraries directory must be added to SYSTEM LIBRARY PATH. See the NCR Teradata documentation for more information about requirements on ODBC or CLI installation.

If you are using the OS/390 and z/OS server, no preconfiguration steps are required. Use the Web Console to configure the Teradata Adapter.

Procedure: How to Set Up the Environment on Microsoft Windows Using ODBC or CLI

On Microsoft Windows, the Teradata environment is set up during the installation of the product.

Procedure: How to Set Up the Environment on UNIX

You can access Teradata using the optional UNIX environment variable, ODBCINI. The ODBCINI variable points to the absolute path of the .odbc.ini file.

For example:

```
ODBCINI=/usr/odbc/.odbc.ini
export ODBCINI
```

Note: You must not use ODBCINI, if the $HOME directory contains the .odbc.ini file.

Configuring the Adapter for Teradata

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Connection Attributes From the Web Console

Declare Connection Attributes Manually

Example:

Declaring Connection Attributes

In order to connect to an Teradata database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Teradata database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Teradata Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	**Description**
Datasource	Valid Teradata data source name (DSN). There is no default DSN; a value must be entered. **For the ODBC interface**, this DSN name should match the entry in the $HOME/.odbc.ini. file. **For the CLI interface**, this field is labeled SERVER and requires a valid Teradata TDP value. On Windows and UNIX this is the value of i_dbcpath in the clispb.dat file. Refer to the Teradata Utilities documentation for details.
User	Authorization by which the user is known to Teradata.
Password	Password associated with the authorization ID. The password is stored in encrypted form.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

Note: The release numbers on the Web Console for DBMS Add and DBMS Change panes refer to the Teradata ODBC driver, not the Teradata DBMS release.

4. Click *Configure*.

Syntax: How to Declare Connection Attributes Manually

```
ENGINE [SQLDBC] SET CONNECTION_ATTRIBUTES [connection]/userid,password
```

where:

`SQLDBC`

Indicates the Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is a logical name (or a data source name) used to identify this particular set of attributes.

For the Adapter for Teradata ODBC, this is the connect descriptor to be used for the Data Source Name (DSN) node in the .odbc.ini.

For the Adapter for Teradata CLI, this is the Teradata Director Program number (TDP*n*), where *n* is the number. This is the value of i_dbcpath in the clispb.dat file.

`userid`

Is the primary authorization ID by which you are known to Teradata.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Teradata database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLDBC SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Overriding the Default Connection

How to:

Change the Default Connection

Example:

Selecting the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLDBC] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLDBC`

Indicates the Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the Teradata database server named SAMPLENAME as the default Teradata database server:

```
ENGINE SQLDBC SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLDBC] SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLDBC`

Indicates the Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Teradata Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Teradata data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Teradata table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.
10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value. See *Controlling the Mapping of Large Character Data Types* for more information.

11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.

12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.
Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)

Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLDBC [AT connection]
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLDBC

Indicates the Data Adapter for Teradata.

`AT connection`

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLDBC AT DSN_A NOCOLS
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLDBC ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

Keyword	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Name of the table or view. This value can include a location or owner name as follows: `TABLENAME=[location.][owner.]tablename` **Note:** Location is valid only with DB2 CAF and specifies the subsystem location name.
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=connection`
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following table lists how the server maps Teradata data types. Note that you can:

- Control the mapping of large character data types.
- Change the mapping of variable-length data types.
- Change the precision and scale of numeric columns.

Be sure to review those options carefully as they affect the mapping of data types.

Teradata Data Type	Data Type		Remarks
	USAGE	**ACTUAL**	
CHAR (*n*)	A*n*	A*n*	*n* is an integer between 1 and 3200
VARCHAR (*n*)	A*n*	A*n*	
LONG VARCHAR	A*n*	A*n*	Equivalent to VARCHAR (64000) Server supports up to 32767 bytes. All data exceeding 32K will be truncated.

Teradata Data Type	Data Type		Remarks
	USAGE	**ACTUAL**	
SMALLINT	I6	I4	2 byte signed binary integer Range: 2^{15} to 2^{15} - 1
INTEGER	I11	I4	4 byte binary integer Range: -2.147G to +2.147G
BYTEINT	I6	I4	1 byte signed binary integer Range: -128 to +127
DECIMAL	P20.2	P8 P10	*n* is an integer between 1 and 18 *m* is an integer between 0 and n Actual=P10 for n18. Also referred to as NUMERIC.
FLOAT	D20.2	D8	8 bytes range: 2 * 10^{307} to 2 * 10^{308} Same as REAL or DOUBLE PRECISION.
DATE	YYMD	A8	Date/Time formats are comprised of several components. Not a true data value stored internally.
TIME	HHISsm	A12	Date/Time formats are comprised of several components. Not a true data value stored internally. The operation with different Teradata TIME formats depends on DateTimeFormat setting in the datasource part of the $HOME/.odbc.ini file. The adapter requires DateTimeFormat=IAI. Neither the ODBC nor the CLI interface supports integer format of TIME (I). Only the ANSI format of TIME (AT) is supported.

Teradata Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
TIMESTAMP	HYYMdm	A20	Date/Time formats are comprised of several components. Not a true data value stored internally.
INTERVAL	A*n*	A*n*	INTERVAL identifies a period of time in different ranges (YEAR, MONTH, DAY, HOUR, MIN, SEC). The Teradata external (client) representation of INTERVAL is always CHAR (*n*) where *n* = *p* + *x*. Precision *p* from 1 to 4 and *x* from 1 to 11 depends on range.
GRAPHIC			Not supported in current release of the server.
VARGRAPHIC			Not supported in current release of the server.
LONG GRAPHIC			Not supported in current release of the server.
VARBYTE			Not supported in current release of the server.

Controlling the Mapping of Large Character Data Types

The SET parameter CONVERSION LONGCHAR controls the mapping of supported Teradata data types listed below. By default, the server maps these data types as alphanumeric (A). The server data type A supports a maximum of 4096 characters for TABLE/MODIFY and 32768 characters for API applications.

The following table lists data type mappings based on the value of LONGCHAR:

Teradata Data Type	**Remarks**	**LONGCHAR ALPHA or BLOB**		**LONGCHAR TEXT**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
CHAR	*n* is an integer between 1 and 4000	A*n*	A*n*	TX50	TX
VARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*	A*n*	TX50	TX
LONG VARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*	A*n*	TX50	TX

Syntax: How to Control the Mapping of Large Character Data Types

```
ENGINE [SQLDBC] SET CONVERSION LONGCHAR {ALPHA|TEXT|BLOB}
```

where:

`SQLDBC`

Indicates the Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

`ALPHA`

Maps the Teradata data types CHAR, VARCHAR, and LONG VARCHAR as alphanumeric (A). ALPHA is the default value when the synonym is created manually.

`TEXT`

Maps the Teradata data types data types CHAR, VARCHAR, and LONG VARCHAR as text (TX). Use this value for WebFOCUS applications. TEXT is the default value when the synonym is created from the Web Console.

`BLOB`

For UNIX, Windows, OpenVMS, and OS/400, is identical to ALPHA. That is, it maps the Teradata data types CHAR, VARCHAR, and LONG VARCHAR as alphanumeric (A).

For OS/390 and z/OS, maps the Teradata data types CHAR, VARCHAR, and LONG VARCHAR as binary large object (BLOB).

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Teradata data type VARCHAR. By default, the server maps this data type as alphanumeric (A).

The following table lists data type mappings based on the value of VARCHAR:

Teradata Data Type	**Remarks**	**VARCHAR ON**		**VARCHAR OFF**	
		USAGE	**ACTUAL**	**USAGE**	**ACTUAL**
VARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*V	A*n*V	A*n*	A*n*

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE [SQLDBC] SET VARCHAR {ON|OFF}
```

where:

`SQLDBC`

Indicates the Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Teradata data type VARCHAR as variable-length alphanumeric (A*n*V).

`OFF`

Maps the Teradata data type VARCHAR as alphanumeric (A). OFF is the default value.

Changing the Precision and Scale of Numeric Columns

How to:

Override the Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override the Default Precision and Scale

```
ENGINE [SQLDBC] SET CONVERSION RESET
ENGINE [SQLDBC] SET CONVERSION format RESET
ENGINE [SQLDBC] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLDBC] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLDBC`

Indicates the Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLDBC SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLDBC SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLDBC SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLDBC SET CONVERSION RESET
```

Customizing the Teradata Environment

In this section:

The Adapter for Teradata provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Controlling the Column Heading in a Request

You can use the SET COLNAME command to control the column heading in a request.

Syntax: How to Control Column Headings

```
ENGINE [SQLDBC] SET COLNAME {NAME|TITLE}
```

where:

`SQLDBC`

Indicates the Adapter for Teradata. You can omit this parameter value if you previously issued the SET SQLENGINE command.

`NAME`

If you specify the NAME option, the Teradata column name will be used as the column heading.

`TITLE`

If you specify the TITLE option, the name you provide is used as the column heading. TITLE is the default value.

Activating NONBLOCK Mode

The Adapter for Teradata has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE [SQLDBC] SET NONBLOCK {0|n}
```

where:

`SQLDBC`

Indicates the Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

`n`

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLDBC] SET PASSRECS {ON|OFF}
```

where:

`SQLDBC`

Indicates the Adapter for Teradata. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Optimization Settings

In this section:

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

How to:

Optimize Requests

Example:

SQL Requests Passed to the RDBMS With Optimization OFF

SQL Requests Passed to the RDBMS With Optimization ON

Reference:

SQL Generation in Optimization Examples

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLDBC] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLDBC`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

OFF instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

ON instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

FOCUS passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

SQL passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLDBC set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLDBC set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLDBC] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLDBC`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLDBC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLDBC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLDBC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Improving Efficiency With Aggregate Awareness

Aggregate awareness substantially improves the efficiency of queries.

Note: For Teradata, aggregate awareness is handled internally. No adapter settings are required. However, the COLLECT STATISTICS command must be issued in Teradata.

Calling a Teradata Macro Using SQL Passthru

SQL Passthru is supported for a Teradata Macro only through the ODBC interface. These macros need to be developed within Teradata using the CREATE or REPLACE MACRO command.

Example: Calling a Macro

The supported syntax to call a Macro is shown below.

```
ENGINE SQLDBC
EX SAMPLE PARM1,PARM2,PARM3...;
TABLE FILE SQLOUT
END
```

CHAPTER 49

Using the Adapter for TOTAL

Topics:

The Adapter for TOTAL allows applications to access TOTAL data sources. The adapter converts application requests into native TOTAL statements and returns optimized answer sets to the requesting application.

Preparing the TOTAL Environment

Before you start the server, you must preconfigure the adapter environment.

The Adapter for TOTAL is currently available under the OS/390 and z/OS operating systems.

The server and the adapter for TOTAL run together as a simple application program, issuing calls to TOTAL using the entry point DATBAS. (The routine DATBAS, which is called to communicate with the TOTAL database management system, is dynamically loaded by the adapter.) Normal environment considerations for TOTAL (such as file allocations) apply, along with additional requirements and allocations for the server.

Syntax: How to Prepare the TOTAL Environment

Include the following code in the iWay ISTART JCL to allocate the data sets used by the server and the adapter and to implement the adapter.

```
//EDASERVE  EXEC PGM=TSCOM300
//STEPLIB    DD  DISP=SHR,DSN=prefix.HOME.LOAD
//           DD  DISP=SHR,DSN=prefix.TOTAL.LINKLIB
//           DD  DISP=SHR,DSN=prefix.TOTAL.INTERFLM
//*
//CSIPARM    DD  DISP=SHR,DSN=prefix.TOTAL.CSIPARM(TXPFOCUS)
//CSISYSIN   DD  DISP=SHR,DSN=prefix.TOTAL.DATA(SUPRCNFG)
```

If you wish to use different TOTAL systems, you can specify the required libraries, CSIPARM, in the //STEPLIB DD statement in the ISTART JCL.

Configuring the Adapter for TOTAL

To configure the Adapter for TOTAL, you simply identify it for configuration from the Web Console.

Note: You will supply specific connection and authentication information within the metadata. See *Managing TOTAL Metadata* on page 49-9.

Procedure: How to Configure the Adapter for TOTAL From the Web Console

Using the Web Console navigation pane:

1. Choose *Data Adapters*.
2. Click the *Adapters* folder to open it.
3. Under *Add*, click *Available on other platforms*.
4. Select *Total* from the list provided in the right-hand pane.
5. Click *Configure*.

Mapping Considerations for TOTAL

In this section:

This topic explains how the Adapter for TOTAL allows you to describe and report from a TOTAL database structure. These concepts affect the way TOTAL files are described to the server.

Mapping Concepts

In order for the server to read a TOTAL database, the user or project designer must prepare a Master File on the server that corresponds to the TOTAL database structure.

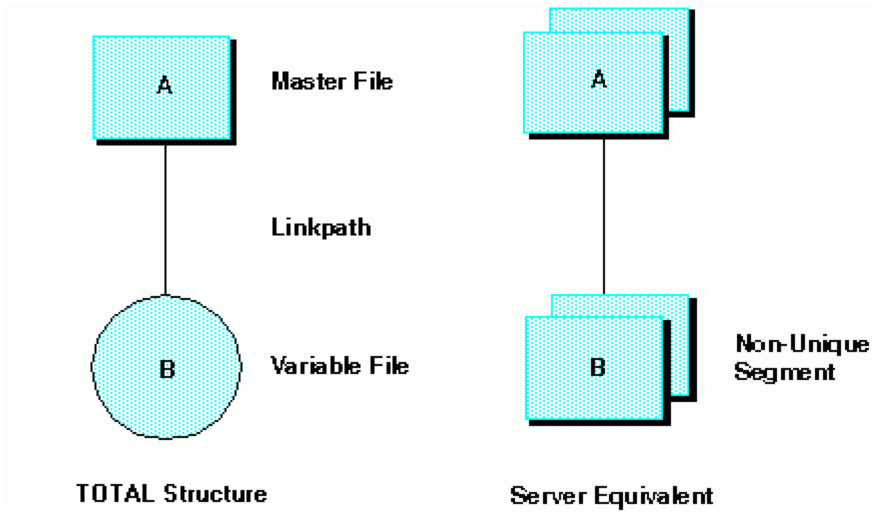
Note: The Master File on the server should not be confused with a TOTAL master file.

The following chart summarizes the server representations of different TOTAL structures:

TOTAL File	**Server Representation**
Single master file	Single segment
Single variable file	Single segment
Variable file accessed from a master file through one linkpath.	Single non-unique segment; child of the segment corresponding to the variable file.
Master file accessed from a variable file through a control key.	Single unique segment; child of the segment corresponding to the variable file.
Variable file accessed from a master file through several linkpaths.	Several non-unique segments, one for each linkpath. These segments are children of the segment corresponding to the Master File.

TOTAL master files cannot be linked to other TOTAL master files, and variable files cannot be linked to other variable files.

The following diagram depicts a simple TOTAL database structure and the equivalent Master File on the server.



Mapping TOTAL Structures

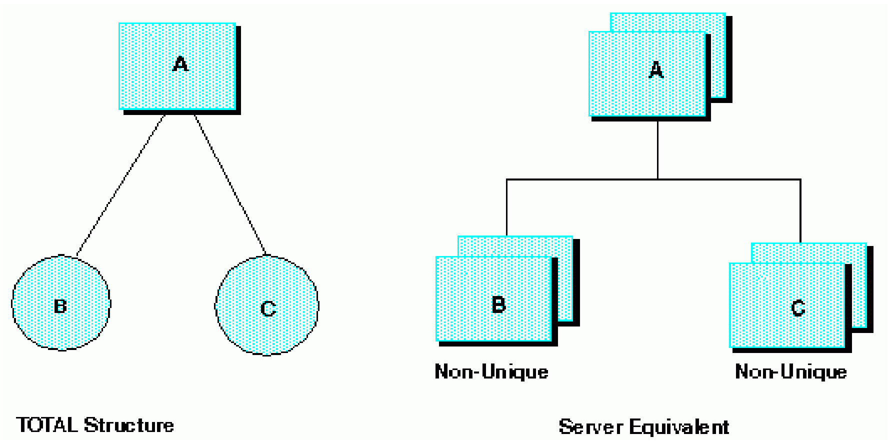
Four TOTAL structures that need special consideration when mapping to server Master Files are:

- One file linked to two or more files.
- One file linked to another file by multiple linkpaths.
- Variable files linked to master files in a ring structure.
- Coded variable files.

One TOTAL File Linked to Two or More Files

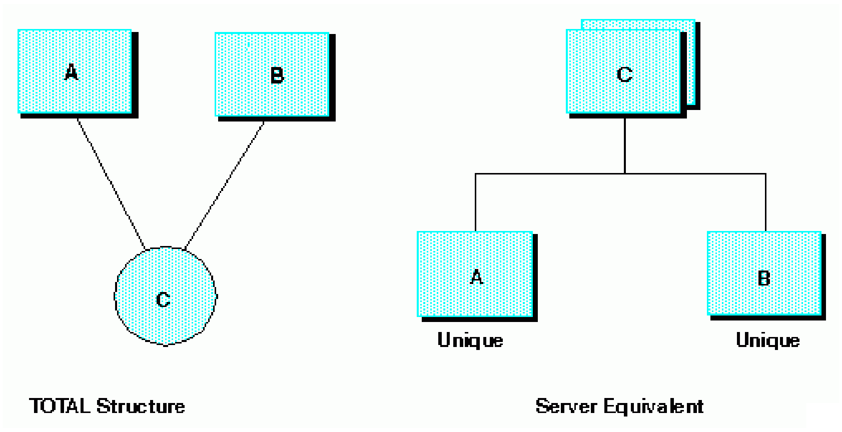
A TOTAL structure can consist of one master file linked to two or more variable files. To map such a structure to a Master File on the server, represent the TOTAL master file as the parent segment and the variable files as its children.

The following diagram depicts a TOTAL structure consisting of one master file linked to two variable files, and the equivalent server structure. (The relative positions of segments B and C under segment A are not significant.)



A TOTAL structure can also consist of two or more master files linked to one variable file. To map such a structure to a Master File on the server, represent the variable file as the parent segment and the master file as its children.

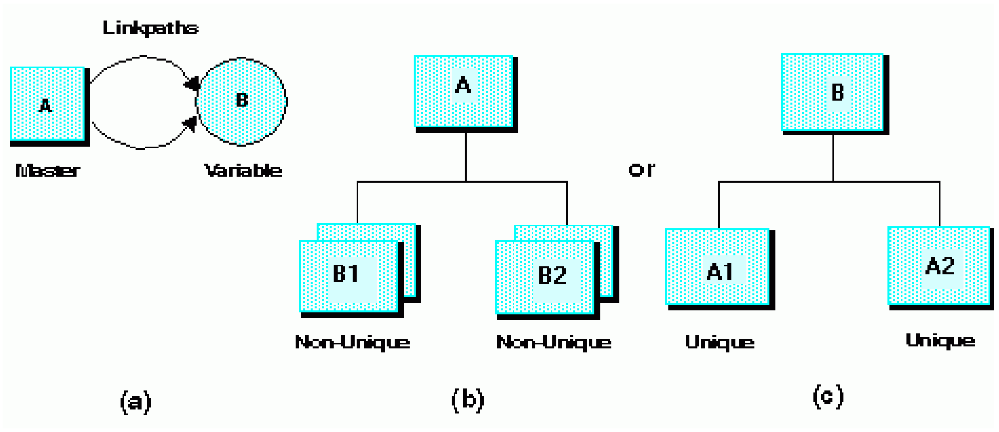
The following diagram shows a TOTAL structure consisting of two master files linked to one variable file, and the server equivalent. (The relative positions of segments B and C under segment A are not significant.)



One File Linked to Another File by Multiple Linkpaths

When a variable file is linked to a master file through two or more linkpaths, the same TOTAL record will be represented by different segments in a Master File on the server—one for each distinct access path.

The following diagram shows a TOTAL structure (a) consisting of a master file and variable file linked together by two linkpaths, and two possible server equivalents (b) and (c). In the first server representation (b), notice that the variable records accessed from the master record are represented by non-unique segments; in the second server representation (c), the master record accessed from the variable record through two different paths is represented by two unique segments.



Ring Structures

A TOTAL database structure can resemble a ring: alternate master and variable files linked in a circle, as illustrated in section (a) of the diagram that follows.

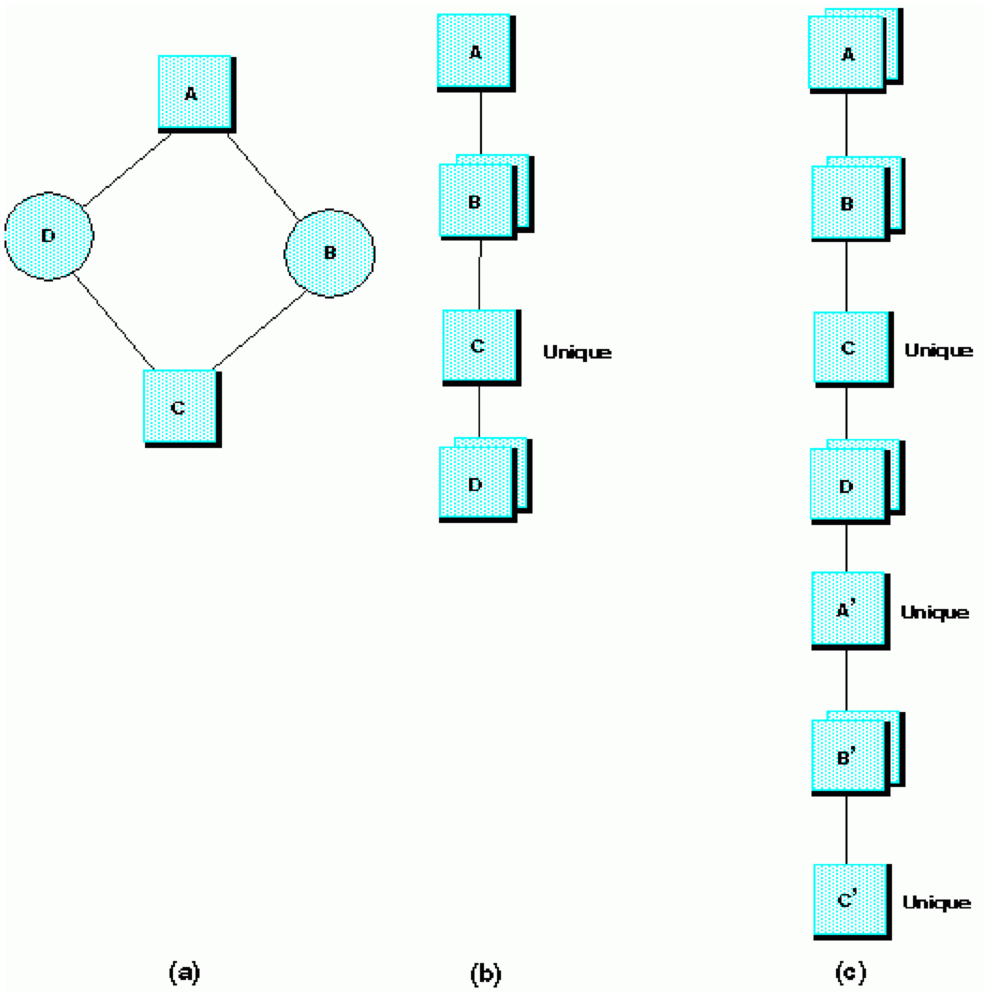
To represent this structure in a Master File on the server, begin the Master File with one file, and represent each file going around the ring as the child segment of the file before it. The result is a straight (linear) descending line of segments, as shown in section (b) of the diagram. This, however, is only part of the mapping.

In a TOTAL ring structure, one can start at any file and reach all the other files by traveling through the linkpaths around the ring. There are four possible sequences for traveling clockwise around the ring.

- Starting from file A: A, B, C, D
- Starting from file B: B, C, D, A
- Starting from file C: C, D, A, B
- Starting from file D: D, A, B, C

The corresponding Master File on the server must show all of these segment sequences going from top to bottom (or all these sequences going from bottom to top). To map the TOTAL ring structure completely, you must repeat some of the segments at the bottom of the description and give the repeated segments new segment and field names in the Master File on the server, while keeping the field aliases the same.

Section (c) or the following diagram illustrates how this is done for the ring structure shown in section (a). Segments A′, B′, and C′ are duplicates of segments A, B, and C. These repeated segments give the Master File on the server the four possible sequences of TOTAL files going from top to bottom: ABCD, BCDA, CDAB, and DABC.



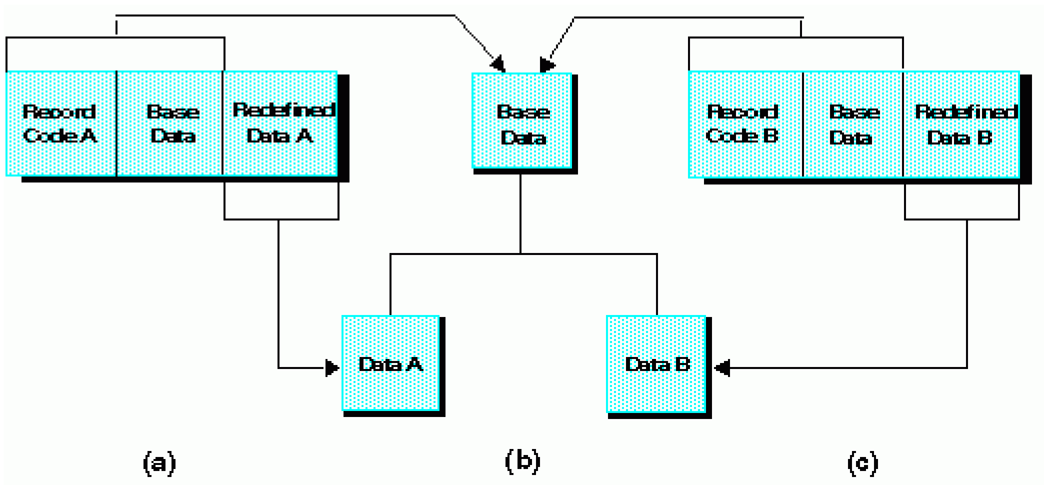
Coded Variable Files

Unlike other TOTAL files, you represent a coded variable file as several segments in a Master File on the server. Place the record code and base data fields in a non-unique parent segment, and each possible configuration of the redefined data fields in a unique child segment. The diagram that follows shows two records of a sample coded variable file, and the placement of its data in the equivalent Master File segments.

Redefined data segments must always be the children of base data segments. Therefore, you cannot make a redefined data segment the root segment of a Master File on the server. You also cannot invert a Master File structure so that the new root segment is a redefined data segment, or so that a redefined data segment is placed in the path connecting the old and the new root segments.

Base data segments can have direct descendants that are Master File segments if the control-key fields corresponding to those master files reside in the base portion of the coded variable file.

The following diagram shows two types of records from a coded variable file (a) and (b), and the placement of their data in the equivalent server structure (c).



Managing TOTAL Metadata

This topic explains how the adapter allows you to describe and report from TOTAL data sources. These concepts affect the way TOTAL files are described to the server.

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you must create a Master File and an Access File that describes the structure of the data source and the server mapping of the TOTAL data types.

Master File

Once you have decided how to map a TOTAL database structure on the server, your next task is to set up two new files—the Master File and the Access File.

The Master File can contain a File record, Segment records, and Field records. The keywords are described in the reference sections that follow. (For information about Access File keywords, see *Access File* on page 49-12.)

Reference: File Keywords

Keyword	Format or Value
`FILENAME`	Server name for the TOTAL database and for the Master File that describes it. The same as the member name of the Master File in the MASTER PDS. Eight characters maximum.
`SUFFIX`	TOTIN when accessing TOTAL.

Reference: Segment Keywords

Keyword	Format or Value
`SEGNAME`	Name of the segment. The name of each segment is unique to the file, even if several segments represent the same TOTAL file. The segment name can be different from the TOTAL file name. Eight characters maximum.

`SEGTYPE`	Type of segment:
	`U`
	Unique segment. Represents TOTAL master files when the segment is not the root segment in the server Master File or the redefined parts of coded variable files.
	`S`
	Non-unique segment. Represents TOTAL master files when the segment is the root segment of the server Master File, variable files, and the base data part of coded variable files.
	`KU or KLU`
	Remotely described unique segment. See *Remote Descriptions* on page 49-11.
	`KL or KM`
	Remotely described non-unique segment. See *Remote Descriptions* on page 49-11.
	For the root segment, you can assign SEGTYPE a null value by either typing SEGTYPE=$ or by omitting SEGTYPE altogether.

`PARENT`	Name of the parent segment. Each descendant segment must include the PARENT attribute to identify its parent in the hierarchy.
`CRFILE`	For remotely described segments. See *Remote Descriptions* on page 49-11.

Reference: Field Keywords

Keyword	Format or Value
`FIELD`	Any name; 66 characters is the maximum length.
`ALIAS`	Name of the field for TOTAL. The total length must not exceed 66 characters.
`USAGE`	Data format of the field as processed by the server. The data formats are the same as for other types of external files. See *Sample Access File* on page 49-14. For control-key fields, assign the USAGE format the same kind of format (packed, floating, integer, and so on) as the ACTUAL keyword, although you can assign different display and editing options (for example, a control-key field with an ACTUAL format of I7 can have a USAGE format of I7C to insert commas).
`ACTUAL`	Data format of the field as it is in the TOTAL database. The data formats are the same as for other types of external files. For a table that converts COBOL to server data formats, see *Sample Access File* on page 49-14.

Remote Descriptions

The SEGTYPEs KL, KM, KU, and KLU specify segments with remote descriptions. These segments have their field attributes described in other Master Files that the server can read at run time. Segments with remote descriptions are useful in situations where several Master Files describe the same TOTAL database. You can describe the fields of a segment in one file, and segments in other files can refer to the first file to avoid having to spell out the field descriptions again.

The separately described segment must have the same name in the file where it is defined and in the file that references it. The attribute CRFILE identifies the Master File in which the segment is defined. For example:

```
SEGMENT=SALES, PARENT=DEPT, SEGTYPE=KLU, CRFILE=RECORDS,$
```

The descriptions of fields of the SALES segment in this file are obtained from the SALES segment in the RECORDS Master File, where they must physically reside: SALES cannot be a KL or KLU segment there. The server reads only the field attributes from the segment, not the segment attributes. The RECORDS file itself need not even be the description of a file as long as it describes the named segment.

If the root segment itself is described separately, you can specify its SEGTYPE as either KM or KU, since it does not matter if root segments are unique or non-unique.

When a TOTAL file is represented by two or more segments in the Master File on the server, only one of the segments can have a remote description. This prevents the use of duplicate field names and aliases.

Segments with remote descriptions provide a convenience that saves typing; there are no logical implications regarding parent-child relationships and how they are implemented.

Access File

Reference:

DBMOD Record

SEGMENT Record

Example:

Sample Access File

The Access File provides the information necessary for selecting the appropriate TOTAL database and access strategy (that is, the linking of items between different tables).

Access Files are members of the partitioned data set allocated to the DDNAME ACCESS. The member name of an Access File must be the same as the member name of the corresponding Master File on the server. The member name is not related to names in the TOTAL database.

The Access File contains two types of logical records: DBMOD and SEGMENT.

Reference: DBMOD Record

Keyword	**Value**
`DBMOD`	TOTAL database descriptor module.
`RELEASE`	TOTAL Release number. Legal values are 6.3 for Release 6.3 and Release 8.0. Required only for Release 6.3.

Reference: SEGMENT Record

The following is a list of attributes of the Access File.

Keyword	Value
`SEGNAME`	Name of the segment whose TOTAL characteristics are described in this SEGNAME record. This keyword is required in every SEGNAME record.
`FILENAME`	Name of the TOTAL file that the segment represents. This keyword is required in every SEGNAME file.
`FILETYPE`	Type of TOTAL file. Legal values are MASTER for Master Files, VARIABLE for standard variable files, CODED-BASE for the base data portion of coded variable files, and CODED-REDEF for the redefined data portion of coded variable files. This keyword is required in every SEGNAME record.
`LINKPATH`	TOTAL field name of the linkpath used to link this segment to its parent in the Master File on the server. This value is required for TOTAL master and variable files because a Master File segment may become the parent of a variable file segment after a file inversion. **Note:** The TOTAL field name for the linkpath is required, not the field name in the Master File on the server. This keyword is required in all records except the SEGNAME record, which describes the root segment in the Master File on the server. The SEGNAME record must omit this keyword.
`MASTERKEY`	Master File field name or alias for the TOTAL control-key field. If the FILETYPE of this segment is MASTER, then the field must reside in this segment. Even if this segment is the root segment of the server Master File, without a parent to be linked to, the MASTERKEY field must still specify the control key of this segment since this information is used to select the entry segment traversal method. If the FILETYPE of this segment is VARIABLE or CODED-BASE, then the MASTERKEY field must be the same field as the MASTERKEY of the segment's parent, which represents a TOTAL master file. This keyword is required for segments with a FILETYPE of MASTER, VARIABLE, or CODED-BASE. It is not required for the root segment.

Keyword	Value
VARIABLEKEY	Master File field name or alias of the field in the variable file whose values correspond to those in the MASTERKEY field. If this segment is a variable file segment, the VARIABLEKEY field resides in this segment and the MASTERKEY field resides in the segment's parent. If this segment is a Master File segment, the VARIABLEKEY field resides in the segment's parent, which must be a variable file (standard or coded) segment. This keyword is required for segments with a FILETYPE of MASTER, VARIABLE, and CODED-BASE. It is prohibited for the root segment.
RECORDCODE	TOTAL record code that identifies a record in this segment. This keyword is required for segments with a FILETYPE of CODED-REDEF. It is prohibited for all other segments.
CODEFIELD	Master File field name or alias for the TOTAL record code field. This keyword is required for segments with a FILETYPE of CODED-BASE. It is prohibited for all other segments.
REFERFIELD	Name or alias of a Master File field, used as follows: • If a request references this field, the adapter makes available to the server the internal reference point for each record retrieved as a value of this field. • If this field is in the root segment of the Master File and a request tests it for internal reference point values, retrieval may be by a direct read on the internal reference point. For information about the role that internal reference points have in record retrieval, see *Entry Segment Retrieval Using an Internal Reference Point* on page 49-38. This keyword is prohibited for segments with a FILETYPE of MASTER.

Example: Sample Access File

A logical record in the Access File consists of a list of attributes (keyword = value pairs) separated by commas and terminated by a comma and dollar sign (,$). The list is freeform and can span several lines. You can generally specify keywords (see *SEGMENT Record* on page 49-13) in any order. The following is a sample Access File member:

```
*SAMPLE EMPLOYEE SKILLS ACCESS FILE
DBMOD=EMKPLDBM,$
SEGNAME=EMPSEG, FILENAME=EMPL,FILETYPE_MASTER,  MASTERKEY=SSN,$
SEGNAME=SKILSEG,FILENAME=SKIL,FILETYPE=VARIABLE,MASTERKEY=SSN,
   VARIABLEKEY=EMPLNO,LINKPATH=EMPLLKSK,$
```

Values that contain commas, equal signs, or dollar signs are enclosed in single quotation marks.

Blank lines are ignored, as are lines that start with an asterisk in column 1, which can be used to insert comments.

You may omit all the keywords in SEGNAME records except for the SEGNAME value. However, when you omit the keywords, their values become positional and must appear in the order shown in *SEGMENT Record* on page 49-13. Indicate the omission of keyword values with two adjacent commas (,,). For example, the following two SEGNAME records are equivalent:

```
SEGNAME=SKILSEG, FILENAME=SKIL, FILETYPE=VARIABLE,
MASTERKEY=SSN,VARIABLEKEY=EMPLNO, LINKPATH=EMPLLKSK, REFERFIELD=IRPFLD,$
```

```
SEGNAME=SKILSEG, SKIL, VARIABLE, EMPLLKSK, SSN, EMPLNO,,, IRPFLD,$
```

Notice the three consecutive commas after the EMPLNO value in record (2), which mark the absence of two keyword values: RECORDCODE and CODEFIELD.

Data Type Support

Reference:

ACTUAL Length

USAGE Length

Server/COBOL Format Conversion Table

TOTAL data formats are usually expressed by COBOL record descriptions. The tables in this section describe how to translate COBOL record descriptions into Master File segment declarations.

Reference: ACTUAL Length

The server ACTUAL length is determined by the number of internal storage bytes used by COBOL in storing the data item. The ACTUAL lengths for the different types of data are as follows. Note that the number of digits in the COBOL PICTURE clause does not count the sign (S) or the implied decimal point (V).

Alphanumeric (A)	Number of characters described in the PICTURE clause.
Zoned decimal (Z)	Number of digits described in the PICTURE clause.
Integer (I)	2 bytes if the PICTURE clause has one to four digits. 4 bytes if the PICTURE clause has five to nine digits.
Floating point (F)	4 bytes
Double Precision (P)	8 bytes
Packed Decimal (0)	If the number of digits in the PICTURE clause is even, divide the number by 2 and add 1. Otherwise, add 1 to the number and divide by 2.

Reference: USAGE Length

The USAGE length describes the number of places required to print the data item, including decimal points, in a report. Allow for the maximum possible number of characters or digits when determining USAGE length. You can add any valid server edit option to the USAGE description without increasing the length.

Reference: Server/COBOL Format Conversion Table

Notes identified by asterisks (*) follow the chart.

COBOL USAGE	COBOL PICTURE	BYTES OF INTERNAL STORAGE	Server ACTUAL FORMAT	Server USAGE FORMAT *
DISPLAY	X(4)	4	A4	A4
DISPLAY	S99	2	Z2	P3**
DISPLAY	9(5)V9	6	Z6.1	P8.1
COMP	S9	2	I2	I1
COMP	S9(4)	2	I2	I4
COMP	S9(5)	4	I4	I5
COMP	S9(9)	4	I4	I9
COMP-1	***	4	F4	F6****
COMP-2	***	8	D8	D15****
COMP-3	9	1	PI	PI
COMP-3	S9V99	2	P2	P5.2
COMP -3	9(4)V9(3)	4	P4	P8.3

Note:

*	The usage format lengths shown are minimum values. Additional edit options may also be added.
**	In USAGE formats, an extra character position is required for the minus sign if negative values are expected.
***	Picture clauses are not permitted for internal floating point items.
****	The usage format length should allow for the maximum possible number of digits.

Sample Data Descriptions for TOTAL

Example:

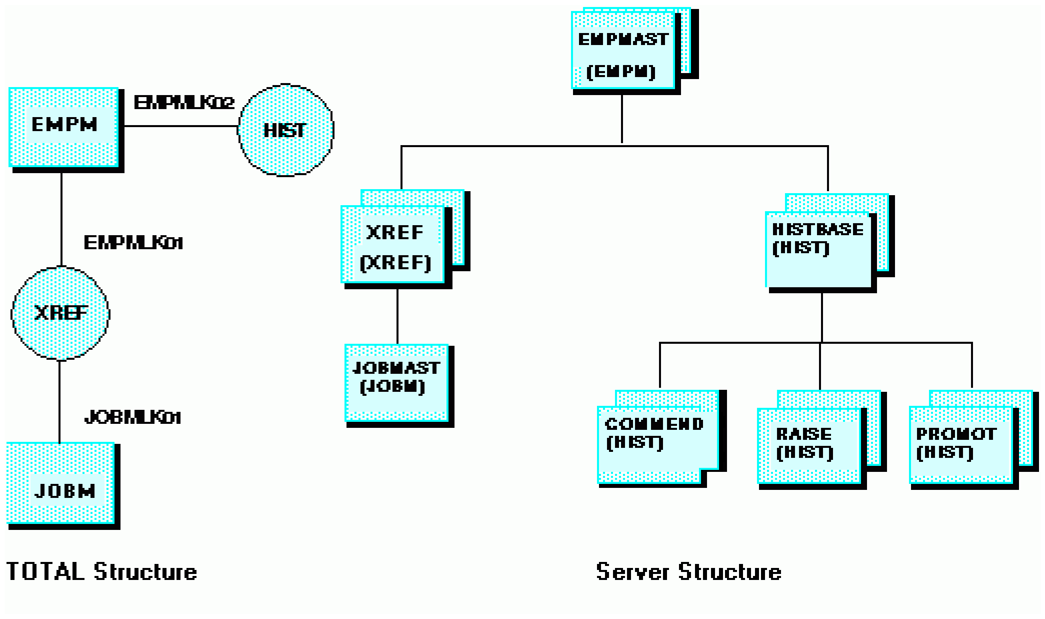
TOTAL Database Description

Server Master File

Server Access File

A TOTAL database is described to the server using a Master File and Access File. This sample database holds a firm's employee files. It contains instances of each type of TOTAL file: master, variable, and coded variable.

The following is a diagram of the TOTAL database structure and the equivalent Master File hierarchy on the server. The linkpaths in the TOTAL database diagram are labeled with the names of the linkpaths' fields. Note that the HIST file is a coded variable file in the TOTAL structure. In the server structure, each segment in the Master File hierarchy shows the name of the TOTAL file it represents in parentheses.



Example: TOTAL Database Description

The following is the DBMOD source DDL statement for a TOTAL data source:

```
BEGIN-MASTER-DATA-SET:
DATA-SET-NAME=EMPM
IOAREA=MASl
MASTER-DATA:
EMPMROOT=8
EMPMCTRL=4               CONTROL KEY (SSN)
EMPMLKOI=8               LINK TO XREF FILE
EMPMLKO2=8               LINK TO HIST FILE
EMPMNAME=24              EMPLOYEE NAME
EMPMBDAY=4               EMPLOYEE BIRTHDATE
END-DATA:
  .
  .
END-MASTER-DATA-SET:

BEGIN-MASTER-DATA-SET:
DATA-SET-NAME=JOBM
IOAREA=MASl
MASTER-DATA:
JOBMROOT=8
JOBMCTRL=4               CONTROL KEY (JOB CODE)
JOBMLKO1=8               LINK TO XREF FILE
JOBMDESC=24              JOB DESCRIPTION
JOBMMINS=4               MINIMUM SALARY
JOBMMAXS=4               MAXIMUM SALARY
END-DATA:
  .
  .
END-MASTER-DATA-SET:

BEGIN-VARIABLE-ENTRY-DATA-SET:
DATA-SET-NAME=XREF
IOAREA=VARl
BASE-DATA:
XREFEMPM=4=EMPMCTRL       KEY FROM EMPM FILE (SSN)
EMPMLKO1=8                LINK TO EMPM FILE
XREFJOBM=4=JOBMCTRL       KEY FROM JOBM FILE (JOB CODE)
JOBMLKO1=8                LINK TO JOBM FILE
XREFDATE=4                DATE OF START OF EMPLOYMENT
XREFSLRY=4                CURRENT SALARY
  .
  .
END-VARIABLE-ENTRY-DATA-SET
```

```
BEGIN-VARIABLE-ENTRY-DATA-SET:
DATA-SET-NAME=HIST
IOAREA=VAR1
BASE-DATA:
HISTEMPM=4                     KEY FROM EMPM FILE (SSN)
EMPMLKO2=8                     LINK TO EMPM FILE
HISTCODE=2                     RECORD CODE (CO, RA OR PR)
HISTDATE=4                     DATE OF HISTORY RECORD
HISTAREA=88                    VARIABLE DATA AREA
RECORD-CODE=CO                 SUPERVISOR COMMENDATION RECORD
HISTSUPR=24                    SUPERVISOR NAME
HISTCCOM=40                    SUPERVISOR'S COMMENTS
*FILLER*=24
RECORD-CODE=RA                 RAISE RECORD
HISTOSAL=4                     OLD SALARY
HISTNSAL=4                     NEW SALARY
HISTRCOM=40                    REASON FOR RAISE
*FILLER*=40
RECORD-CODE=PR                 PROMOTION RECORD
HISTOJOB=24                    OLD JOB DESCRIPTION
HISTNJOB=24                    NEW JOB DESCRIPTION
HISTPCOM=40                    SUPERVISOR'S DESCRIPTION
END-DATA:
  .
  .
END-VARIABLE-ENTRY-DATA-SET:
```

Example: Server Master File

```
FILE=EMPLOYEE, SIJFFIX=TOTIN

SEGNAME=EMPMAST, SEGTYPE=S
FIELDNAME=SSN,         ALIAS=EMPMCTRL,  USAGE=I9L,   ACTUAL=I4,  $
FIELDNAME=NAME,        ALIAS=EMPNAME,   USAGE=A24,   ACTUAL=A24, $
FIELDNAME=BIRTHDAY,    ALIAS=EMPMBDAY,  USAGE=I6MDY, ACTUAL=I4,  $

SEGNAME=XREF, PARENT=EMPMAST, SEGTYPE=S
FIELDNAME=XSSN,        ALIAS=XREFEMPM,  USAGE=I9L,   ACTUAL=I4,  $
FIELDNAME=XJOBCODE,    ALIAS=XREFJOBM,  USAGE=A4,    ACTUAL=A4,  $
FIELDNAME=START DATE,  ALIAS=XREFDATE,  USAGE=I6MDY, ACTUAL=I4,  $
FIELDNAME=SALARY,      ALIAS=XREFSLRY,  USAGE=I6,    ACTUAL=I4,  $

SEGNAME=JOBMAST, PARENT=XREF, SEGTYPE=U
FIELDNAME=JOBCODE,     ALIAS=JOBMCTRL,  USAGE=A4,    ACTUAL=A4,  $
FIELDNAME=DESCRIPTION, ALIAS=JOBMDESC,  USAGE=A24,   ACTUAL=A24, $
FIELDNAME=MIN SALARY,  ALIAS=JOBMMINS,  USAGE=I6,    ACTUAL=I4,  $
FIELDNAME=MAX=SALARY,  AL1AS=JOBMMAXS,  USAGE=I6,    ACTUAL=14,  $

SEGNAME=HISTBASE, PARENT=EMPMAST, SEGTYPE=S
```

```
FIELDNAME=HSSN,         ALIAS=HISTEMPM,  USAGE=I9MDY, ACTUAL=I4,  $
FIELDNAME=TYPE,         ALIAS=HISTCODE,  USAGE=A2,    ACTUAL=A2,  $
FIELDNAME=DATE,         ALIAS=HISTDATE,  USAGE=I6MDY, ACTUAL=I4,  $
FIELDNAME=HIST_IRP,     ALIAS=HIRP,      USAGE=I9,    ACTUAL=I4,  $

SEGNAME=COMMEND, PARENT=HISTBASE, SEGTYPE=U
FIELDNAME=SUPERVISOR,   AL1AS=HISTSUPR,  USAGE=A24,   ACTUAL=A24, $
FIELDNAME=C_COMMENTS,   ALIAS=HISTCCOM,  USAGE=A40,   ACTUAL=A40, $

SEGNAME=RAISE, PARENT=HISTBASE, SEGTYPE=U
FIELDNAME=OLD_SALARY,   ALIAS=HISTOSAL,  USAGE=I6,    ACTUAL=I4,  $
FIELDNAME=NEW-SALARY,   ALIAS=HISTNSAL,  USAGE=I6,    ACTUAL=I4,  $
FIELDNAME=R_COMMENTS,   ALIAS=HISTRCOM,  USAGE=A40,   ACTUAL=A40, $

SEGNAME=PROMOT, PARENT=HISTBASE, SEGTYPE=U
FIELDNAME=OLD JOB,      ALIAS=HISTOJOB,  USAGE=A24,   ACTUAL=A24, $
FIELDNAME=NEW-JOB,      ALIAS=HISTNJOB,  USAGE=A24,   ACTUAL=A24, $
FIELDNAME=P _COMMENTS,  ALIAS=HISTPCOM,  USAGE=A40,   ACTUAL=A40, $
```

Example: Server Access File

```
DBMOD=EMPLMOD, $

SEGNAME=EMPMAST,     FILENAME=EMPM,     FILETYPE=MASTER,
MASTERKEY=SSN, $

SEGNAME=XREF,        FILENAME=XREF,     FILETYPE=VARIABLE,
LINKPATH=EMPMLKO1,   MASTERKEY=SSN,     VARIABLEKEY=XSSN, $

SEGNAME=JOBMAST,     FILENAME=JOBM,     FILETYPE=MASTER,
LINKPATH=JOBMLKO1,   MASTERKEY=JOBCODE, VARIABLEKEY=XJOBCOnE, $

SEGNAME=HISTBASE,    FILENAME=HIST,     FILETYPE=CODED-BASE,
LINKPATH=EMPMLKO2,   MASTERKEY=SSN,     VARIABLEKEY=HSSN,
CODEFIELD=TYPE,      REFERFIELD=HIST_IRP, $

SEGNAME=COMMEND,     FILENAME=HIST,     FILETYPE=CODED-REDEF,
RECORDCODE=CO, $

SEGNAME=RAISE,       FILENAME=HIST,     FILETYPE=CODED-REDEF,
RECOROCODE=RA, $

SEGNAME=PROMOT,      FILENAME=HIST,     FILETYPE=CODED-REDEF,
RECORDCODE=PR, $
```

File Retrieval

You often have a choice of how to represent the TOTAL database structure as a Master File structure on the server. Familiarity with server retrieval logic will help you decide between alternate Master File structures.

This topic describes options for:

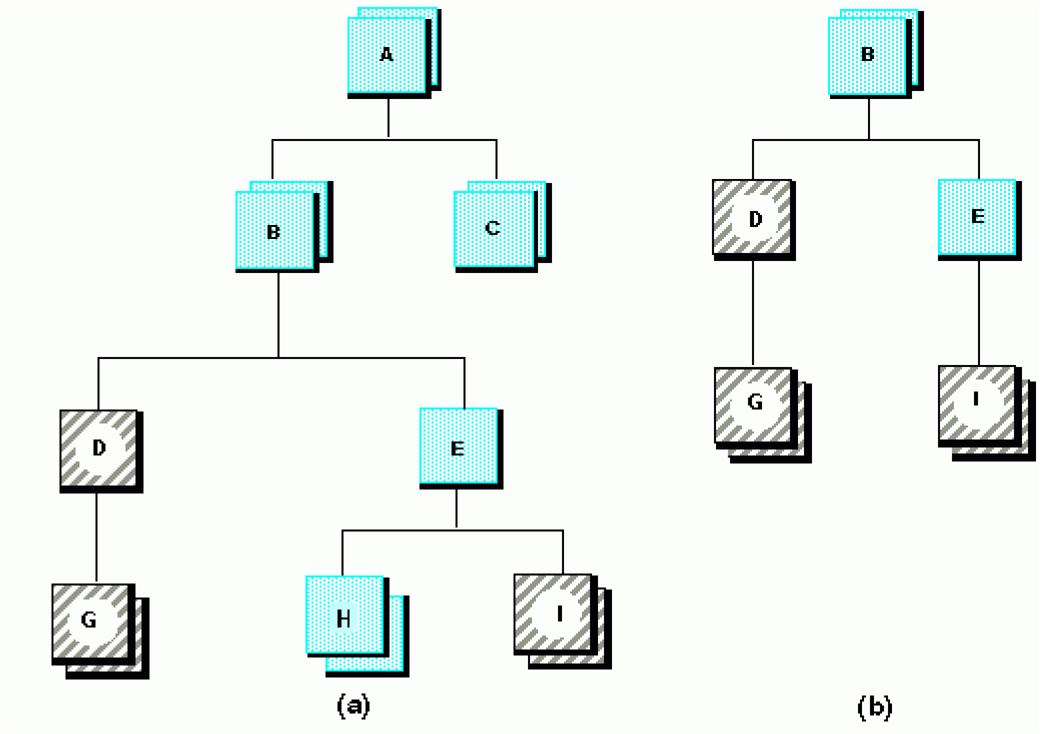
- Controlling order of retrieval based on subtrees and record selection tests.
- Handling processing of missing records.
- Inverting files.

Note that you can also affect file retrieval by joining TOTAL files to files from any other supported data source.

Retrieval Subtree

When the server retrieves records for a report, it first constructs a smaller subtree from the Master File structure. The subtree consists of all the segments that contain a field which was referenced by the retrieval request, plus any other segments needed to tie these segments into a connected structure.

In the following diagram, section (a) shows a sample structure diagram. If a request needs records from segments D, G, and I, the server constructs the subtree shown in section (b). Although segments E and B contain nothing needed for the request, they are included in the subtree to tie segment I to segments D and G. Since the segments of the subtree are all descendants of segment B, segment B is the entry segment of the subtree.

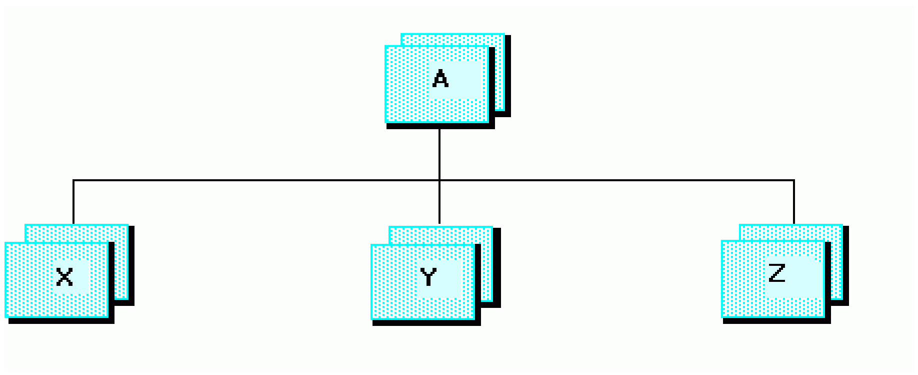


Retrieval Sequence

In this section:

The order of record retrieval can be determined by examining a structure diagram of the subtree that the server is using to read the database. If there is no unique segment except for the entry segment, then the order of retrieval is from top to bottom, left to right.

In the following diagram, the order of retrieval would be: A, then all of the Xs within A, then all of the Ys within the same A, then all of the Zs within that A, then the next A, and so on. This subtree shows only non-unique children.



Retrieval Sequence With Unique Segments

How to:

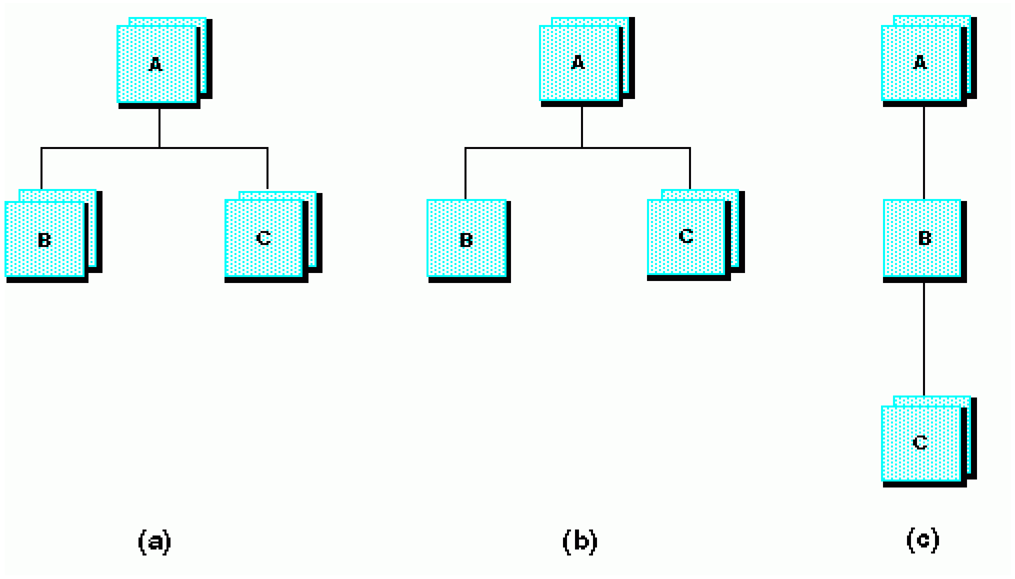
Diagram a Retrieval View

The retrieval sequence with subtrees containing unique segments is top to bottom, left to right, but the unique segments are treated differently. Since records in a unique segment correspond one-for-one with the records in its parent, during retrieval the server treats the unique segment as an extension of its parent. Therefore, if a parent segment has unique and non-unique children, the unique children are always retrieved first.

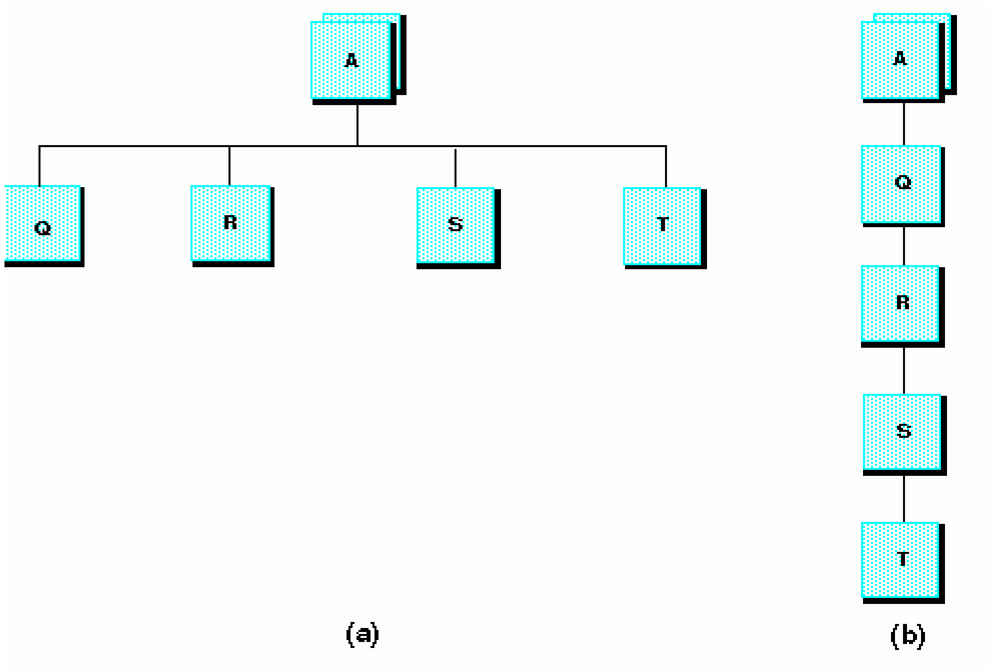
To illustrate the retrieval sequence in such a subtree, you can redraw the diagram so that the order of retrieval follows the top to bottom, left to right rule. This is called a retrieval view. See *Diagram a Retrieval View* on page 49-26.

A retrieval view also shows the sort statements that are valid. In a sort statement, such as BY or ACROSS, the segment containing the sort field must lie on the same path as the segment containing the field being sorted. That is, one segment must be descended from the other. The path connecting the two segments is determined by the retrieval view.

The following illustration includes three sample structure diagrams. In section (a) the statement PRINT B BY C is *invalid* because the segments containing fields B and C do not lie on the same path. However, if the segment containing B is a unique segment, as in section (b) of the diagram, then the two segments do lie on the same path, as shown by the retrieval view in section (c). For such a structure, the statement PRINT B BY C is *valid*.



The next illustration shows a subtree with only unique children (a), and its retrieval view (b). The server retrieves the segments in top to bottom, left to right order. Section (a) of the diagram shows the subtree. Its retrieval sequence is A, Q, R, S, and then T. The retrieval view in section (b) shows the leftmost unique segment as a child of the parent segment A, and each of the other three segments as the child of the segment to the left of it.



Syntax: How to Diagram a Retrieval View

The command CHECK can draw a diagram of the retrieval view

```
CHECK FILE filename PICTURE RETRIEVE
```

where:

filename

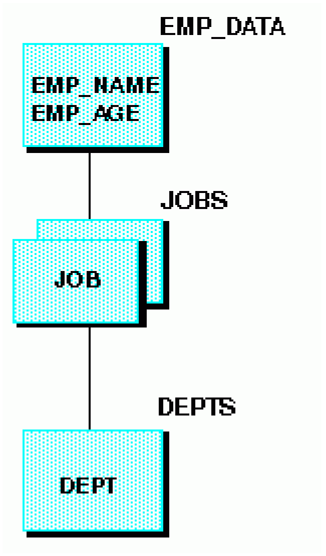
Is the name of the Master File, whose structure you wish to draw.

Effects of Record Selection Tests on Retrieval

If a record in a segment fails a record selection test (the IF/WHERE statements in the request that place conditions on records), then the server does not retrieve the corresponding records in descendant segments. For example, suppose you entered the following request, which uses the Master File EMPLOYEE:

```
SELECT JOB,DEPT FROM EMPLOYEE
 WHERE EMP_AGE GE 50
 ORDER BY DEPT;
```

Using the following subtree, every time a record in segment EMP_DATA has the value of the field EMP_AGE less than 50, the server ignores the corresponding records in segments JOBS and DEPTS, and retrieves the next record in segment EMP_DATA.



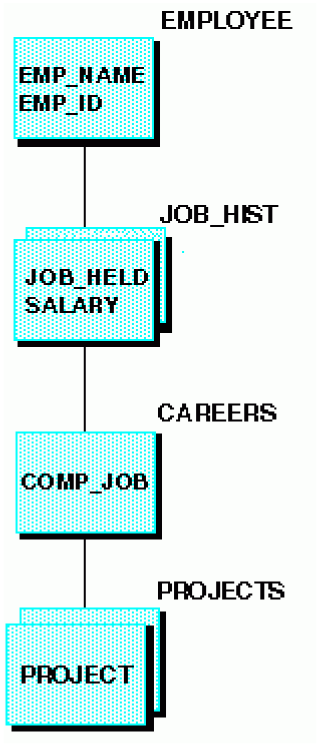
The higher in a structure you apply a record selection test, the more records the server skips if a record fails the test. For example:

1. EMPLOYEE with names and IDs of employees.
2. JOB_HIST with the jobs and salaries held by each employee.
3. CAREERS with a list of jobs offered by the company.
4. PROJECTS with the projects performed in each job.

This following request lists all employees involved in blueprint projects:

```
SELECT PROJECT,EMP_NAME FROM EMPLOYEE
WHERE PROJECT = 'BLUEPRINTS';
```

The server retrieves each employee's name, each job held, the corresponding job record in the CAREERS segment, and the projects listed for that job. It then tests each PROJECT field for the value BLUEPRINTS.



You can make the request more efficient by placing a record selection test on a segment higher in the structure. In this company, only draftsmen handle blueprints:

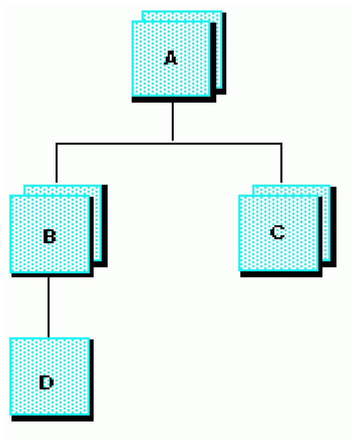
```
SELECT PROJECT,EMP_NAME FROM EMPLOYEE
WHERE PROJECT = 'BLUEPRINTS' AND JOB_HELD = 'DRAFTSMAN';
```

With this request, the server retrieves and tests PROJECT records only when the JOB_HELD field reads DRAFTSMAN.

Processing Selection Tests With Unique Segments

If a record in a unique segment fails a selection test, the server retrieves the next record of the parent of the segment, as illustrated in the next diagram. If a record in segment C fails a record selection test, the server retrieves the next record in segment C. But if a record in segment D fails a screening test, then the server retrieves the next record in segment B.

If the record in the entry segment fails a record selection test, the server retrieves the next record of the entry segment, even if the entry segment is unique.



Missing Records

After the server retrieves a record in a parent segment, it retrieves the corresponding records in the child segment. If the corresponding records are not there, then the processing of the parent record and the output depend on whether the child segment is unique or non-unique.

Handling Missing Records With Unique Segments

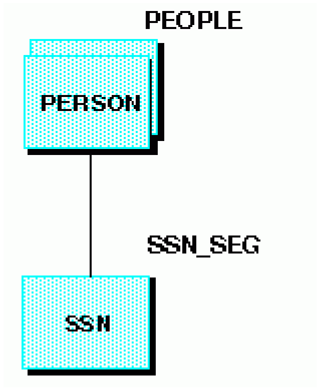
Example:

Processing Missing Records in a Unique Segment

If a unique child segment has a missing record, then the server creates a temporary record in place of the missing record, filling these fields with default values (blanks for alphanumeric fields, zeroes for numeric fields).

Example: Processing Missing Records in a Unique Segment

Suppose that a segment called PEOPLE contains the field PERSON. This segment has a unique child SSN_SEG containing the numeric field SSN for the social security number. Gary Smith does not have a social security number and so has no SSN record. In all reports that reference SSN, the entry GARY SMITH would show the default value, zero, for the SSN field.



Handling Missing Records With Non-Unique Segments

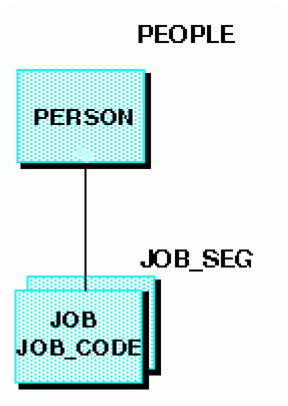
Example:

Handling Missing Records With Non-Unique Segments

If a non-unique child segment has a missing record, the result depends on the setting of the ALL parameter.

Example: Handling Missing Records With Non-Unique Segments

In this example, the segment PEOPLE is a unique segment and has a non-unique child segment JOB_SEG containing the fields JOB and JOB_CODE, as shown in the following diagram.



When the ALL setting is OFF, the server processes the parent record, provided that the request does not include record selection tests on fields in the child segment. Missing data is marked on the output by the NODATA character, which is usually a period (.).

If no jobs have been entered for Gary Smith, the report generated by the request

```
SELECT JOB, PERSON FROM PEOPLE
ORDER BY PERSON
```

omits mention of Gary Smith.

When the ALL setting is ON, the server processes the parent record, provided the request does not include record selection tests on fields in the child segment. Missing data is marked on the output by the NODATA character (a period, by default). The previous request now displays everyone's name, even those without JOB_SEG records:

```
PERSON              JOB
--------            ------------
SANDS JOHN          WELDER
SMALL BILL          TOOL-AND-DIE
SMITH GARY          .
THOMAS HENRY        WELDER
```

The NODATA character indicates that there is no record in the JOB_SEG segment for Gary Smith.

However, if the request contains screening conditions—for example,

```
SELECT JOB, PERSON FROM PEOPLE
WHERE JOB_CODE = 33
ORDER BY PERSON;
```

the output displays only the names of welders (JOB_CODE 33), omitting Gary Smith:

```
PERSON              JOB
--------            ------------
SANDS JOHN          WELDER
THOMAS HENRY        WELDER
```

If the ALL setting is PASS, the server processes the parent record regardless of the presence of record selection tests. Therefore, the previous sample request displays the names of both welders and people with no records in JOB_SEG:

```
PERSON              JOB
--------            ------------
SANDS JOHN          WELDER
SMITH GARY          .
THOMAS HENRY        WELDER
```

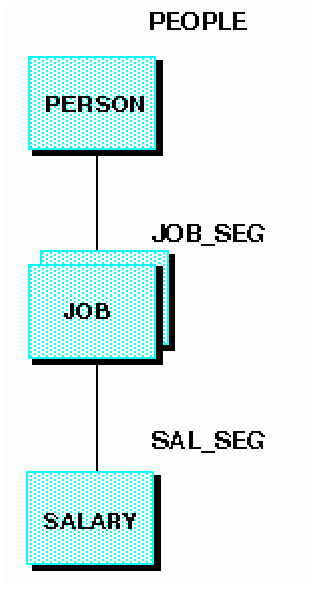
Processing Records With Missing Descendants

If the setting of the ALL parameter is OFF, you can still apply the effect of the ON setting to segments you choose. To do this, add the ALL prefix to one of the segment fields using a remote procedure call (RPC). The prefix causes the server to process records in that segment even if they have missing descendants, as long as the request does not have any record selection tests on the descendant record fields. This is true only for immediate descendants. If records are missing in children of descendants, the server does not process the parent record.

For example, suppose that the RPC

```
TABLE FILE PEOPLE
PRINT JOB AND SALARY BY ALL.PERSON
END
```

uses the following server structure:



The report will include people with no JOB_SEG records, but will not include people with JOB_SEG records but no corresponding SAL_SEG records.

Note: The ALL settings ON and PASS override the ALL prefix. The ALL prefix is only effective when the ALL setting is OFF.

File Inversion

How to:

Invert a File

Create an Inverted Structure Diagram

Example:

Inverting a File

Using File Inversion to Control Retrieval Paths

Using File Inversion to Execute a Request

You can alter a Master File structure at execution time by making one of the segments the root segment. The server then reverses some of the parent-child relationships among segments so that the segment you chose becomes the root segment of the structure. This process is called file inversion.

To invert a file, the server reverses parent-child relationships along the path linking the original root segment and the new root segment (root path). All other parent-child relationships remain unchanged.

Note that you cannot invert a file if the path linking the old and new root segments runs through segments representing the redefined data portions of coded variable files.

Syntax: How to Invert a File

To invert a file, follow the file name in your request with a period and the name of one of the fields in the segment you want for the root.

```
file.fieldname
```

where:

file

Is the name of the database.

fieldname

Is the name of a field in the new root segment.

Example: Inverting a File

The file SALES has a segment CUSTINFO containing the field CUSTOMER. To invert the file so that segment CUSTINFO is the new root segment, refer to the file as follows:

```
TABLE FILE SALES.CUSTOMER
```

Syntax: How to Create an Inverted Structure Diagram

The command CHECK can draw a diagram of the inverted Master File.

```
CHECK FILE filename.fieldname PICTURE
```

where:

filename

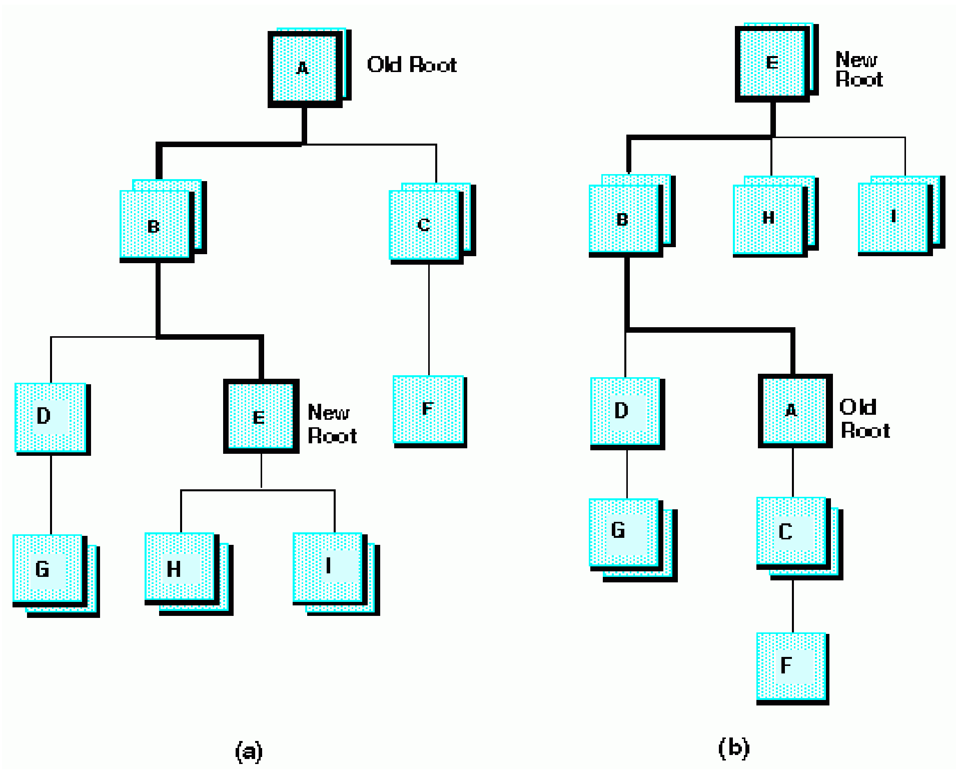
Is the name of the Master File on the server.

fieldname

Is the name of the one of the fields in the new root segment.

Example: Using File Inversion to Control Retrieval Paths

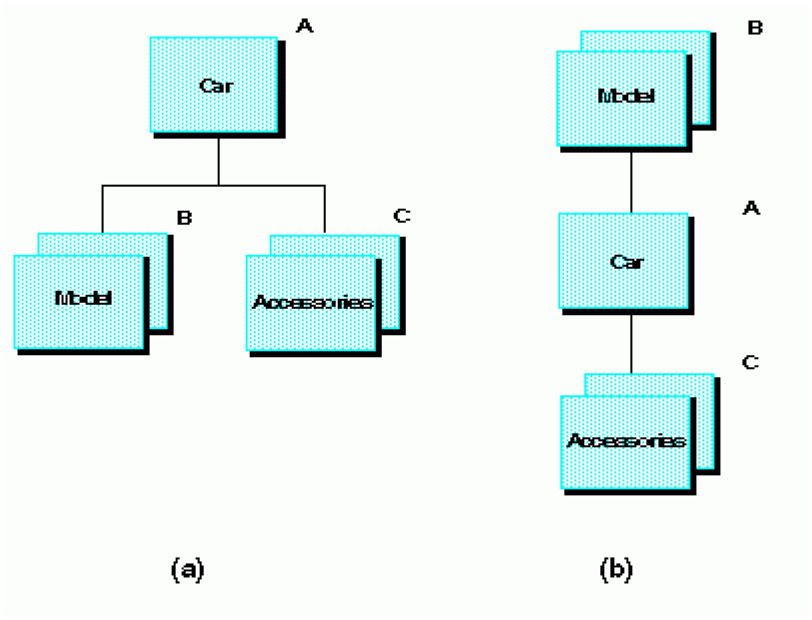
The following shows a Master File structure (a) and its inverted view (b) if segment E is made the root segment. Note that the relation of segment A to segment C remains unchanged since segment C is not in the root path. (The root path is shaded.)



File inversion is useful because it allows you to alter the sequence of retrieval at run time. The retrieval sequence can significantly affect the cost of producing a report.

Example: Using File Inversion to Execute a Request

Certain report requests that the server flags may be executable using inverted file views. However, record occurrences will multiply. Suppose that you are using the Master File structure pictured in section (a) of the following diagram



and you enter this request:

```
LIST ACCESSORIES BY MODEL
```

The request would cause a message because segments C and B are not on the same path.

However, in the inverted view shown in section (b) of the diagram, segment C is descended from segment B. Using the inverted view, the server will execute the request.

As it executes the request, the server pairs every record of segment C corresponding to a record in segment A with every record in segment B that also corresponds to that record in A. If, for example, A has two B descendants and four C descendants, the report will contain eight lines of output.

This effect is advantageous when it is necessary to pair every record associated with one linkpath to a record associated with another linkpath. However, at times, record pairing may produce undesirable results.

Navigation Strategies

To obtain all of the necessary records to fulfill a request, the Adapter for TOTAL navigates the TOTAL database using the following techniques:

- Retrieval from entry segments, which includes four variations:
  - Retrieval using a control key.
  - Retrieval using an internal reference points.
  - Retrieval using FINDX.
  - Retrieval using RDNXT.
- Retrieval from non-entry segments.

Subsequent sections discuss the navigation strategies used for each type of access.

Retrieval From Entry Segments

The TOTAL retrieval commands that the adapter issues in response to a request depend on the record selection tests (IF/WHERE statements) that the request places on entry segment fields. These commands, in decreasing order of preference, are:

For entry segments representing TOTAL master files:

1. READM on specified values of the control key.
2. Qualified serial retrieval using FINDX.
3. Straight serial retrieval using RDNXT.

For entry segments representing TOTAL variable files:

1. READD on specified internal reference points.
2. Qualified serial retrieval using FINDX.
3. Straight serial retrieval using RDNXT

The intent behind the selection logic is to implement as many record selection tests as possible at the TOTAL level. This minimizes the number of I/O operations required to read the desired data. Straight serial reads are the least desirable as they retrieve every entry level record and leave the server with the task of selecting those records that meet the tests and discarding the rest.

You can explicitly limit the number of records retrieved from TOTAL and the number of records accepted by the server by including the READLIMIT and RECORDLIMIT keywords in a remote procedure call (RPC).

- To limit the number of records retrieved from the TOTAL database, the syntax is:

  ```
  IF READLIMIT EQ number
  ```

- To limit the number of records accepted by the server, the syntax is:

  ```
  IF RECORDLIMIT EQ number
  ```

You can add these conditions to the request, or they can be part of the security measures enforced by the DBA package.

Entry Segment Retrieval Using a Control Key

If the entry segment represents a TOTAL master file, and the master file control key is a field described in the Master File on the server, than an IF statement such as

```
IF control-key EQ value1 OR value2 OR ...
```

will cause the adapter to issue READM commands to retrieve the TOTAL records. The list of values can also be placed in a separate file and called with the statement

```
IF control-key EQ (ddname)
```

where:

ddname

Is the ddname of the file containing the list of values.

Entry Segment Retrieval Using an Internal Reference Point

The internal reference field is a field named by the Access File attribute REFERFIELD. It contains the internal reference points of the TOTAL records in the database.

If a request places a selection test on this field, the adapter retrieves the records directly with READD commands. The syntax is

```
WHERE field EQ value1 OR value2 OR ...
```

where:

field

Is the internal reference field.

value1, value2, ...

Are the internal reference points.

You can obtain the internal reference points by retrieving the REFERFIELD field in a previous request.

For example, if REFERFIELD=IRP was specified for a variable file in the Access File, then the request

```
SELECT DEPT, SALES FROM CAR WHERE IRP = (SELECT IRP FROM
CAR WHERE COUNTRY = 'JAPAN')
```

places the internal reference points in a sequential file, which can be used in the next request:

```
TABLE FILE CAR
PRINT DEPT AND SALES
IF IRP EQ (HOLD)
```

Using this method, a serial search of the variable file, using the condition WHERE COUNTRY EQ JAPAN, needs to be performed only once.

Entry Segment Retrieval Using FINDX

If a request places one or more record selection tests on fields, and these tests can be translated into a single FINDX argument parameter, the adapter issues the FINDX commands. If a choice of conditions exists, the adapter logic uses the single condition that is applicable to the greatest number of fields. For example, the screening condition

```
WHERE ZIP = ‘11004’ AND
SEX EQ M AND AGE EQ 40
```

would be translated into the FINDX argument

```
sex, age,.EQ.,M,40,END.
```

where:

`sex/age`

Denote the corresponding TOTAL field names.

`EQ`

Is the chosen condition, as applicable to two fields (while the LT condition is applicable to only one).

The test with the LT condition, WHERE ZIP < ‘110041’, is implemented by the server.

Entry Segment Retrieval Using RDNXT

If the entry segment fits none of the above conditions, the adapter implements a simple sequential read using the TOTAL RDNXT command.

Retrieval From Non-entry Segments

There are two ways the adapter retrieves records from TOTAL files represented by non-entry segments:

- If the segment is a TOTAL variable file whose parent is a master file, the adapter uses READV commands to follow the linkpath chain starting at the current instance of the parent master file.
- If the segment is a master file whose parent is a variable file, the adapter uses the READM command with the key value fetched from the parent variable file to retrieve an instance of the child master file.

CHAPTER 50

Using the Adapter for UniVerse

Topics:

The Adapter for UniVerse allows applications to access UniVerse data sources. The adapter converts data or application requests into native UniVerse statements and returns optimized answer sets to the requesting program.

Preparing the UniVerse Environment

For the Adapter for UniVerse, no environment variables need to be set prior to starting the server.

Configuring the Adapter for UniVerse

In this section:

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Example:

In order to connect to an UniVerse database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one UniVerse database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to UniVerse Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *SQL* group folder, then expand the adapter folder and click a connection. The adapter's Configuration page opens.
3. Provide valid values for all input fields.

Attribute	**Description**
Connection Name	User name for the connection information. If not specified, the data source name will be used as the connection name.
Datasource	UniVerse data source name.
User	User name by which you are known to UniVerse.
Password	Password associated with the user name.
Schema	UniVerse schema name including the entire path to it enclosed in single quotation marks.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

Note: User name and password are only checked when connecting to a remote server. When connecting to a local UniVerse server they are ignored.

4. Click *Configure.*

Syntax: How to Declare Connection Attributes Manually

```
ENGINE [SQLUV] SET CONNECTION_ATTRIBUTES [connection]
  [datasource]/userid,password [;dbname]
```

where:

`SQLUV`

Indicates the Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is a logical name used to identify this particular set of attributes.

`datasource`

Is the name of the UniVerse data source you wish to access.

`userid`

Is the primary authorization ID by which you are known to UniVerse.

`password`

Is the password associated with the primary authorization ID.

`dbname`

Also referred to as schema, is the name of the UniVerse database used for this connection. The database name, including path, must be inclosed in single quotation marks.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the UniVerse database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLUV SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

Overriding the Default Connection

How to:

Change the Default Connection

Example:

Selecting the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE [SQLUV] SET DEFAULT_CONNECTION [connection]
```

where:

`SQLUV`

Indicates the Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued: FOC1671, Command out of sequence.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued: FOC1671, Command out of sequence.

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the UniVerse database server named SAMPLENAME as the default UniVerse database server:

```
ENGINE SQLUV SET DEFAULT_CONNECTION SAMPLENAME
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE [SQLUV] SET AUTODISCONNECT ON {FIN|COMMAND}
```

where:

`SQLUV`

Indicates the Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing UniVerse Metadata

In this section:

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the UniVerse data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each UniVerse table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym Using the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane. The Select Adapter and Connection to Create Synonym pane opens.

3. Click a connection for the configured adapter. The right pane displays selection options that are appropriate for each adapter.

   Use Current Database. Select the *Use Current Database* check box to use the database defined for the chosen connection. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

   Select Database. If you wish to use a different database, leave the *Use Current Database* check box blank and choose a database from the drop-down list. **(Appears only for Informix, Microsoft SQL Server, and Sybase ASE.)**

Restrict Object Type to. Restrict candidates for synonym creation based on selected object type(s). Note that the object types vary depending on RDBMS support.

Filter by Name and Owner. Selecting this option adds the Owner and Table Name parameters to the screen:

- **Owner.** Type a string for filtering the owner IDs, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Table Name.** Type a string for filtering the table or view names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables or views whose names begin with the letters ABC; %ABC to select tables or views whose names end with the letters ABC; %ABC% to select tables or views whose names contain the letters ABC at the beginning, middle, or end.

4. Click *Select Candidates*. All tables that meet the specified criteria appear.
5. From the Select Application Directory drop-down list, select a directory. baseapp is the default value.
6. If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

   If all tables and views have unique names, leave prefix and suffix fields blank.
7. To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
8. To reflect the current cardinality (number of rows or tuples) in the table during metadata creation, select the *Cardinality* check box. The use of cardinality is for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

   Note: If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.
9. To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

   If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

10. From the SET CONVERSION LONGCHAR drop-down list, select: TEXT, ALPHA, or BLOB. TEXT is the default value.
11. Complete your table or view selection:

    To select all tables or views in the list, select the check box to the left of the *Default Synonym Name* column heading.

    To select specific tables or views, select the corresponding check boxes.
12. The Default Synonym Name column displays the name that will be assigned to the synonym. To assign a different name, replace the displayed value.
13. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```
14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Refresh Synonym	Regenerates the synonym. Use this option if the underlying object has been altered.

Recreate DBMS Table	Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)
Delete ALL Data	Deletes all existing data. You are asked to confirm this selection before the data is deleted.
Insert Sample Data	Inserts a specified number of sample records, populating all fields with counter values.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM app/synonym FOR table_view DBMS SQLUV [AT connection|AT '']
[NOCOLS]
END
```

where:

app

Is the 1- to 64-character application namespace where you want to create the synonym.

If your server is APP-enabled, you must use this application name.

If your server is not APP-enabled, you must not use this application name.

synonym

Is an alias for the data source (maximum 64 characters).

table_view

Is the name for the table or view. The value assigned to this parameter can include the name of the owner (also known as schema).

SQLUV

Indicates the Data Adapter for UniVerse.

`AT connection`

Is the name previously specified in a SET CONNECTION_ATTRIBUTES command. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

The connection specification varies for different adapters. It may be synonymous with DSN, TNS, source, location, etc.

This parameter is optional. If specified, the CONNECTION attribute is added to the Access File.

`AT ' '`

Adds CONNECTION=' ' in the Access File. This indicates a connection to the local database server.

`NOCOLS`

Specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Using CREATE SYNONYM

```
CREATE SYNONYM nf29004 FOR EDAQA.NF29004 DBMS SQLUV AT CONN_UV
END
```

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLUV ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4,     I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=NF29004 ,
TABLENAME=NF29004,
CONNECTION=CONN_UV,
KEYS=0
,$
```

Reference: Access File Keywords

Keyword	**Description**
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`TABLENAME`	Name of the table or view. This value can include a location or owner name as follows: `TABLENAME=[location.][owner.]tablename` **Note:** Location is valid only with DB2 CAF and specifies the subsystem location name.
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`* CONNECTION=' ' indicates access to the local UniVerse database server. Absence of the CONNECTION attribute indicates access to the default database server.
`KEYS`	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
`WRITE`	Specifies whether write operations are allowed against the table.

Data Type Support

The following chart provides information about the default mapping of UniVerse data types to server data types:

UniVerse Data Type	**Data Type**		**Remarks**
	USAGE	**ACTUAL**	
BIT (n) in (1...2032) BIT(1) - default value	N/A	N/A	Not supported.
CHAR (n) in (1...254) CHAR(1) - default value	A(1...254)	A(1...254)	
DATE	DATE	YYMD	
DECIMAL(p,s) DEC(9)-default value	DEC(1...9)	Dec(1...31)	The server supports up to Dec(31,9). If the number is larger than 31, it will not be displayed.
DOUBLE PRESICION	D8	D20.2	
FLOAT(p)	D8	D20.2	
INTEGER	I4	I11	
NATIONAL CHARACTER	A254	A254	
NUMERIC	P8	P11	
NATIONAL CHARACTER VARYING	A32767	A32767	
REAL	D8	D20.2	
SMALLINT	I4	I6	
TIME	A8	A8	
VARBIT	N/A	N/A	Not supported.
CHAR VARYING	A32767	A32767	

Changing the Precision and Scale of Numeric Columns

How to:

Override the Default Precision and Scale

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change these settings manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Override the Default Precision and Scale

```
ENGINE [SQLUV] SET CONVERSION RESET
ENGINE [SQLUV] SET CONVERSION format RESET
ENGINE [SQLUV] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [SQLUV] SET CONVERSION format [PRECISION MAX]
```

where:

`SQLUV`

Indicates the Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` which indicates that the command applies only to INTEGER columns.

`DECIMAL` which indicates that the command applies only to DECIMAL columns.

`FLOAT` which indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set scale to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE SQLUV SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SQLUV SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE SQLUV SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SQLUV SET CONVERSION RESET
```

Customizing the UniVerse Environment

In this section:

The Adapter for UniVerse provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Displaying Multivalued Columns in UniVerse

The OPENMODE NNF setting determines how tables with multivalued columns are treated. For the Adapter for UniVerse to function properly, OPENMODE must be set to NNF in edasprof.prf.

Syntax: How to Set OPENMODE NNF

```
ENGINE [SQLUV] SET OPENMODE NNF
```

where:

`SQLUV`

Indicates the Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

`OPENMODE NNF`

Allows the user application to see UniVerse tables the way they actually exist, including any multi-valued columns.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE [SQLUV] SET PASSRECS {ON|OFF}
```

where:

`SQLUV`

Indicates the Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Controlling Transactions

In order to perform Data Definition Language commands in UniVerse using the server, the TRANSACTION setting must be set to OFF.

Syntax: How to Issue the TRANSACTIONS Command

```
ENGINE SQLUV SET TRANSACTIONS {ON|OFF}
```

where:

`SQLUV`

Indicates the Adapter for UniVerse. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Turns transaction processing on. ON is the default value.

`OFF`

Turns transaction processing off.

Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL [SQLUV] SET {OPTIMIZATION|SQLJOIN} setting
```

where:

`SQLUV`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

OFF instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

ON instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some case. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

FOCUS passes join logic to the RDBMS only when the results will be the same as from a FOCUS-managed request. Misjoined unique segments, the multiplicative effect, and multiplied lines in PRINT and LIST based report requests invoke the data adapter-managed native join logic

SQL passes join logic to the RDBMS in all possible cases. The multiplicative effect does not disable optimization, even in cases involving aggregation (SUM, COUNT). Does not pass join logic to the RDBMS for tables residing on multiple subsystems and for tables residing on multiple DBMS platforms.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
sql SQLUV set optimization off
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT FROM
 "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
 FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
sql SQLUV set optimization on
 join emp_id in empinfo to all who in fundtran as j1
 table file empinfo
  sum ave.current_salary ed_hrs by who by last_name
  if department eq 'mis'
 end
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT) FROM
   "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2 WHERE (T2.EID =
   T1.EID) AND (T1.DPT = 'MIS') GROUP BY T2.EID,T1.LN ORDER BY
   T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing Virtual Fields

How to:

Optimize Requests Containing Virtual Fields

Example:

Using IF-THEN_ELSE Optimization Without Aggregation

Using IF-THEN_ELSE Optimization With Aggregation

Using IF-THEN_ELSE Optimization With a Condition That is Always False

Reference:

SQL Limitations on Optimization of DEFINE Expressions

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests Containing Virtual Fields

When you issue the SET OPTIFTHENELSE command, the adapter attempts to deliver the construct of a DML IF-THEN-ELSE DEFINE field to the RDBMS as an expression. The DEFINE field must be an object of a selection test or an aggregation request. The DEFINE definition may be specified in the DML request or in the Master File.

```
SQL [SQLUV] SET OPTIFTHENELSE {ON|OFF}
```

where:

`SQLUV`

Indicates the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Enables IF-THEN-ELSE optimization.

`OFF`

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set OPTIFTHENELSE OFF to disable the feature.

IF-THEN-ELSE optimization applies to SELECT statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLUV SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ '  ')
       AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
    SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1 WHERE
(((((T1."LN" = ' ') AND (T1."FN" = ' ')) AND (T1."DPT" =
'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLUV SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
       ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((T1."LN"
= 'SMITH'))) FOR FETCH ONLY;
```

```
TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
    SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1 WHERE (((NOT
(T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLUV SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
    SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR
FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

  Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting; consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

CHAPTER 51

Using the Adapter for VSAM

Topics:

The Adapter for VSAM allows applications to access VSAM data sources. The adapter converts application requests into native VSAM statements and returns optimized answer sets to the requesting application or it inserts the data from an application to the data source. Only ESDS and KSDS data sources are supported.

The Adapter for VSAM can access and change CICS-owned VSAM files if the Full-Function Server (FFS) is running on OS/390 or z/OS in the same LPAR as the target CICS region. The adapter, running outside CICS, employs EXCI to communicate with the remote component, the long running CICS transaction.

Note: Throughout this chapter, VSAM under CICS control is referred to as VSAM CICS.

Preparing the VSAM Environment

The Adapter for VSAM does not require setting any environment variables.

Preparing the Environment for VSAM CICS

The server starts a special process, CSCOM3, to control communications with all target CICS regions. This process can be stopped and started from the adapter's Web Console page.

One CSCOM3 process handles all configured EXCI communications to CICS regions. CSCOM3 sends the command to XMITRUE to initialize global storage for the server.

Note:

- CSCOM3 must be running prior to the execution of any request against VSAM CICS. CSCOM3 maintains open EXCI pipes to CICS regions until it is stopped. Having an open pipe may prevent CICS from being recycled.
- When CICS is being recycled, the CSCOM3 service must be stopped. In the event that CICS is forced to shutdown, CSCOM3 must also be stopped, and restarted when CICS is available for use. When CSCOM3 is stopped, it sends a request to XMITRUE to release all global storage. This may have an adverse affect on in-flight requests.

Reference: VSAM CICS Components

One transaction and three programs must be defined in CICS:

- **XMIV.** Is a transaction pointing to the program XMI.
- **XMI.** Is the main program. It is responsible for all physical data I/O.
- **XMITRUE.** Is a dual purpose program. When called by CSCOM3, it sets up global storage areas in CICS for each server odin node combination.
- **XMIABEND.** Is an error handling program. It gets control when XMI or XMITRUE abends. It is responsible for capturing trace and log information, as well as global storage cleanup.

Example: Sample CICS Installation

```
//jobname JOB (accid),'CICS 3.3.0 INSTALL'
//     MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=userid,
//   PERFORM=3
//*
//DBCRDO   EXEC PGM=DFHCSDUP,REGION=256M
//STEPLIB  DD DSN=CICSTS.V1R3M0.SDFHLOAD,DISP=SHR
//DFHCSD2  DD DSN=userid.CICS410.DFHCSD,DISP=SHR
//DFHCSD   DD DSN=userid.CICSTS13.DFHCSD,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSIN    DD  *

  DEFINE PROGRAM(XMI) GROUP(XMI)
  DESCRIPTION(XMI MAIN MODULE)
         LANGUAGE(C) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL) USELPACOPY(NO)
         STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY) EXECKEY(CICS)
         EXECUTIONSET(FULLAPI)

  DEFINE PROGRAM(XMIABEND) GROUP(XMI)
  DESCRIPTION(XMI ABEND MODULE)
         LANGUAGE(C) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL) USELPACOPY(NO)
         STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY) EXECKEY(USER)
         EXECUTIONSET(FULLAPI)

  DEFINE PROGRAM(XMITRUE) GROUP(XMI)
  DESCRIPTION(XMITRUE GLOBAL STORAGE ANCHOR)
         LANGUAGE(C) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL) USELPACOPY(NO)
         STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY) EXECKEY(CICS)
         EXECUTIONSET(FULLAPI)

  DEFINE TRANSACTION(XMIV) GROUP(XMI)
  DESCRIPTION(XMI MAIN TRANSACTION)
         PROGRAM(XMI) TWASIZE(0) PROFILE(DFHCICST) STATUS(ENABLED
         TASKDATALOC(ANY) TASKDATAKEY(USER) STORAGECLEAR(NO)
         RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
         PRIORITY(1) TRANCLASS(DFHTCL00) DTIMOUT(NO) INDOUBT(BACKOUT
         RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(NO) TRACE(YES)
         CONFDATA(NO) RESSEC(NO) CMDSEC(NO)
/*
//
```

Configuring the Adapter for VSAM or VSAM CICS

In this section:

Configuration requirements vary for the Adapters for VSAM and VSAM CICS:

- **For VSAM,** if you configure the adapter manually, you must specify the location of the data files you want to access in a supported server profile. If, however, you configure the adapter from the Web Console, you simply click the *Configure* button. You will specify the location of the data files to access when you create a synonym.
- **For VSAM CICS**, you must specify connection and authentication information for each of the connections you want to establish. You can declare connection attributes from the Web Console (see *How to Declare Connection Attributes for VSAM CICS From the Web Console* on page 51-5), or manually in the ODIN configuration file.

Procedure: How to Configure the Adapter for VSAM From the Web Console

To configure the adapter from the Web Console,

1. Start the Web Console and select *Data Adapters* from the left pane.
2. Expand the *Add* folder, expand the *Other DBMS* group folder, then expand the *VSAM* folder and click a connection. The Add VSAM to Configuration pane opens.
3. No input is required. Simply click the *Configure* button.

Syntax: How to Specify the Location of VSAM Files Manually

VSAM data sets may be allocated with JCL

```
//QAVSM DD DISP=SHR, DSN=cluster_name
```

or with DYNAM

```
DYNAM ALLOC FILE QAVSM DA cluster_name SHR REUSE
```

or in a Master File as

```
DATASET='dataset_name,' $
```

where:

`cluster_name`

Is the name of the VSAM cluster.

`dataset_name`

Is the name of the dataset on MVS.

Note that if you are using an alternate index (a separate, additional index structure that enables you to access records in a VSAM data source based on a key other than the data source's primary key), the alternate index must be related to the base cluster it describes by a path, which is stored in a separate data source. Therefore, in addition to defining the location of the base cluster, you must define the path clusters either in IRUNJCL or using a DYNAM command. For details about logical processing with alternate indexes, see *Reporting From Files With Alternate Indexes* on page 51-60.

Note: The Adapter for VSAM supports full read/write access.

Example: Defining Path Clusters in JCL

```
//VSAMFL1  DD DISP=OLD,DSN=EDABXV.VSAM.AXINDEX1.PATH
//VSAMFL2  DD DISP=OLD,DSN=EDABXV.VSAM.AXINDEX2.PATH
```

Procedure: How to Declare Connection Attributes for VSAM CICS From the Web Console

To configure the adapter for VSAM CICS from the Web Console:

1. Start the Web Console and select *Data Adapters* from the left pane.
2. Expand the Add folder, expand the Other DBMS group folder, then expand the VSAM CICS folder and click a connection. The Add VSAM CICS to Configuration pane opens.
3. Enter the following parameters:

Connection name	Is a logical name used to identify a specific set of connection attributes.
Proxy	Defines the name of the supplied proxy (or user written) program that converts the Full Function Server calling syntax to the syntax that is valid for the CICS program to be executed. **Note:** This parameter is optional and should only be used at the direction of local support personnel. Leave the field blank for normal operation.

Mirror	Defines the IBM-supplied mirror transaction CPMI that the CICS Adapter calls by default. If this transaction cannot be used, another transaction name can be created to point to the IBM-supplied mirror program DFHMIRS. This alternate mirror transaction name can be supplied in this field.
Partner LU Name (Application ID)	Defines the application ID of the CICS region.
Local LU Name	Defines the LU name to be used. This value is taken from an LU entry in the major node. This value is also specified in the CICS connection/sessions definition.
Trace DSN	Trace dataset name.
Log DSN	Log dataset name.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Configure*.

For related information about VSAM CICS logs, see *Logging VSAM CICS Processing* on page 51-7.

Declaring Connection Attributes for VSAM CICS Manually

To establish communication between a Full-Function Server and CICS, you can define communication parameters in the ODIN configuration file. The ODIN node is created when you add a new connection for the Adapter the VSAM CICS.

Example: ODIN Configuration File With CICS VSAM Node

```
NODE = VSMCICS
BEGIN
  PROTOCOL = EXCI
  PROXY = XMITRUE
  MIRROR = EXCI
  CLASS = CICSCLIENT
  PARTNER_LU_NAME = EDBGM006
  LOCAL_LU_NAME = T29DPB50
  TRACEDSN = userid.CICSVSAM.TRACE
  LOGDSN = userid.CICSVSAM.LOG
END
```

Logging VSAM CICS Processing

Logging is always on. All log information is sent to the CICS log CSSL. In most cases, this shows up under JES as MSGUSR.

Log information is written to the specified log data set in addition to the standard CICS log. However, the two logs may not be completely synchronized. Some errors may not permit writing to the specified Log file.

When CSCOM3 receives a STOP request, it transfers all relevant CICS log records to the local HFS cscom3.log file. The CICS Log file is cleared as part of CSCOM3 START request processing.

The client component of the Adapter for VSAM CICS is executed as an agent process under TSCOM3. The server component of the Adapter for VSAM CICS is executed in the CICS region as part of the XMI program associated with the XMIV with transaction. The client component is responsible for the logical processing of the client's request.

The server component executes physical I/O operations with the CICS controlled VSAM data sets. The server component also dynamically defines in the CICS region (if not pre-defined by the user), the dataset whose name is passed to the client.

Client component trace records are stored in the TSCOM3-owned HFS trace file. Server component trace records are forwarded to the transaction-owned Temporary Storage Queue. All records from the queue are sent to the server and appended to the HFS trace file upon completion of the request.

During abend processing, the information stored in the TS-Queue is copied to the specified trace file. The file is then transferred to the server as part of CSCOM3 STOP request processing. Tracing can be turned on and off from the server Web Console.

Communication between the server and CICS is handled by the standard communication layer on the server-side and by the mirror transaction (EXCI) on the CICS side. All information is passed in the COMAREA buffer in packed format.

Example: Defining Trace and Log Files

Both trace and log files must be defined and available to all users prior to server startup.

Define these files as follows:

```
//jobname JOB (accid),'CICS 3.3.0 INSTALL'
//      MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=userid,
//      REGION=256M
//*

//SYSPRINT DD SYSOUT=*
//SYSIN    DD *

    DEL userid.CICSVSAM.TRACE
    DEL userid.CICSVSAM.LOG

    DEF CL ( NAME( userid.CICSVSAM.TRACE ) -
              VOLUMES(volume) -
              CYLINDERS(1 1) -
              RECSZ(200 200) -
              CISZ(4096) -
              INDEXED  -
              UNIQUE  -

              SHR(2 3))

    DEF CL ( NAME( userid.CICSVSAM.LOG ) -
              VOLUMES(volume)

              RECSZ(200 200) -
              CISZ(4096)
              UNIQUE  -
              KEYS(8 0) -
              SHR(2 3))
/*
//
```

Managing VSAM Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the VSAM data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each VSAM file or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File that represents the server's metadata.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The Select Synonym Candidates pane (Step 1 of 2) opens.

4. Under File System Selection, choose one of the following options from the drop-down list:

   - *Fully qualified PDS name* to indicate a partitioned dataset on MVS.

     In the input boxes provided, type a PDS name preceded by // and a Member name containing the location of the COBOL FD source. If you wish, you can filter the member name using a wildcard character (%).

     or

   - *Absolute HFS directory pathname* to indicate a hierarchical file structure on USS.

     In the input boxes provided, type a Directory name to specify the HFS location that contains the COBOL FD and a File name and File extension. If you wish, you can filter the file and extension using a wildcard character (%).

5. Click *Submit*. The Create Synonym pane (Step 2 of 2) opens.

   If selected, the filtered COBOL definition is displayed at the top of the pane.

6. Enter the following parameters, as required:

Parameter	**Description**
Select Application	Select an application directory. baseapp is the default value.
Prefix/Suffix	If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all tables and views have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

7. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

   When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

8. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

9. Optionally, select *Customize options* to customize how the COBOL FD is translated. For details about these options, see *Customization Options for COBOL File Descriptions* on page 51-12. If you do not select the check box, default translation settings are applied.

10. Complete your selection:

    - To select all synonyms in the list, select the check box to the left of the *Default Synonym Name* column heading.

    - Enter a cluster name to associate it with a particular metadata description. The cluster name is embedded in the Master File. If you do not enter the cluster name during the synonym creation process, you will have to add it dynamically at run time.

    - To select specific synonyms, select the corresponding check boxes.

11. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

12. Click *Create Synonym*.

    Synonyms are created and added under the specified application directory.

    A status window displays the message:

    ```
    All Synonyms Created Successfully
    ```

13. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: Customization Options for COBOL File Descriptions

You can customize how a COBOL FD is translated by selecting *Customize options* in the Synonym creation pane. The following options are added to the right pane:

Parameter	**Definition**
On Error	Choose *Continue* to continue generating the Master File when an error occurs. Choose *Abort* to stop generating the Master File when an error occurs. Choose *Comment* to produce a commented Master File when an error occurs. Continue is the default value.

Parameter	Definition
Hyphens as	Choose *No* to remove all hyphens in the COBOL name from the Master File field names. Choose *Yes* to replace all hyphens in the COBOL name with the underscore character. *Yes* is the default value.
Redefines	You may treat COBOL REDEFINE fields in one of three ways: Choose *Segments* to describe REDEFINE fields as segments in the Master File. Segments is the default value. Choose *Comments* to describe REDEFINE fields as comments in the Master File. Choose *None* to exclude REDEFINE fields altogether.
Occurs as	Choose *Segments* to describe OCCURS structures as segments, or *Field* otherwise. Segments is the default value.
Alignment	Choose *Yes* to insert slack bytes into a record to ensure alignment of numeric fields. Choose *No* to generate Master Files without alignment of slack bytes. No is the default value.
Number of Hyphens to skip	FD Translator removes characters from the left, up to and including the Nth hyphen (depending on the value of N chosen in the menu). A value of *0* will retain the entire COBOL name. *All* means all prefixes will be removed. *0* is the default value.
Order fields	Choose *Yes* to generate Order fields in a Master File. Choose *No* to generate a Master File without Order fields. *No* is the default value.

Parameter	Definition
Level 88 as	Choose *Comment* to include COBOL Level 88 fields as comments in the Master Files. Choose *Skip* to exclude level 88 fields. *Skip* is the default value.
Zoned Numeric Fields	Sets how zoned numeric values will be stored.
Numeric Fields Edit Options	
Zeroes	Choose *Suppress* to suppress printing of the digit zero for a field whose value is zero. Choose *Display* to display leading zeroes, for example, 00124. Choose *None* for no formatting.
Negative value	Choose *Bracket* to bracket negative values, for example, (1234). Choose *Credit* to credit negative values, for example, 1234 CR. Choose *None* for no formatting.
Dollar Sign	Choose *Floating* to display a floating dollar sign and commas, for example, $1,1234. Choose *Fixed* to display a fixed dollar sign and commas, for example, $ 1,1234. Choose *None* for no formatting.
Separate Thousands	Choose *Comma* to include commas where appropriate. Choose *None* for no formatting.

For additional information about customization options, see Appendix D, *Translating COBOL File Descriptions*.

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM baseapp/synonym FOR cobol_fd DBMS VSAM
DATABASE 'vsam_dataset_name'
[CHECKNAMES][UNIQUENAMES]
END
```

where:

synonym

Is the name to be assigned the resulting synonym.

cobol_fd

Is the HFS location or MVS PDS, including the member name containing the COBOL FD source.

vsam_dataset_name

Is the MVS name of the VSAM cluster.

CHECKNAMES

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ',';'$'. No checking is performed for names.

UNIQUENAMES

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '| ', '&', ' (', ') ', ' <', '> ', '' ', ' =', ''''

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Example: Sample COBOL FD

```
01 COUNTRY-REC.
   02 G1.
      03  COUNTRY_COD1    PIC X(5).
      03  FILLER          PIC X(3).
    02  COUNTRY_NAM1    PIC X(15).
    02  FILLER          PIC X(1).
```

Example: Using CREATE SYNONYM for a VSAM Data Source

```
CREATE SYNONYM baseapp/VSAM2901
FOR /pgm/edaport/R729999B/tscq/nf2901.cbl
DBMS VSAM
DATABASE 'EDAQA.SQLTRANS.NF29001.VSAM'
END
```

Synonym:

```
FILENAME=VSAM2901, SUFFIX=VSAM    ,
 DATASET=EDAQA.SQLTRANS.NF29001.VSAM, $
  SEGMENT=SEG1, SEGTYPE=S0, $
$  GROUP=COUNTRYREC, USAGE=A24, ACTUAL=A24, $
   GROUP=G1, ALIAS=KEY, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=COUNTRY_COD1, USAGE=A5, ACTUAL=A5, $
    FIELDNAME=FILLER, USAGE=A3, ACTUAL=A3, $
    FIELDNAME=COUNTRY_NAM1, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=FILLER, USAGE=A1, ACTUAL=A1, $
```

Standard Master File Attributes for a VSAM Data Source

In this section:

Most standard Master File attributes are used with VSAM data sources in the standard way.

- **SUFFIX.** The SUFFIX attribute in the file declaration for these data sources has the value VSAM.
- **SEGNAME.** The SEGNAME attribute of the first or root segment in a Master File for a VSAM 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.
- **GROUP.** The keys of a VSAM data source are defined in the segment declarations as GROUPs consisting of one or more fields.

Describing a Group Field

How to:

Describe a VSAM Group Field

Example:

Describing a VSAM Group Field

Describing a VSAM Group Field With Multiple Formats

Accessing a Group Field With Multiple Formats

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 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 appears:

```
(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 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.
- The MISSING attribute is not supported on the group field, but is supported on the individual fields comprising the group.

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, can 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 is a VSAM data structure. The Master File looks 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   ,$
```

Example: Describing a VSAM Group Field With Multiple Formats

```
GROUP = A, ALIAS = KEY, USAGE = A14, ACTUAL = A8   ,$
 FIELDNAME = F1, ALIAS = F1, USAGE = P6, ACTUAL=P2 ,$
 FIELDNAME = F2, ALIAS = F2, USAGE = I9, ACTUAL=I4 ,$
 FIELDNAME = F3, ALIAS = F3, USAGE = A2, ACTUAL=A2 ,$
```

The lengths of the ACTUAL attributes for subordinate fields F1, F2, and F3 total 8, which is the length of the ACTUAL attribute of the group key. The display lengths of the USAGE attributes for the subordinate fields total 17. However, the length of the group key USAGE attribute is found by adding their internal storage lengths as specified by their field types: 8 for USAGE=P6, 4 for USAGE=I9, and 2 for USAGE=A2, for a total of 14.

Example: Accessing a Group Field With Multiple Formats

When you use a group field with multiple formats in a query, you must account for each position in the group, including trailing blanks or leading zeros. The following example illustrates how to access a group field with multiple formats in a query:

```
GROUP = GRPB, ALIAS = KEY, USAGE = A8, ACTUAL = A8 ,$
 FIELDNAME = FIELD1, ALIAS = F1, USAGE = A2, ACTUAL = A2 ,$
 FIELDNAME = FIELD2, ALIAS = F2, USAGE = I8, ACTUAL = I4 ,$
 FIELDNAME = FIELD3, ALIAS = F3, USAGE = A2, ACTUAL = A2 ,$
```

The values in fields F1 and F3 may include some trailing blanks, and the values in field F2 may include some leading zeros. When using the group in a query, you must account for each position. Because FIELD2 is a numeric field, you cannot specify the IF criteria as follows:

```
IF GRPB EQ 'A 0334BB'
```

You can eliminate this error by using a slash (/) to separate the components of the group key:

```
IF GRPB EQ 'A/334/BB'
```

Note: Blanks and leading zeros are assumed where needed to fill out the key.

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.

Using the OCCURS Attribute

How to:

Specify a Repeating Field

Example:

Using the OCCURS Attribute

The OCCURS attribute is an optional segment attribute used to describe records containing repeating fields or groups of fields. 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. 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 for *occurstype* 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 (for example, if the field lengths for the segment add up to 40, and 120 characters are read in, it means there are three occurrences).

Place the OCCURS attribute in your segment declaration after the PARENT attribute.

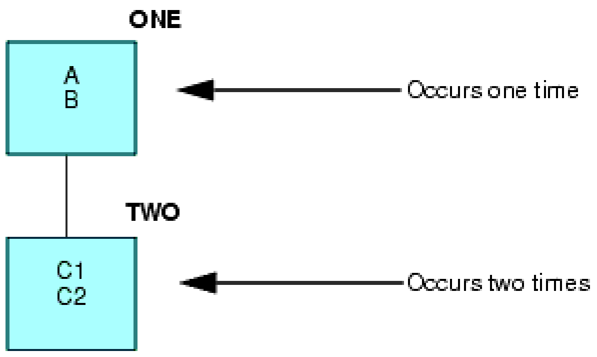
When different types of records are combined in one data source, each record type can contain only one segment defined as OCCURS=VARIABLE. It may have OCCURS descendants (if it contains a nested group), but it may not be followed by any other segment with the same parent—that is, there can be no other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



Describe this data source as follows:

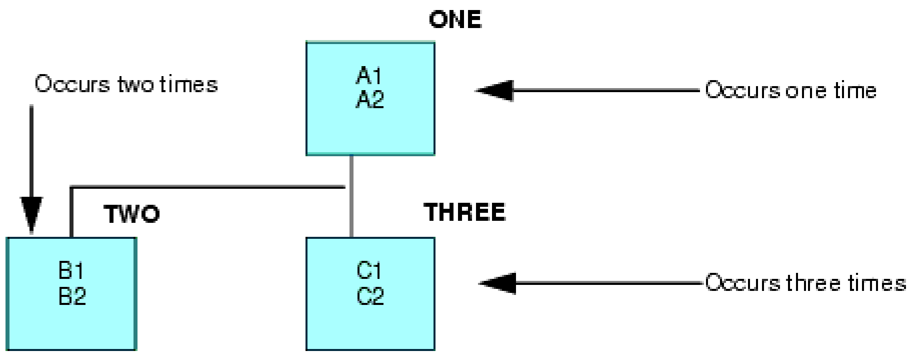
```
FILENAME = EXAMPLE1, SUFFIX = FIX, $
 SEGNAME = ONE, SEGTYPE=S0, $
  FIELDNAME = A,  ALIAS=, USAGE = A2, ACTUAL = A2, $
  FIELDNAME = B,  ALIAS=, USAGE = A1, ACTUAL = A1, $
 SEGNAME = TWO, PARENT = ONE, OCCURS = 2, SEGTYPE=S0, $
  FIELDNAME = C1, ALIAS=, USAGE = I4, ACTUAL = I2, $
  FIELDNAME = C2, ALIAS=, USAGE = I4, ACTUAL = I2, $
```

Describing 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:



Describing a Nested Set of Repeating Fields

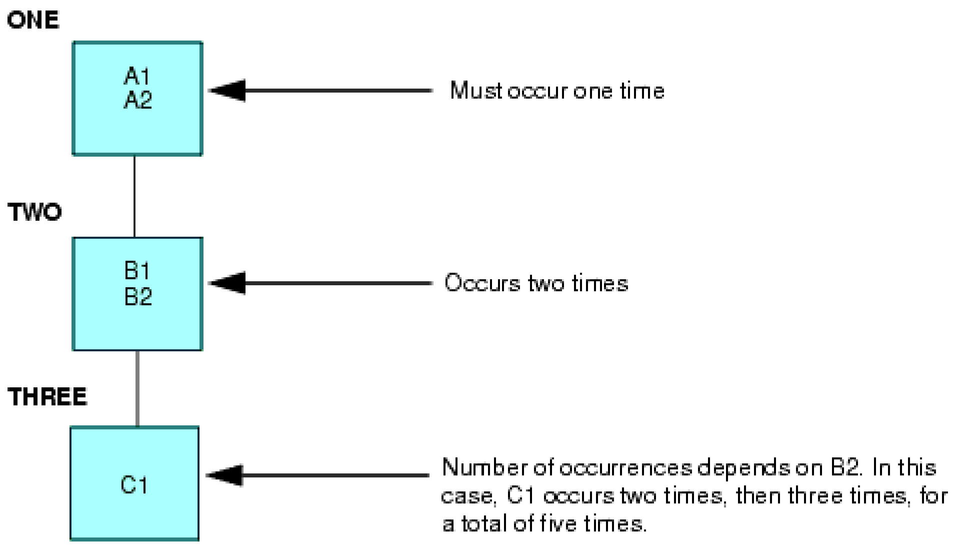
Example:

Describing Parallel and Nested Repeating Fields

Nested sets of repeating fields are those whose occurrence depends on one another in some way. 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. There is not a set of occurrences of C1 which is not 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:


ONE
A1
A2
Must occur one time
TWO
B1
B2
Occurs two times
THREE
C1
Number of occurrences depends on B2. In this case, C1 occurs two times, then three times, for a total of five times.

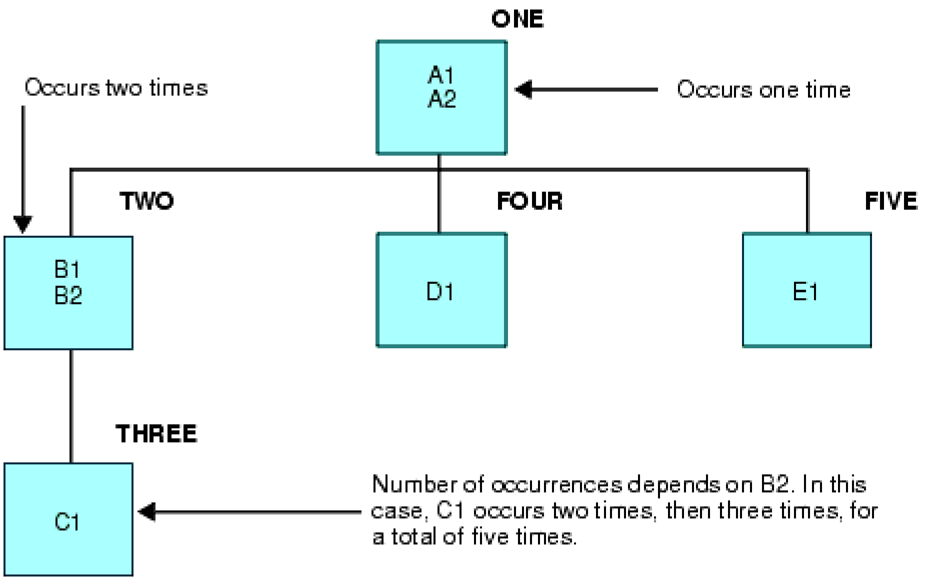
Since field C1 repeats with relation to fields B1 and B2, which repeat in relation to fields A1 and A2, field C1 is described as a separate, descendant segment of Segment TWO, which is in turn a descendant of Segment ONE.

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:


ONE
A1
A2
Occurs one time
Occurs two times
TWO
FOUR
FIVE
B1
B2
D1
E1
THREE
C1
Number of occurrences depends on B2. In this case, C1 occurs two times, then three times, for a total of five times.

Describe this data source as follows. 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 (all those to the right of the first occurrence of E1) are read 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 is the rightmost segment. Note that there can be only one segment defined as OCCURS=VARIABLE for each record type.

Using 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 is 32 characters.

You can also use the POSITION attribute to re-describe fields with SEGTYPE=U. See *Redefining a Field in a VSAM Data Source* on page 51-31.

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 of the multiple field occurrences.

Example: Specifying the Position of a Repeating Field

Consider the following data structure:

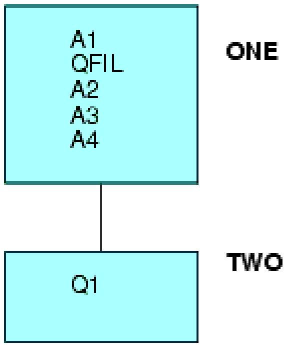
A1	Q1	Q1	Q1	Q1	A2	A3	A4

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 multiply occurring descendant 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.

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 4095, 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 4095 bytes.

Notice that this structure works 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 does 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.

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 I*n*, 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 VSAM Data Source

How to:

Redefine a Field

Example:

Redefining a VSAM Structure

Reference:

Special Considerations for Redefining a Field

Redefining record fields in non-FOCUS data sources is supported. This enables you to describe a field 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.

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. Using the unique segment with redefined fields helps avoid problems with multipath reporting.

A one-to-one relationship forms 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.
- For non-alphanumeric fields, you must know your data. Attempts to print numeric fields that contain alphanumeric data 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.

Syntax: How to Define the Maximum Record Length

```
SET MAXLRECL = nnnnn
```

where:

nnnnn

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 MAXLRECL command.

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

Describing Multiple Record Types

In this section:

VSAM, 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 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 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 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. Do this by creating a dummy root segment for the records.

Key-sequenced VSAM 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 may share some other identifying piece of information, such as the PUBNO in the 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 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 *Describing VSAM Positionally Related Records* on page 51-39, 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 can contain a different two-digit code for each unique record type. Describe this identifying field with the field name RECTYPE.

Another technique for redefining the parts of records is to use the MAPFIELD and MAPVALUE attributes described in *Describing a Repeating Group Using MAPFIELD* on page 51-52.

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 is ignored. The syntax to describe the RECTYPE field is

```
FIELDNAME = RECTYPE, ALIAS = value, USAGE = format, ACTUAL = format,
ACCEPT = {list|range} ,$
```

where:

value

Is the record type in alphanumeric format, if an ACCEPT list is not specified. If there is an ACCEPT list, this can be any value.

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:

A*n* (where *n* is 1-4095) indicates character data, including letters, digits, and other characters.

I*n* 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).

The USAGE format can be I1 through I9, depending on the magnitude of the ACTUAL format.

P*n* (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. 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 P*m.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.

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. Separate each item in the list with 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: Specifying the RECTYPE Field

The following field description is of a one-byte packed RECTYPE field containing the value 1:

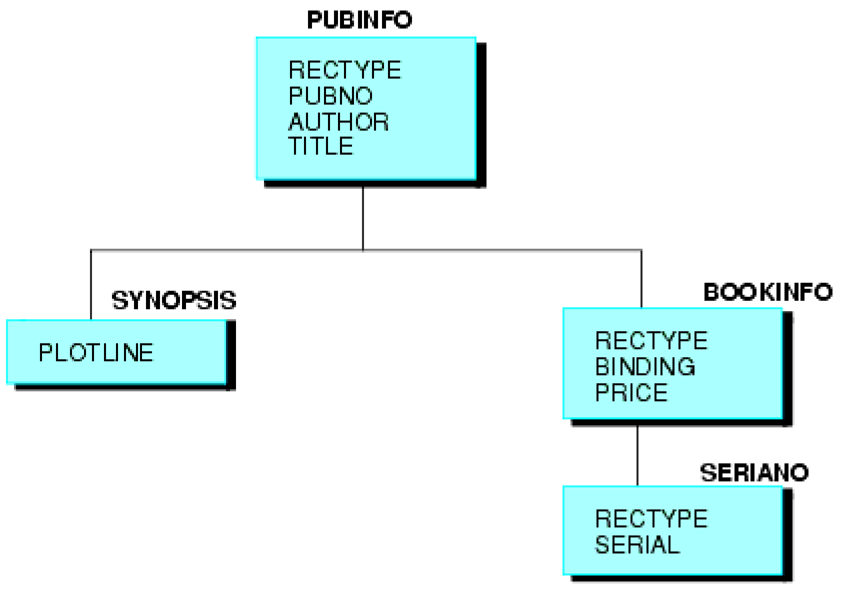
```
FIELD = RECTYPE, ALIAS = 1, USAGE = P1, ACTUAL = P1, $
```

The following field description is of 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:



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. All of this information is 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 is 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 in the diagram to the right of the OCCURS=VARIABLE segment, OCCURS=VARIABLE is the rightmost 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 it 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 has its own identifying serial number, which is 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.

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 OCCURS, contains a field named RECTYPE and that the ALIAS for the field contains a unique value for each segment (P, B, and S). If there is a record in this data source with a RECTYPE other than P, B, or S, the record is ignored. The RECTYPE field must fall in the same physical location in each record.

Ordering of Records in the Data Source

Physical order determines parent/child relationships in sequential records. Every parent record does not need descendants. Specify how you want data in missing segment instances handled in your reports by using the SET command to change the ALL parameter.

In the example in *Describing Positionally Related Records* on page 51-37, 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 appears 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

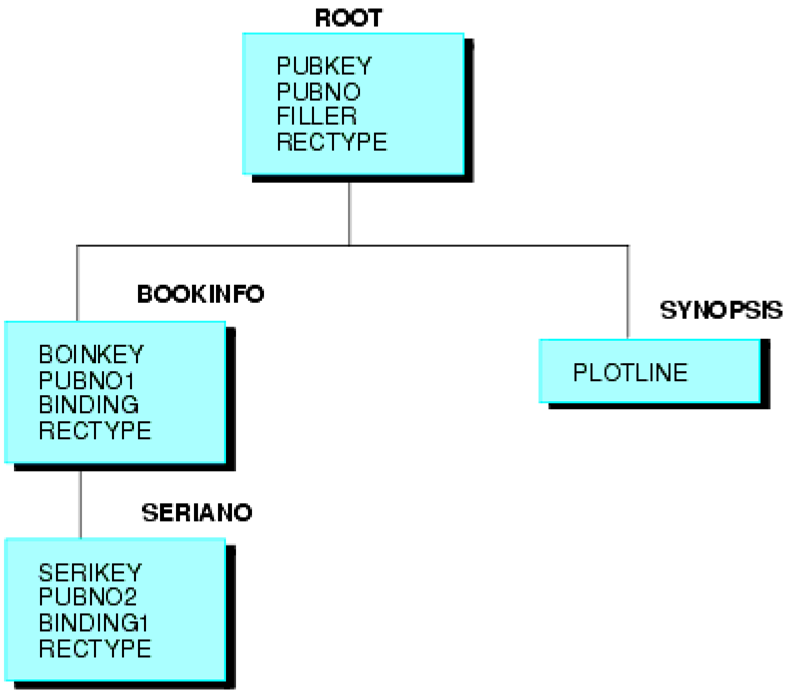
Consider the following VSAM data source that contains three types of records. The ROOT records have a key that consists of the publisher's number, PUBNO. The BOOKINFO segment has a key that consists of that same publisher's number, plus a hard- or soft-cover indicator, BINDING. The SERIANO segment key 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.

Note: Each segment's key describes as much of the true key as needed to find the next instance of that segment.

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 may 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, retrieval can occur quickly, and the processing continues. To further speed retrieval, add search criteria based on the BINDING field, which is also part of the key.

Describing Unrelated Records

Some VSAM and data sources do not have records that are related to one another. That is, the VSAM key of one record type is independent of the keys of other record types. To describe data sources with unrelated records, define a dummy root segment for the record types. The following rules apply to the dummy root segment:

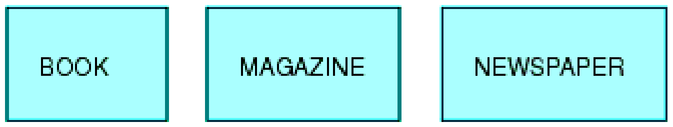
- The name of the root segment must be DUMMY.
- It must have only one field with a blank name and alias.
- The USAGE and ACTUAL attributes must both be A1.

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 key. If the data source does not have a key, the group should not be described. RECTYPEs may be anywhere in the record.

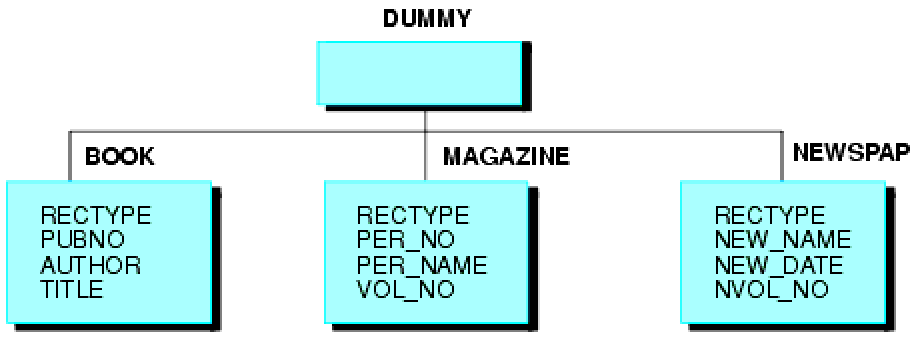
Example: Describing Unrelated Records Using a Dummy Root Segment

The library data source has three types of records: book information, magazine information, and newspaper information. Since these three record types have nothing in common, they cannot be described as parent records followed by detail records.

The data source can look like this:



A structure such as the following can 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 the 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

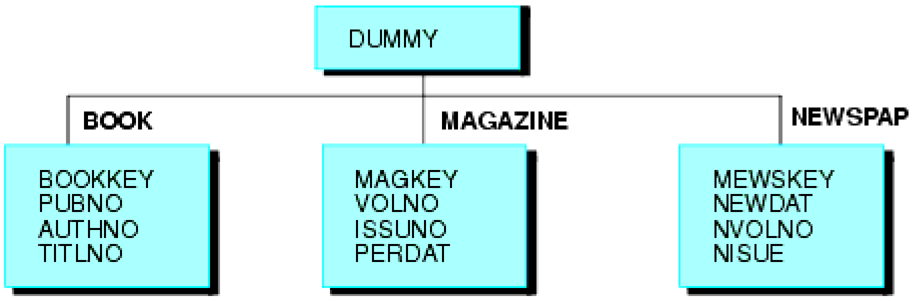
  Note the difference between the use of the RECTYPE here and its use when the records are positionally related. In this case, the codes are unrelated and the database designer has chosen to accumulate the records by type first (all the book information together, all the magazine information together, 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:



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 VSAM data source has multiple record types that share the same layout, you can specify a single generalized segment that describes all record types that have 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 using a generalized segment, you identify RECTYPE values using the ACCEPT attribute. You can assign any value 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 a VSAM Master File, 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. A unique set of field names must be provided for record type M and record type I, although they have the same layout.

The generalized RECTYPE capability enables you to code just one set of field names that applies to the record layout for both record type M and 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,  $
```

Combining Multiply-Occurring Fields and Multiple Record Types

In this section:

You can have two types of descendant segments in a single fixed-format VSAM, 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.

01	T1	N1	B1	B2	C1	C1	C1	D1	D1	D1	D1	D1	D1	D1	B1	B2	C1	C1	D1	D1	D1	D1	D1	D1	D1

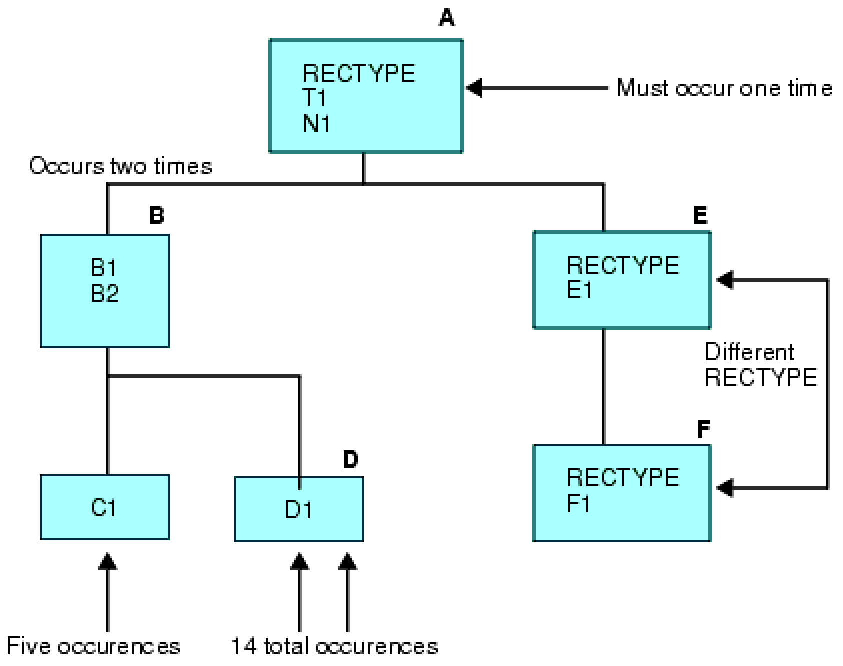
02	E1

03	F1

The 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 that are related through their record types. The record type attribute consists of certain prescribed values, and is stored in a fixed location in the records. 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) or another 03.
- 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 rightmost 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.

Describing a VSAM Repeating Group With RECTYPEs

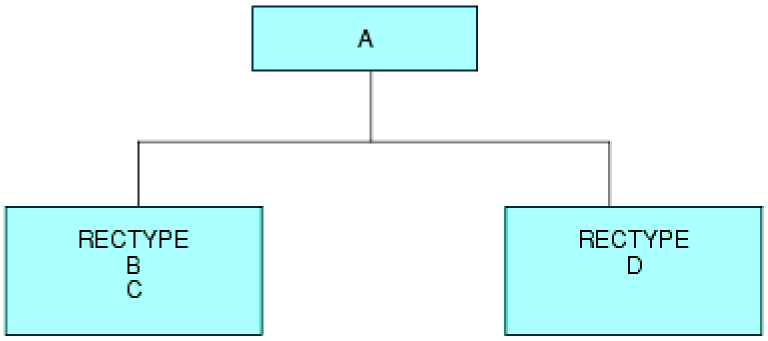
Example:

Describing 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 the OCCURS segment. This enables you to describe complex data sources, 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

How to:

Describe a Repeating Group With MAPFIELD

Use MAPFIELD for a Descendant Repeating Segment in a Repeating Group

Example:

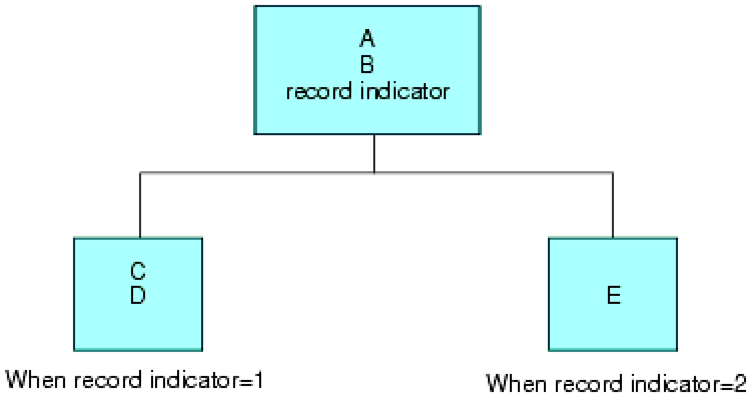
Using MAPFIELD and MAPVALUE

In another 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 appears like this:

```
A B      record indicator (1)    C D    C D     C D
A B      record indicator (2)    E E
```

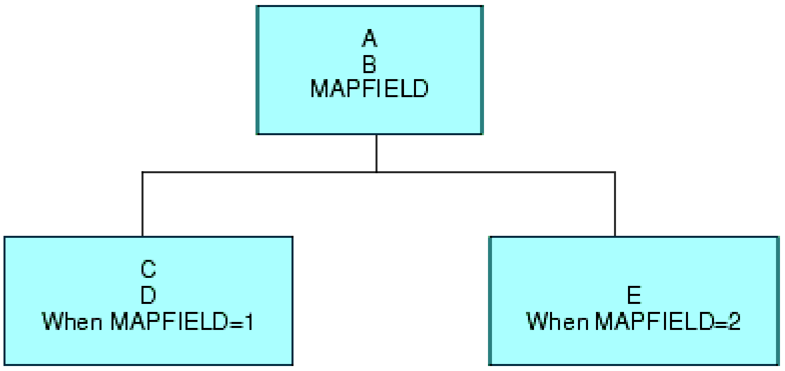
The first record contains header information, values for A and B, followed by an OCCURS segment of C and D that was identified by its preceding record indicator. The second record has a different record indicator and contains a different repeating group, this time for E.

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 as RECTYPE, but since each OCCURS segments has its own value 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 is the record indicator; it may have any name.

Syntax: How to Describe a Repeating Group With MAPFIELD

```
FIELD = name, ALIAS = MAPFIELD, USAGE = format, ACTUAL = format,$
```

where:

`name`

Is the name you choose to provide for this field.

`ALIAS`

MAPFIELD is assigned as the alias of the field that is 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 value, if an ACCEPT list is not specified. If there is an ACCEPT list, this can be any value.

`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 remapped. This means that the segment definition cannot contain POSITION=*fieldname*.

MAPFIELD and MAPVALUE may be used with SUFFIX=FIX and SUFFIX=VSAM data sources.

Establishing 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 enable 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, you may improve job throughput. 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

```
ENGINE VSAM SET BUFND {n|8}
ENGINE VSAM SET BUFNI {n|1}
```

where:

n

Is the number of data or index buffers. BUFND=8 and BUFNI=1 (eight data buffers and one index buffer) are the default values.

To determine how many buffers are in effect at any time, issue the query:

```
ENGINE VSAM SET ?
```

Using a VSAM Alternate Index

Example:

Describing a VSAM Alternate Index

Using IDCAMS

VSAM key-sequenced data sources support the use of alternate key indexes (keys). 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 components are referred to as the base cluster.

An alternate index is a separate, additional index structure that enables 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 occasionally need to retrieve records 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 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 identify the indexed view explicitly 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 a WHERE or IF 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 see in the Master File in *Describing a VSAM Alternate Index* on page 51-58, the LNAME field is defined as an alternate index field. The records in the data source are retrieved according to their last names, and certain IF screens on the field LNAME result in direct reads. Note that if the alternate index field name is omitted, the primary key (if there is any) is used for a sequential or a direct read, and the alternate indexes are 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 ALIAS=KEY must be used to describe the primary key.

Only one record type can be referred to in the request when using alternate indexes, but there is no restriction on the number of OCCURS segments.

Note that the path name in the allocation is 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:

```
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 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.

VSAM Record Selection Efficiencies

The most efficient way to retrieve selected records from a VSAM KSDS data source is by applying an IF screening test against the primary key. This results in a direct reading of the data using the data source's index. Only those records that you request are retrieved from the file. The alternative method of retrieval, the sequential read, forces the adapter to retrieve all the records into storage.

Selection criteria that are based on the entire primary key, or on a subset of the primary key, cause direct reads using the index. A partial key is any contiguous part of the primary key beginning with the first byte.

IF selection tests performed against virtual fields can take advantage of these efficiencies as well, if the full or partial key is embedded in the virtual field.

The EQ and IS relations realize the greatest performance improvement over sequential reads. When testing on a partial key, equality logic is used to retrieve only the first segment instance of the screening value. To retrieve subsequent instances, NEXT logic is used.

Screening relations GE, FROM, FROM-TO, GT, EXCEEDS, IS-MORE-THAN, and NOT-FROM-TO all obtain some benefit from direct reads. The following example uses the index to find the record containing primary key value 66:

```
IF keyfield GE 66
```

It then continues to retrieve records by sequential processing, because VSAM stores records in ascending key sequence. The direct read is not attempted when the IF screening conditions NE, IS-NOT, CONTAINS, OMITS, LT, IS-LESS-THAN, LE, and NOT-FROM are used in the report request.

Reporting From Files With Alternate Indexes

Similar performance improvement is available for ESDS and KSDS files that use alternate indexes. An alternate index provides access to records in a key sequenced data set based on a key other than the primary key.

All benefits and limitations inherent with screening on the primary or partial key are applicable to screening on the alternate index or partial alternate index.

Note: It is not necessary to take an explicit indexed view to use the index.

CHAPTER 52

Using the Adapter for Web Services

Topics:

- Preparing the Web Services Environment
- Configuring the Adapter for Web Services
- Managing Web Services Metadata

The Adapter for Web Services allows client applications to access Web Services providers. The adapter converts:

- Internal data manipulation language (DML) requests or SQL requests into Web Services requests.
- Web Services responses into answer sets.

Preparing the Web Services Environment

The Adapter for Web Services minimally requires a Java Virtual Machine (JVM), Version 1.3 or higher, running on the same machine as the adapter for Web Services.

Configuring the Adapter for Web Services

Configuring the adapter consists of specifying connection and authentication information for at least one connection.

Declaring Connection Attributes

In order to access the Web application server hosting the target Web service, the adapter requires connection information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console configuration page. The Web Console adds the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Web application server by including multiple SET CONNECTION_ATTRIBUTES commands. The connection attributes used to check the metadata description, stored in the Access File, are not used during the request for service.

Procedure: How to Declare Connection Attributes From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the *Procedures* group folder, then expand the *Web Services* adapter folder and click a connection. The Add Connection for Web Services pane opens.

3. Enter the following parameters:

Parameter	Description
Connection name	Logical name used to identify this particular set of attributes.
WSDL URL	URL of the WSDL that describes the Web service. This is required for creating a synonym only; otherwise, it is ignored.
Select profile	The level of profile that the server is running. The standard iWay global profile, edasprof.prf, is the default. If you wish to define a user profile (*user*.prf) or a group profile (if available on your platform), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

4. Click *Next*. The Select End Point for Web Services Connection pane opens.
5. Select the End Points URL for the Web Application Server from the drop-down list.
6. Click the *Configure* button.

   Tip: Once the adapter is configured, the connection name appears in the navigation pane under *Configured*. If you left-click either the adapter name or the connection name, an appropriate set of options becomes available.

Syntax: How to Declare Connection Attributes Manually

The user ID and password are explicitly specified for each connection and passed to Web Services, at connection time, for authentication. The syntax is

```
ENGINE SOAP SET CONNECTION_ATTRIBUTES
  connection/ID,password:'endpointURL [WSDL_URL']'
```

where:

connection

Is the logical name used to identify this particular set of attributes.

endpointURL

Is the URL of the Web Application Server.

WSDL_URL

Is the URL of the WSDL that describes the Web service. This is required if you will be creating a synonym.

Note: ID and password are not implemented in the current release.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allow the application to access the Web application named SAMPLEAPP.

```
ENGINE SOAP SET CONNECTION_ATTRIBUTES
xignite/,:'http://www.xignite.com/xEdgar.asmx
http://www.xignite.com/xEdgar.asmx?WSDL'
```

Managing Web Services Metadata

In this section:

When the server accesses a Web Services provider, it needs to know how to pass parameters to and accept responses from the Web service operation. For each operation the server will access, you create a synonym that describes the operation and the server mapping of the Web Services data types.

Creating Synonyms

How to:

Create a Synonym From the Web Console

Create a Synonym Manually

Example:

Creating a Synonym

WSDL Segment for Generating a Master File for GetQuotes Web Service

WSDL Segment for Generating an Access File for GetQuotes Web Service

Reference:

Managing Synonyms

CHECKNAMES for Special Characters and Reserved Words

Master File Attributes

Access File Attributes

A synonym defines a unique logical name (also known as an alias) for each Web Services operation. Synonyms are useful because:

- They insulate client applications from changes to a request's location and identity. You can move or rename a request without modifying the client applications that use it; you need make only one change, redefining the request's synonym on the server.
- They provide support for the server's extended metadata features, such as virtual fields and security mechanisms.

Creating a synonym generates a Master File and an Access File. These are metadata that describe the Web Services request to the server.

Each synonym you create, whether manually or from the Web Console, represents a single operation.

Note: Each synonym describes the Web Services operations of a particular provider. If the operation parameters are changed, the synonym must be recreated.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata pane opens.

   To create a synonym, you must have configured the adapter. See *Configuring the Adapter for Web Services* on page 52-2 for more information.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The Select Operations for SOAP pane (Step 1 of 2) opens.

4. Enter the following parameters:

Parameter	**Description**
Select all Operations	Select this option to obtain a full list of operations. This is the default.
Filter Operations by Name	Select this option to filter the operations for which to create synonyms. When you select this option, you are prompted for an operation filter. Enter a string for filtering the operations, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all operations whose names begin with the letters ABC; %ABC to select operations whose names end with the letters ABC; %ABC% to select operations whose names contain the letters ABC at the beginning, middle, or end.

5. Click *Select Operations*. All operations that meet the specified criteria are displayed in the Select Synonym Candidate (Step 2 of 2) pane.

6. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

   When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

7. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

8. Enter the following additional parameters:

Parameter	**Description**
Select Application	Select an application directory. baseapp is the default value.
Prefix/Suffix	If you have Web services with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all operations have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

9. Complete your selection:

   To select all operations in the list, select the check box to the left of the *Default Synonym Name* column heading.

   To select specific operations, select the corresponding check boxes.

10. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

11. Click *Create Synonym*.

Synonyms are created and added under the specified application directory.

A status window displays the message:

```
All Synonyms Created Successfully
```

12. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '| ', '&', ' (', ') ', ' <', '> ', " ', ' =', "'"

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM [app/]synonym FOR operation DBMS SOAP AT connection
[CHECKNAMES] [UNIQUENAMES]
END
```

where:

app

Is the application namespace in which you want to create the synonym.

If your server is APP-enabled, you can specify this; otherwise, you must omit it.

synonym

Is an alias for a request (maximum 64 characters).

operation

Is a Web service operation.

SOAP

Indicates the Adapter for Web Services.

connection

Is the name of the connection to the Web application server as previously declared in the adapter’s configuration. When the synonym is created, this value is assigned to the CONNECTION attribute in the Access File.

Note: When the connection’s attributes were declared, the declaration must have specified the URL of the WSDL that describes the Web service operation. For more information about declaring connection attributes, see *Declaring Connection Attributes* on page 52-2.

CHECKNAMES

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

UNIQUENAMES

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

Example: Creating a Synonym

```
CREATE SYNONYM baseapp/LookupCIK FOR LookupCIK DBMS SOAP AT xignite
END
```

Sample WSDL: lookupcik.xml

```
<SOAP-ENV:Envelopexmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelop
e/" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:m0="http://www.xignite.com/services/">
        <SOAP-ENV:Header>
                <Header>
                        <m0:Username>String</m0:Username>
                        <m0:Password>String</m0:Password>
                        <m0:Tracer>String</m0:Tracer>
                </Header>
        </SOAP-ENV:Header>
        <SOAP-ENV:Body>
                <m:LookupCIK xmlns:m="http://www.xignite.com/services/">
                        <m:Name>String</m:Name>
                </m:LookupCIK>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The CREATE SYNONYM command generates the following Master and Access files.

Master File: LOOKUPCIK.MAS

```
FILENAME=M6ILO, SUFFIX=SOAP    , $
  SEGMENT=ROOT, SEGTYPE=S0, $
   GROUP=LOOKUPCIK, ALIAS=LookupCIK, USAGE=A30, ACTUAL=A30, $
    FIELDNAME=NAME, ALIAS=Name, USAGE=A30, ACTUAL=A30, $
    FIELDNAME=__RESPONSE, USAGE=TX80, ACTUAL=TX, $
  SEGMENT=RESPONSE, SEGTYPE=S0, SEGSUF=XML , PARENT=ROOT,
       POSITION=__RESPONSE, $
    FIELDNAME=RESPONSE, ALIAS=LookupCIKResponse, USAGE=A1, ACTUAL=A1, $
  SEGMENT=LOOKUPCIKRESULT, SEGTYPE=S0, PARENT=RESPONSE, $
    FIELDNAME=LOOKUPCIKRESULT, ALIAS=LookupCIKResult, USAGE=A1,ACTUAL=A1,
      REFERENCE=RESPONSE, PROPERTY=ELEMENT,  $
  SEGMENT=CIKLOOKUP, SEGTYPE=S0, PARENT=LOOKUPCIKRESULT, $
    FIELDNAME=CIKLOOKUP, ALIAS=CIKLookup, USAGE=A1, ACTUAL=A1,
      REFERENCE=LOOKUPCIKRESULT, PROPERTY=ELEMENT,  $
    FIELDNAME=CIK, ALIAS=CIK, USAGE=A30, ACTUAL=A30,
      REFERENCE=CIKLOOKUP, PROPERTY=ELEMENT,  $
    FIELDNAME=NAME, ALIAS=Name, USAGE=A30, ACTUAL=A30,
      REFERENCE=CIKLOOKUP, PROPERTY=ELEMENT,  $
```

Access File: LOOKUPCIK.ACX

```
SEGNAME=ROOT, CONNECTION=xignite, VERSION=1.1, OBJECT=LookupCIK,
  ACTION=http://www.xignite.com/services/LookupCIK,
  TARGETNS=http://www.xignite.com/services/, STYLE=DOCUMENT, $
```

Issue the following request:

```
TABLE FILE LOOKUPCIK
PRINT CIKLOOKUP.CIK CIKLOOKUP.NAME
IF ROOT.NAME IS 'LU'
END
```

The output is:

```
 NUMBER OF RECORDS IN TABLE=        5  LINES=      5

 PAGE     1

 CIK                                NAME
 ---                                ----
 0001116521                         LUCENT EN CORP
 0001278775                         LUCENT RETIREES ORGANIZATION
 0001175530                         LUCENT TECHNOLOGIES CAPITAL TR
 0001006240                         LUCENT TECHNOLOGIES INC
 0001261534                         LUCENTE FRANK JR
```

Example: WSDL Segment for Generating a Master File for GetQuotes Web Service

The following shows the elements for the input and output for the GetQuotes Web Service function:

```
- <s:element name="GetQuotes">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="Symbol" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="GetQuotesResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="GetQuotesResult"
type="s0:ArrayOfQuote" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:complexType name="ArrayOfQuote">
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="unbounded" name="Quote"
nillable="true" type="s0:Quote" />
  </s:sequence>
  </s:complexType>
- <s:complexType name="Quote">
- <s:complexContent mixed="false">
- <s:extension base="s0:Common">
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="Symbol" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Name" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Date" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Time" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="Open" type="s:double" />
  <s:element minOccurs="1" maxOccurs="1" name="High" type="s:double" />
  <s:element minOccurs="1" maxOccurs="1" name="Low" type="s:double" />
  <s:element minOccurs="1" maxOccurs="1" name="Last" type="s:double" />
  <s:element minOccurs="1" maxOccurs="1" name="Volume" type="s:double" />
  <s:element minOccurs="1" maxOccurs="1" name="Change" type="s:double" />
  <s:element minOccurs="1" maxOccurs="1" name="PercentChange"
type="s:double" />
  <s:element minOccurs="0" maxOccurs="1" name="Previous_Close"
type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Bid" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Bid_Size" type="s:string"
/>
  <s:element minOccurs="0" maxOccurs="1" name="Ask" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Ask_Size" type="s:string"
```

```
/>
  <s:element minOccurs="0" maxOccurs="1" name="High_52_Weeks"
type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Low_52_Weeks"
type="s:string" />
  </s:sequence>
  </s:extension>
  </s:complexContent>
  </s:complexType>
```

The following Master File is created from the Create Synonym process:

```
FILENAME=M6ILO, SUFFIX=SOAP    , $
  SEGMENT=ROOT, SEGTYPE=S0, $
   GROUP=GETQUOTES, ALIAS=GetQuotes, USAGE=A30, ACTUAL=A30, $
    FIELDNAME=SYMBOL, ALIAS=Symbol, USAGE=A30, ACTUAL=A30, $
    FIELDNAME=__RESPONSE, USAGE=TX80, ACTUAL=TX, $
  SEGMENT=RESPONSE, SEGTYPE=S0, SEGSUF=XML     , PARENT=ROOT,
POSITION=__RESPONSE, $
    FIELDNAME=RESPONSE, ALIAS=GetQuotesResponse, USAGE=A1, ACTUAL=A1, $
  SEGMENT=GETQUOTESRESULT, SEGTYPE=S0, PARENT=RESPONSE, $
    FIELDNAME=GETQUOTESRESULT, ALIAS=GetQuotesResult, USAGE=A1,
ACTUAL=A1,
      REFERENCE=RESPONSE, PROPERTY=ELEMENT,  $
  SEGMENT=QUOTE, SEGTYPE=S0, PARENT=GETQUOTESRESULT, $
    FIELDNAME=QUOTE, ALIAS=Quote, USAGE=A1, ACTUAL=A1,
      REFERENCE=GETQUOTESRESULT, PROPERTY=ELEMENT,  $
    FIELDNAME=SYMBOL1, ALIAS=Symbol, USAGE=A30, ACTUAL=A30,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=NAME, ALIAS=Name, USAGE=A30, ACTUAL=A30,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=DATE1, ALIAS=Date, USAGE=A30, ACTUAL=A30,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=TIME1, ALIAS=Time, USAGE=A30, ACTUAL=A30,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=OPEN1, ALIAS=Open, USAGE=D20.2, ACTUAL=A20,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=HIGH, ALIAS=High, USAGE=D20.2, ACTUAL=A20,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=LOW, ALIAS=Low, USAGE=D20.2, ACTUAL=A20,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=LAST, ALIAS=Last, USAGE=D20.2, ACTUAL=A20,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=VOLUME, ALIAS=Volume, USAGE=D20.2, ACTUAL=A20,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=CHANGE, ALIAS=Change, USAGE=D20.2, ACTUAL=A20,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=PERCENTCHANGE, ALIAS=PercentChange, USAGE=D20.2,
ACTUAL=A20,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=PREVIOUS_CLOSE, ALIAS=Previous_Close, USAGE A30,
ACTUAL=A30,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=BID, ALIAS=Bid, USAGE=A30, ACTUAL=A30,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=BID_SIZE, ALIAS=Bid_Size, USAGE=A30, ACTUAL=A30,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=ASK, ALIAS=Ask, USAGE=A30, ACTUAL=A30,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=ASK_SIZE, ALIAS=Ask_Size, USAGE=A30, ACTUAL=A30,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=HIGH_52_WEEKS, ALIAS=High_52_Weeks, USAGE=A30, ACTUAL=A30,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
    FIELDNAME=LOW_52_WEEKS, ALIAS=Low_52_Weeks, USAGE=A30, ACTUAL=A30,
      REFERENCE=QUOTE, PROPERTY=ELEMENT,  $
```

The first segment contains the input parameters to the Web Service function. In the example above, SYMBOL is the only input parameter.

Note: You should not reference fields with a format of A1 or TX in queries. These fields are used for internal purposes only.

Example: WSDL Segment for Generating an Access File for GetQuotes Web Service

The iWay Server requires information about how to access the Web Service function. This information exists in the Access File that is derived from the WSDL file. The Access File is created at the same time the Master File is created during the Create Synonym process.

The following example shows certain information from the WSDL file that is used in the Access File:

```
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://www.xignite.com/services/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://www.xignite.com/services/
xmlns="http://schemas.xmlsoap.org/wsdl/">
   <operation name="GetQuotes">
   <soap:operation soapAction="http://www.xignite.com/services/GetQuotes"
style="document" />
```

The following shows an example of an Access File that is created during the Create Synonym process:

```
SEGNAME=ROOT, CONNECTION=Quotes, VERSION=1.1, OBJECT=GetQuotes,
  ACTION=http://www.xignite.com/services/GetQuotes,
  TARGETNS=http://www.xignite.com/services/, STYLE=DOCUMENT, $
```

The mapping for the Access File works as follows:

`SEGNAME`	Default is ROOT
`CONNECTION`	Connection name provided for the Connect String in the EDASPROF file.
`VERSION`	Version of XML
`OBJECT`	Derived from the following: `< operation name="GetQuotes">`
`ACTION`	Derived from the following: `soapAction ="http://www.xignite.com/services/GetQuotes"`
`TARGETNS`	Derived from the Target Namespace of the document: `targetNamespace ="http://www.xignite.com/services/"`
`STYLE`	Derived from the style element: `style="document"`

Reference: Master File Attributes

Attribute	Description
`PROPERTY`	Indicates whether the field corresponds to an XML attribute or an XML element. PROPERTY is used only in Master File segments that map directly to the response document.
`REFERENCE`	Identifies this field's parent element in the XML hierarchy, as defined by the Web Services document's XML schema. The value's format is *`segmentname.fieldname`* where: *`segmentname`* Is the name of the Master File segment in which the field resides. *`fieldname`* Is the name of the field that corresponds to the parent element.

Reference: Access File Attributes

Attribute	Description
`SEGNAME`	Value must be identical to the SEGNAME value in the Master File.
`CONNECTION`	Indicates a previously declared connection. The syntax is: `CONNECTION=`*`connection`*
`VERSION`	Web Services version being used. The adapter supports Version 1.1.
`ACTION`	Web service operation being described.
`OBJECT`	Fully-qualified name of the Web Services request.
`ENCODING`	URL of the encoding schema.
`TARGETNS`	Target name space specified in the operation's WSDL description. (This will either be global to the Web service's WSDL, or unique to this operation's WSDL.)
`STYLE`	Value of the Web Services document WSDL's style element (RPC or Document).

Data Type Support

The following table lists how the server maps XSD data types in a Master File.

XSD Data Type	USAGE	ACTUAL
string	A30	A30
double	D20.2	A20
float	F15.2	A15
decimal	P20.3	A20
int	I11	A11
short	I6	A6
long	P20	A20
boolean	A5	A5
dateTime	HYYMDm	A27
time	HHISsm	A15
date	YYMD	A10
gYearMonth	HYYM	A8
gYear	HYY	A5
gMonthDay	HMD	A6
gDay	HD	A3
gMonth	HM	A4
integer	P33	A33
nonPositiveInteger	P33	A33
negativeInteger	P33	A33
nonNegativeInteger	P32	A32
unsignedLong	P20	A20
unsignedInt	P10	A10
unsignedShort	I5	A5

XSD Data Type	**USAGE**	**ACTUAL**
positiveInteger	P32	A32
byte	I4	A4
unsignedByte	I4	A4
normalizedString	A30	A30
token	A30	A30
Name	A30	A30
NMTOKEN	A30	A30
ID	A30	A30
hexBinary	A30	A30
language	A30	A30
anyURI	A30	A30
QName	A30	A30

Changing the Length of Character Strings

By default, when the Adapter for Web Services creates a synonym, it maps all fields defined as string in the XML schema to 10-byte fixed-length alphanumeric in the Master File. You can change the length (the variability and the number of bytes) using the SET VARCHAR and SET LENGTH commands.

Syntax: How to Switch Between Variable and Fixed-length Strings

By default, when the Adapter for Web Services creates a synonym, it maps all fields defined as string in the XML schema to fixed-length alphanumeric in the Master File. You can change this default to variable length using the SET VARCHAR command

```
ENGINE SOAP SET VARCHAR {ON|OFF}
```

where:

`SOAP`

Indicates the Adapter for Web Services.

`ON`

Maps all fields defined as string in the XML schema to variable-length alphanumeric (A*n*V) in the Master File's USAGE and ACTUAL attributes.

`OFF`

Maps all fields defined as string in the XML schema to fixed-length alphanumeric (A) in the Master File's USAGE and ACTUAL attributes. OFF is the default value.

Syntax: How to Change String Length

By default, when the Adapter for Web Services creates a synonym, it maps all fields defined as string in the XML schema to 10-byte alphanumeric in the Master File. You can change this default length using the SET FIELDLENGTH command

```
ENGINE SOAP SET FIELDLENGTH length
```

where:

`SOAP`

Indicates the Adapter for Web Services.

`length`

Is the length, in bytes, assigned to the ACTUAL and USAGE attributes of all fields defined in the XML schema as string. The maximum length is 3000.

Changing the Precision and Scale of Numeric Fields

How to:

Change the Precision and Scale of Numeric Fields

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric fields returned by a request by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL attributes of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax: How to Change the Precision and Scale of Numeric Fields

```
ENGINE SOAP SET CONVERSION RESET
ENGINE SOAP SET CONVERSION format RESET
ENGINE SOAP SET CONVERSION format [PRECISION pp [ss]]
ENGINE SOAP SET CONVERSION format [PRECISION MAX]
```

where:

`SOAP`

Indicates the Adapter for Web Services.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER` indicates that the command applies only to INTEGER columns.

`DECIMAL` indicates that the command applies only to DECIMAL columns.

`REAL` indicates that the command applies only to single precision floating point columns.

`FLOAT` indicates that the command applies only to double precision floating point columns.

`pp`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

ss

Is the scale. This is valid with DECIMAL, REAL, and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set mm to 0 (zero).

`MAX`

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	**MAX Precision**
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER fields to 7:

```
ENGINE SOAP SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE SOAP SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER fields to the default:

```
ENGINE SOAP SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE SOAP SET CONVERSION RESET
```

CHAPTER 53

Using the Adapter for XML

Topics:

The Adapter for XML allows applications to access XML data sources. The adapter converts application requests into native XML statements and returns optimized answer sets to the requesting application. If the adapter has read/write capabilities, it inserts the data from an application to the data source.

Preparing the XML Environment

The Adapter for XML does not require any environment variables to be set up.

Configuring the Adapter for XML

You can configure the Adapter for XML from the Web Console.

Procedure: How to Configure the Adapter From the Web Console

1. Start the Web Console and, in the navigation pane, click *Data Adapters*.
2. Expand the *Add* folder, expand the XML-*based* group folder, then expand the *XML* adapter folder and click a connection. The Add XML to Configuration pane opens.
3. No input is required. Simply click the *Configure* button.

Associating a Master File With an XML Document

How to:

Associate a Master File With an XML Document

Example:

Issuing a FILEDEF Command for an XML Column

Issuing a FILEDEF Command for an XML Collection

You must explicitly issue a FILEDEF command to associate a Master File with an XML document. In order to properly code the FILEDEF command, you need to determine how the XML documents are organized. If the documents are in one file and are described by the same DTD, the organization is called a COLUMN. If there are various documents in one directory and all are described by the same DTD, then the organization is called a COLLECTION.

Source	Collection Organization	Column Organization
disk	supported	supported
http	not supported	supported
https	not supported	supported

Syntax: How to Associate a Master File With an XML Document

```
FILEDEF ddname source filename
```

where:

`ddname`

Is the logical reference name matching the desired Master File.

`source`

Is the location of the XML document. Possible values are `http`, `https`, and `disk`.

`filename`

For an XML Column, is the full path to the source file.

For an XML Collection, is the full path to the source directory. The directory name must be followed by the wildcard characters '*.*'.

The file name must start with 'http://' or ‘https://' if the XML documents are to be read using an HTTP or HTTPS session.

Example: Issuing a FILEDEF Command for an XML Column

For Windows, when you have the ordercol.xml document under C:\, use the following FILEDEF in your profile:

```
FILEDEF ordercol disk C:\ordercol.xml
```

For UNIX, when you have the ordercol.xml document under /usera/xml/, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK /usera/xml/ordercol.xml
```

For OS/390 and Z/OS, when you have the ordercol.xml document in TEST.XML.DOC, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK //'TEST.XML.DOC'
```

For HTTP and HTTPS, when you have the ordercol.xml document under http://www.test.com/xmltest/ (or https://www.test.com/xmltest/), use the following FILEDEF in your profile:

```
FILEDEF order http http://www.test.com/xmltest/ordercol.xml
```

or

```
FILEDEF order http https://www.test.com/xmltest/ordercol.xml
```

Example: Issuing a FILEDEF Command for an XML Collection

For Windows, when you have the order1.xml and order2.xml documents under C:\ORDERS, use the following FILEDEF in your profile:

```
FILEDEF order disk C:\orders\*.*
```

For UNIX, when you have the ordercol.xml document under /usera/orders, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK //'TEST.XML.DOC(dtd)'
```

For OS/390 and Z/OS, when you have the order document in TEST.XML.DOC(dtd), use the following FILEDEF in your profile:

```
FILEDEF order DISK /usera/orders/*.*
```

Accessing XML Documents From Relational DBMS CLOB Fields

XML documents might be stored in any fields or columns in any data source. Reporting from such documents is supported by defining their structure as sub-trees attached to a parent segment which describes the original data. The CREATE SYNONYM process must be run against the data in the DBMS and against the DTD describing the XML document. The two Master Files must then be manually combined to make the DTD Master File a child of the Master File created against the DBMS. A FILEDEF is not needed in this instance.

Managing XML Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the XML data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each XML data structure that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

For details on how to create a synonym through the Web Console, see the *Server Configuration and Operation* manual for your platform. The CREATE SYNONYM command creates a Master File and an Access File based on a given DTD. The DTD can be embedded within an XML document or be and external DTD.

Procedure: How to Create a Synonym From the Web Console

1. Start the Web Console and, in the navigation pane, click *Metadata*. The Managing Metadata page opens.

   To create a synonym, you must have configured the adapter. See *Configuring the Adapter for XML* on page 53-2 for more information.

2. Click *New Synonym* in the navigation pane to open the Select Adapter and Connection to Create Synonym pane.

3. Click a connection for the configured adapter. The XML Definition Document pane (Step 1 of 2) opens.

4. Enter the following parameters to specify the locations of the document definitions (schema) and, optionally, of the xml data files:

Parameter	**Description**
URL	The options are: **File System,** which indicates the location of the data source. On Windows and UNIX, this may be a local full path or a URL. If it is a URL, it must start with http:// or https://. **HTTP,** which enables you to specify a relative path to the location of the XML document (This functionality is not available when the XML document mentioned in the CREATE SYNONYM command is a local file.) Other options on this pane vary depending on this selection.
Collection of XML definitions	If you chose File System, enter the location of the schema information.

Parameter	**Description**
Files Extension	If you chose File System, you may enter a file extension as a filtering criterion: .dtd or .xsd. Enter * to list files with all extensions.
HTTP Address	If you chose HTTP, enter the http address of a directory that contains the .dtd or .xsd files you are using to create the synonym.
Document Name	If you chose HTTP, enter the name of the .dtd or .xsd file you are using for the synonym.
Directory Path	If you chose File System, this optional entry specifies the location of the xml data files. If provided, this entry is used to create a FILEDEF.

5. Click *Submit*. The Create Synonym (Step 2 of 2) pane opens.
6. Click the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

   When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.
7. Click the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

8. Enter the following additional parameters:

Parameter	**Description**
Select Application	Select an application directory. baseapp is the default value.
Prefix/Suffix	If you have Web services with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters. If all operations have unique names, leave prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

9. Complete your selection:

To select all documents in the list, select the check box to the left of the *Default Synonym Name* column heading.

To select specific documents, select the corresponding check boxes.

10. The Default Synonym Name column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

11. If you entered a directory location (optional) for the xml data files in step 1, a File Name column is displayed. If you wish, you can choose individual document files from the corresponding drop-down list.

12. Alternatively, you can explicitly enter the locations and file names of a individual documents in the Dataset Location column.This specification is optional.

13. Click *Create Synonym*.

Synonyms are created and added under the specified application directory.

A status window displays the message:

```
All Synonyms Created Successfully
```

14. From the message window, you can click *Go to Metadata page* to view, edit, test, or manage the new synonym.

Reference: Managing Synonyms

In the Metadata navigation pane, click the name of the synonym to access the following options:

Open	Opens the Master File in the right pane, for viewing and editing using a graphical interface. You can also refresh the synonym.
Edit as Text	Enables you to manually edit the synonym's Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to manually edit the synonym's Access File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Access File.
Sample Data	Retrieves up to 20 rows from the associated data source.
Drop Synonym	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Reference: CHECKNAMES for Special Characters and Reserved Words

CHECKNAMES checks and adjusts the following special characters and names:

- Full list of special characters to be converted to underscore:

  '-', ' ', '\', '/', ',', '$', '.', ';', '+', '*', '%', '| ', '&', ' (', ') ', ' <', '> ', " ', ' =', "'"

- List of reserved words that are not to be used as names in the created synonym:

  ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG, BEGIN, BETWEEN, BINARY, BIT, BOTH, BY, CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COMMIT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR, DATE, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION, GET, GRANT, GROUP, HAVING, IF, IN, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERATE, JOIN, KEY, LEADING, LEAVE, LEFT, LIKE, LOOP, MAX, MIN, NATURAL, NOT, NULL, NULLIF, NUMERIC, OF, ON, ONLY, OPEN, OPTION, OR, ORDER, OUT, OUTER, PRECISION, PREPARE, PRIMARY, PROCEDURE, REAL, REFERENCES, REPEAT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS, SCHEMA, SELECT, SET, SMALLINT, SOME, SUM, TABLE, THEN, TIME, TIMESTAMP, TO, TRAILING, UNION, UNIQUE, UNTIL, UPDATE, USER, USING, VALUES, VARCHAR, VARYING, VIEW, WHEN, WHERE, WHILE, WITH, WORK, END, COLFETCH, CONCAT, DATABASE, DATETIME, DAY, DAYS, EXPLAIN, GRAPHIC, HOUR, HOURS, IMAGE, INCLUDE, INDEX, LOCK, LOGICAL, LONG, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MINUTE, MINUTES, MONEY, MONTH, MONTHS, NUMBER, OPTIMIZE, PACKAGE, PERCENT, PLAN, PROGRAM, PURGE, QUERYNO, RAW, RESET, SECOND, SECONDS, SERIAL, SMALLFLOAT, STOGROUP, SYNONYM, SYSNAME, TABLESPACE, TEXT, TRUNCATE, USER_TYPE_NAME, VARBINARY, VARGRAPHIC, YEAR, YEARS

Syntax: How to Create a Synonym Manually

```
CREATE SYNONYM appname/synonym FOR xmlname DBMS XML AT location
[CHECKNAMES] [UNIQUENAMES]
END
```

where:

appname

Is the 1 to 64 character application namespace where you want to create the synonym. If your server is APP enabled, this application name must be used. If your server is not APP enabled, then this application name must not be used.

synonym

Is an alias for the data source. This value must correspond to the logical name defined in the FILEDEF command.

xmlname

Is the name of the data source. The data source can be either an XML document or a DTD document.

`location`

Indicates the location of the data source. On Windows and UNIX, this may be a local full path or a URL. If it is a URL, it must start with http:// or https://.

In the case of HTTP or HTTPS, *location* may be a relative path to the location of the XML document mentioned in the CREATE SYNONYM command. This functionality is not available when the XML document mentioned in the CREATE SYNONYM command is a local file.

Note: The data source for the CREATE SYNONYM command may be a DTD, an XML document with embedded DTD, or an XML document which points to an external DTD. In the case of the latter, the DOCTYPE statement from the XML document:

```
<!DOCTYPE root_element_name SYSTEM "dtd_location">
```

is being parsed in order to retrieve the location of the DTD.

If *location* is a full path, the DTD will be processed from that path.

If *location* is a file name, the DTD will be assumed to be under location (in the same directory as the XML document mentioned in the CREATE SYNONYM command).

`CHECKNAMES`

Converts all special characters to underscores and performs a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) The CHECKNAMES parameter ensures that names adhere to iWay specifications. See *CHECKNAMES for Special Characters and Reserved Words* for more information.

When this parameter is omitted (the default), only the following characters are converted to underscore are: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

`UNIQUENAMES`

Sets the scope for field and group names to the entire synonym; this ensures that no duplicate names are used, even in different segments of the synonym.

When this option is omitted (the default), the scope is the segment.

Note: CREATE SYNONYM can span more than one line, however a single element cannot span more than one line.

If there is no user-defined entity in the DTD, the Access File contains a $ sign. If there is a user-defined entity in the DTD, then the Access File contains the following for each entity

```
Eentity_name Ventity_value
```

where:

`entity_name`

Is the entity name extracted from the DTD.

`entity_value`

Is the entity value extracted from the DTD.

Example: Creating a Synonym for the XMLSLONG DTD

The following example shows how to create a synonym for the XMLSLONG DTD.

```
CREATE SYNONYM LONGNAME FOR xmlslong.dtd DBMS XML AT &LOCATION
```

XMLSLONG.dtd

```
<!ELEMENT Long_root_element (Long_entity_name,Line*,Comment*)>
<!ATTLIST Long_root_element key CDATA #REQUIRED>
<!ELEMENT Long_entity_name (#PCDATA)>
<!ELEMENT Line (#PCDATA)>
<!ELEMENT Comment (#PCDATA)>
<!ENTITY a "11111111111111111111111111111111111111111111111111111111111">
<!ENTITY b "22222222222222222222222222222222222222222222222222222222222">
<!ENTITY c "33333333333333333333333333333333333333333333333333333333333">
<!ENTITY d "44444444444444444444444444444444444444444444444444444444444">
<!ENTITY e "55555555555555555555555555555555555555555555555555555555555">
<!ENTITY f "66666666666666666666666666666666666666666666666666666666666">
<!ENTITY g "77777777777777777777777777777777777777777777777777777777777">
<!ENTITY h "HELLO HELLO HELLO HELLO HELLO HELLO HELLO HELLO HELLO HELLO">
<!ENTITY i "88888888888888888888888888888888888888888888888888888888888">
<!ENTITY j "12345678901234567890123456789012345678901234567890123456789">
<!ENTITY k "00000000000000000000000000000000000000000000000000000000000">
<!ENTITY l "date">
<!ENTITY n "name">
<!ENTITY p "It is possible to have">
```

Generated Master File LONGNAME.mas

```
FILENAME=LONGNAME, SUFFIX=XML , $
SEGMENT=LONG_ROOT_ELEMENT, SEGTYPE=S0, $
FIELDNAME=LONG_ROOT_ELEMENT, ALIAS='Long_root_element', USAGE= A1,
ACTUAL=A1, $
FIELDNAME=KEY, ALIAS='key', USAGE=A10V, ACTUAL=A10V, $
FIELDNAME=LONG_ENTITY_NAME, ALIAS='Long_entity_name', USAGE= A10V,
ACTUAL=A10V, $
SEGMENT=LINE, SEGTYPE=S0, PARENT=LONG_ROOT_ELEMENT, $
FIELDNAME=LINE, ALIAS='Line', USAGE= A10V, ACTUAL=A10V, $
SEGMENT=COMMENT, SEGTYPE=S0, PARENT=LONG_ROOT_ELEMENT, $
FIELDNAME=COMMENT, ALIAS='Comment', USAGE= A10V, ACTUAL=A10V, $
```

Generated Access File Longname.acx

```
Ea V11111111111111111111111111111111111111111111111111111111111
Eb V22222222222222222222222222222222222222222222222222222222222
Ec V33333333333333333333333333333333333333333333333333333333333
Ed V44444444444444444444444444444444444444444444444444444444444
Ee V55555555555555555555555555555555555555555555555555555555555
Ef V66666666666666666666666666666666666666666666666666666666666
Eg V77777777777777777777777777777777777777777777777777777777777
Eh VHELLO HELLO HELLO HELLO HELLO HELLO HELLO HELLO HELLO HELLO
Ei V88888888888888888888888888888888888888888888888888888888888
Ej V12345678901234567890123456789012345678901234567890123456789
Ek V00000000000000000000000000000000000000000000000000000000000
El Vdate
En Vname
Ep V
```

Example: Creating a Synonym for Direct Access to a Data Source

For Windows, when you have the order.dtd document under C:\, use the following syntax:

```
CREATE SYNONYM ORDER FOR order.dtd DBMS XML AT C:\
END
```

For UNIX, when you have the order.dtd document under /usera/, use the following syntax:

```
CREATE SYNONYM ORDER FOR order.dtd DBMS XML AT /usera/
END
```

For HTTP or HTTPS Stream, when you have the order.xml document under http://www.test.com/xmltest (or https://www.test.com/xmltest), use the following syntax:

```
CREATE SYNONYM ORDER FOR order.dtd DBMS XML AT
http://www.test.com/xmltest/
END
```

or

```
CREATE SYNONYM ORDER FOR order.dtd DBMS XML AT
https://www.test.com/xmltest/
END
```

Procedure: How to Create a Synonym to Access XML Documents From Relational DBMS CLOB Fields

You have a table in a DBMS with one or more columns storing XML data. In order to report from the XML data, there are various steps to be taken:

1. Create the Master File for the relational data source using the format for that DBMS.

   ```
   FILE=XMLPRO1 ,SUFFIX=SQLPRO ,$
   SEGNAME=XMLPRO1 ,SEGTYPE=S0 ,$
   FIELD=FLD1 ,FLD1 ,A2   ,A2 ,MISSING=ON ,$
   FIELD=FLD2 ,FLD2 ,TX50 ,TX ,MISSING=ON ,$
   FIELD=FLD3 ,FLD3 ,TX50 ,TX ,MISSING=ON ,$
   ```

2. Create a Master File from the DTD for the XML data in the column in the table in the DBMS. If there are two DTDs, create a Master File for each DTD.

3. Manually combine the Master Files. On each root segment for the XML Master File, add three fields: position, parent and segsuf. The POSITION keyword identifies the field containing the XML document. The PARENT field describes the original data source. The field SEGSUF defines the root segment of an XML document representing sub-tree. The total length of all fields in the Master File must not exceed the FOCUS limitation of 32k. If it does, the query will fail.

```
FILENAME=BASEAPP/ORDER, SUFFIX=XML     , $
  SEGMENT=ORDER, SEGTYPE=S0, $
    FIELDNAME=ORDER, ALIAS='Order', USAGE=A1, ACTUAL=A1, $
    FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=STATUS, ALIAS='Status', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=TOTALPRICE, ALIAS='TotalPrice', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=DATE, ALIAS='Date', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=PRIORITY, ALIAS='Priority', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=CLERK, ALIAS='Clerk', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=SHIPPRIORITY, ALIAS='ShipPriority', USAGE=A10, ACTUAL=A10,$
    FIELDNAME=COMMENT, ALIAS='Comment', USAGE=A10, ACTUAL=A10, $
FILENAME=BASEAPP/PORDER, SUFFIX=XML     , $
  SEGMENT=PORDER, SEGTYPE=S0, $
    FIELDNAME=PORDER, ALIAS='POrder', USAGE=A1, ACTUAL=A1, $
    FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=ADDRESS, ALIAS='Address', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=STATE, ALIAS='State', USAGE=A10, ACTUAL=A10, $
```

Combined Master file:

```
FILE=XMLPRO1    ,SUFFIX=SQLPRO ,$
SEGNAME=XMLPRO1 ,SEGTYPE=S0    ,$
FIELD=FLD1 ,FLD1 ,A2   ,A2  ,MISSING=ON ,$
FIELD=FLD2 ,FLD2 ,TX50 ,TX  ,MISSING=ON ,$
FIELD=FLD3 ,FLD33,TX50 ,TX  ,MISSING=ON ,$
SEGMENT=ORDER, SEGTYPE=S0,POSITION=FLD2,PARENT=XMLPRO1,SEGSUF=XML, $
  FIELDNAME=ORDER, ALIAS='Order', USAGE=A1, ACTUAL=A1, $
  FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=STATUS, ALIAS='Status', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=TOTALPRICE, ALIAS='TotalPrice', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=DATE, ALIAS='Date', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=PRIORITY, ALIAS='Priority', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CLERK, ALIAS='Clerk', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=SHIPPRIORITY, ALIAS='ShipPriority', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=COMMENT, ALIAS='Comment', USAGE=A10, ACTUAL=A10, $
 SEGMENT=PORDER, SEGTYPE=S0,POSITION=FLD3,PARENT=XMLPRO1,SEGSUF=XML $
    FIELDNAME=PORDER, ALIAS='Order', USAGE=A1, ACTUAL=A1, $
    FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=ADDRESS, ALIAS='Address', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=STATE, ALIAS='State', USAGE=A10, ACTUAL=A10, $
```

Data Type Support

How to:

Set the Actual and Usage Attributes

The CREATE SYNONYM process uses a DTD as the source for creating the synonym. DTDs do not contain data type descriptions, therefore the actual and usage attributes of all fields in the Master File are set to ALPHA data type.

If you try to perform arithmetic operations (-, +, >, <, etc.) on fields that are numbers or dates in the XML document but are mapped as ALPHA in the Master File, you may not get the expected results since the arithmetic operations are performed on string literals. To override this problem you may modify the USAGE attribute of a field as described in the following sections.

Note: The data type defined in the ACTUAL attribute of a field must remain as ALPHA.

In order to change the default setting you must issue the SET VARCHAR command.

The following table lists how the server maps XSD data types in a Master File. You can change some of these mapping defaults, as described in *Changing the Length of Character Strings* on page 53-18 and *Changing the Precision and Scale of Numeric Fields* on page 53-19.

XSD Data Type	**USAGE Attribute**	**ACTUAL Attribute**
decimal	P20.3	A20
integer	P33	A33
nonPositiveInteger	33	A33
negativeInteger	P33	A33
boolean	A5	A5
long	P20	A20
int	I11	A11
short	I6	A6
byte	I4	A4
nonNegativeInteger	P32	A32
unsignedLong	P20	A20
unsignedInt	P10	A10
unsignedShort	I5	A5
unsignedByte	I4	A4
positiveInteger	P32	A32
double	D20.2	A20
float	F15.2	A15
dateTime	HYYMDm	A27
time	HHISsm	A15
date	YYMD	A10
gYearMonth	HYYM	A8
gYear	HYY	A5

XSD Data Type	USAGE Attribute	ACTUAL Attribute
gMonthDay	HMD	A6
gDay	HD	A3
gMonth	HM	A4
string	A30	A30
normalizedString	A30	A30
token	A30	A30
Name	A30	A30
NMTOKEN	A30	A30
ID	A30	A30
hexBinary	A30	A30
language	A30	A30
anyURI	A30	A30
QName	A30	A30

Syntax: How to Set the Actual and Usage Attributes

```
ENGINE XML SET VARCHAR {ON|OFF}
```

The ACTUAL and USAGE attributes of each field in the Master File is set arbitrarily to A10. You can override this setting using the SET FIELDLENGTH command

```
ENGINE XML SET FIELDLENGTH nnn
```

where:

nnn

Is the length assigned to actual and usage attributes of all fields in the Master File during the create synonym process. The maximum length is 3000.

Changing the Length of Character Strings

How to:

Switch Between Variable and Fixed-length Strings

Change String Length

By default, when the Adapter for XML creates a synonym, it maps all fields defined as string in the XML schema to 10-byte fixed-length alphanumeric in the Master File. You can change the length (the variability and the number of bytes) using the SET VARCHAR and SET LENGTH commands.

Syntax: How to Switch Between Variable and Fixed-length Strings

By default, when the Adapter for XML creates a synonym, it maps all fields defined as string in the XML schema to fixed-length alphanumeric in the Master File. You can change this default to variable length using the SET VARCHAR command

```
ENGINE [XML] SET VARCHAR {ON|OFF}
```

where:

`XML`

Indicates the Adapter for XML. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps all fields defined as string in the XML schema to variable-length alphanumeric (A*n*V) in the Master File's USAGE and ACTUAL attributes.

`OFF`

Maps all fields defined as string in the XML schema to fixed-length alphanumeric (A) in the Master File's USAGE and ACTUAL attributes. This is the default.

Syntax: How to Change String Length

By default, when the Adapter for XML creates a synonym, it maps all fields defined as string in the XML schema to 10-byte alphanumeric in the Master File. You can change this default length using the SET FIELDLENGTH command

```
ENGINE [XML] XML SET FIELDLENGTH length
```

where:

`XML`
: Indicates the Adapter for XML. You can omit this value if you previously issued the SET SQLENGINE command.

`length`
: Is the length, in bytes, assigned to the ACTUAL and USAGE attributes of all fields defined in the XML schema as string. The maximum length is 3000.

Changing the Precision and Scale of Numeric Fields

How to:

Change the Precision and Scale of Numeric Fields

Example:

Setting the Precision and Scale Attributes

You can alter the length and scale of numeric fields returned by a request by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL attributes of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Syntax: How to Change the Precision and Scale of Numeric Fields

```
ENGINE [XML] SET CONVERSION RESET
ENGINE [XML] SET CONVERSION format RESET
ENGINE [XML] SET CONVERSION format [PRECISION precision [scale]]
ENGINE [XML] SET CONVERSION format [PRECISION MAX]
```

where:

`TMNIN`
: Indicates the Adapter for XML. You can omit this value if you previously issued the SET SQLENGINE command.

`RESET`

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

`format`

Is any valid format supported by the data source. Possible values are:

`INTEGER`

Indicates that the command applies only to INTEGER.

`DECIMAL`

Indicates that the command applies only to DECIMAL columns.

`REAL`

Indicates that the command applies only to single precision floating point columns.

`FLOAT`

Indicates that the command applies only to double precision floating point columns.

`precision`

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (See description of MAX).

`scale`

Is the scale. This is valid with DECIMAL, REAL, and FLOAT data types. If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set *mm* to 0 (zero).

```
MAX
```

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	33
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: Setting the Precision and Scale Attributes

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE XML SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE XML SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE XML SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE XML SET CONVERSION RESET
```

Conversion

The data in an XML document may reflect dates or numeric values, however, all the fields in a Master File created by the CREATE SYNONYM command are set to the ALPHA data type

Numeric Values

In order to enable arithmetic operations on numeric fields, the data type specified in the USAGE attribute of a numeric field needs to be modified, depending on the data in the XML document, to one of the following data types: Integer (I), Double Float (D) or Decimal (P). If the data type is modified to Double Float or Decimal, use scale and precision as necessary to describe the data in the XML document.

Furthermore, it is recommended that the length of the ALPHA data type specified in the ACTUAL attribute of the numeric field be modified to reflect the maximum length of the data in the XML document.

Dates in XML

In order to enable arithmetic operations on dates, the data type specified in the USAGE attribute of a date field needs to be modified, depending on the date format used in the XML document, to one of the following data types: YYMD, MDYY or DMYY.

Furthermore, the length of the ALPHA data type specified in the ACTUAL attribute of the date field needs to be modified to 10.

Note:

1. Once you have set the data type in the Master File to one of the date data types, you must make sure that in each of the XML documents from which you report the date format is the same as the one you defined in the Master File. If the data types differ, an error will result.
2. The date format in the answer set is not derived from the date format used in the XML document and is always set to YYYY/MM/DD.

Example: Using Dates in XML

If in the XML document you have the following format:	**Then use USAGE=**
1996-01-30	YYMD
01-30-1996	MDYY
30-01-1996	DMYY

Using XML XFOCUS

In this section:

The server can receive XML documents from an MQseries queue or from a SOAP protocol stream and then perform one of two operations: EXECUTE or ARCHIVE.

Using XML XFOCUS to Execute XML Documents

The execute operation takes an XML document, extracts a request (data retrieval or RPC), processes the request, and returns an answer set in XML format to the calling environment. The eda.dtd triggers the submission for execution. The XML document must conform to the eda.dtd. This is a sample of an XML document. In this request, an sql select from car is being requested:

```
<?xml version="1.0" ?>
<!DOCTYPE xmlfoc01 SYSTEM "eda.dtd">
   <eda>
     <request>
      <connection>
        <dsn>unused</dsn>
        <user>unused</user>
        <password>unused</password>
        <sql>
         <query>select car from car</query>
        </sql>
       </connection>
      </request>
    </eda>
```

Using XML XFOCUS to Archive XML Documents

The archive operation takes any XML document, extracts its content and, preserving the hierarchical structure, stores it in a FOCUS database. The database can then be queried from a Business Intelligence viewpoint or the XML can be extracted in its original format for further processing. The steps to configure this archival procedure are:

1. Configure an MQXML listener. To configure the listener you need to have set up three local queues under your MQseries queue manager. For example, INQUEUE, OUTQUEUE and ERRORQUEUE.

   From the Web Console home page, click *Listeners*. This brings you to the Listeners configuration screen where you can add MQXML as the listener. When you click *MQXML*, you will see:

   QMANGER = the mqseries queue manager.

   INQUEUE = the queue to which the XML document is written.

   OUTQUEUE = the queue to which the status message is written.

   ERRORQUEUE = the queue to which the XML document is sent if there is a problem with the configuration and it cannot be archived.

   Once the listener is configured, you will need to click the button to save and restart the server. The listener (MQXML) status should now be active.

2. Configure for the Adapter for XML. Go back to the home page and either click *Data Adapters* or *Configure Adapter*. This will bring you to the Configuring Data Adapters screen. From here you choose XML as the adapter you want to add and click the CONFIGURE button. When configuration is complete, you can click the selected adapter and see a pull down menu from which you can choose Create Synonym and go the XML directory input screen. You must create a synonym for the DTD of the files you will be archiving.

   When the synonym is created, click the synonym to see the ARCHIVE option under the pull down menu. Click this option and you will see that three files will be created once you click the CREATE button. These files have the same names as the DTD, but with different extensions. For example, if the DTD is called SAMPLES.DTD, you will see:

   ```
   SOURCE = baseapp/sample

   TARGET = baseapp/sample_arc

   LOAD_PROCEDURE = baseapp/sample_load
   ```

   Remember the name of the load procedure because you will need it to configure the service. You must repeat the synonym creation process for each DTD you want to archive.

3. To configure a new service, go to WORKSPACE on the Web Console home page, select CONFIGURE from the pull down menu. This will bring you to the WORKSPACE CONFIGURATION screen. Click *Services* and you will jump a new page where you will see an option for service with DEFAULT. Pull down on DEFAULT and pick the NEW SERVICES option.

   New service name - The new service you are adding. It must be in uppercase and be the name of the DTD.

   Maximum - Accept the default (10).

   Number_ready - Accept the default (2).

   Deployment - Must be pooled.

   Pool_user - The user ID of the administrator of the server.

   Pool_password - The password of the administrator of the server.

   Max_sessions_per_agent- Accept the default (1).

   Queueing - Must be on.

   maxium_q - Set to 60.

   queue_limit - Take default (1)

   idle_session_limit=Accept the default (1).

   idle_agent_limit-Accept the default (120).

   profile - Enter name of load procedure from the archive screen (for example, sample_load).

   cpu_limit - Accept the default (1).

   memory_limit - Accept the default (1).

   max_connection_per_user- Accept the default (1).

4. Once finished, click the SAVE AND RESTART button to update and restart the server. You have completed the configuration for the XML XFOCUS archive.

APPENDIX A

XA Support

Topics:

This topic describes XA Transaction Management and implementation specifics.

XA Transaction Management

An application usually defines steps performed against any given resource in terms of *transactions*.

A *transaction* is a complete unit of work in which:

- Results are either all committed or all rolled back.
- The shared resource is transformed from one valid state to another.
- Changes to shared resources do not become visible outside the transaction until the transaction commits.
- The changes of commit are able to survive system or media failure.

The server has the ability to access multiple data sources and to carry out heterogeneous joins across different data sources. The transaction that describes work done by more than one Resource Manager is called a Global or Distributed Transaction.

Most relational and some non-relational resources are maintained by their Resource Managers. The X/Open Distributed Transaction Processing (DTP) model is a software architecture that allows multiple application programs to access resources provided by multiple resource managers, and allows their work to be coordinated into global transactions.

Read/write applications accessing CICS and IMS transactions, relational data sources, as well as CICS-controlled VSAM data sets are able to perform transactions managed in XA-compliant mode. The Transaction Coordinator component uses a two-phase commit protocol for completion of customer transactions across different database management systems.

Using the XA Transaction Management feature does not require any changes in existing ODBC, JDBC, or WebFOCUS Maintain applications. To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web Console configuration functions. In Transaction Coordination Mode, all requests performed against the listed XA-compliant DBMSs will be processed as XA-distributed transactions, and completed in two-phase commit mode. This guarantees the integrity of data modification on all of the involved DBMSs. It protects part of the data modifications from being committed on one DBMS and terminated on another.

Supported Interfaces

XA-compliant adapters are:

- DB2 (using CLI interface) on UNIX/Windows/Linux
- DB2 (using CLI Interface with RRS) on OS/390 and z/OS*
- Oracle
- Microsoft SQL Server (using OLE DB interface)
- Informix
- Sybase ASE
- Ingres II*

Non-XA-compliant adapters are:

- Adabas Stored Procedures*
- CICS Transactions*
- IMS Transactions*
- VSAM CICS*

Implementation

In order to adhere to the DTP model, the server-implemented Transaction Coordinator provides the ability to manage global transactions. Although the Transaction Coordinator is a private implementation, it fully adheres to the XA protocol in the boundaries of its implementation.

- Because the Transaction Coordinator is a private implementation, it does not have a public interface. There is no mechanism for foreign applications to use the Transaction Coordinator.
- Not every Resource Manager, even XA-compliant, is supported unless explicitly stated.

The implementation specifics of the Transaction Coordinator are:

- It is invoked by a keyword in edaserve.cfg (transaction_coordination_mode=on).
- Every unit of work initiated inside any agent will be started as a global transaction, within this agent.
- Coordination will apply to all currently supported XA-compliant interfaces.
- It cannot be turned off.
- All XA-compliant relational resources will participate, enabled implicitly.

- All non-relational resources can be explicitly disabled from participation.
- All transactions will be in the scope of a global transaction. Each commit/rollback will subsequently begin a transaction.
- No CREATE/DROP TABLE commands in XA mode.

Vendor Specifics

The implementation specifics for DB2 with CLI on UNIX, Windows, and OS/390 and z/OS are:

- The DB2CLI.INI file must have an entry in the common section

  ```
  [COMMON]

  DISABLEMULTITHREAD=1
  ```

- Only the global file, DB2CLI.INI, can be used. It must reside in:
  - For UNIX/NT, installation home.
  - For OS/390 and z/OS, exported by DSNAOINI variable.

The implementation specifics for DB2 with CLI on OS/390 and z/OS are:

- Requires RRS
- RRSAF in DB2CLI.INI

The implementation specific of MS SQL Server is:

- Required DTS

Note: If the client and the serve are on different machines, DTS must be installed on both machines.

The implementation specifics of Sybase are:

- The Sybase XA configuration file is needed.
- No bulk load/fast load functionality in XA; no messages are produced; and insert will operate in normal mode.
- Users can create their XA configuration file and point to it in the profile.

Syntax: How to Create the XA Configuration File

```
ENGINE SQLSYB SET XACONFIGFILE path
```

where:

path

Should include file name.

APPENDIX B

Aggregate Awareness Support

Topics:

Aggregate Awareness substantially improves the efficiency of queries. In order to use this feature successfully, you need an understanding of the principles of relational adapter operation and knowledge of the RDBMS optimizer.

Relational Adapters and Aggregated SQL Queries

Relational adapters construct optimized SQL queries based on a WebFOCUS request and the hierarchical structure of a file. A relational adapter always attempts to construct an SQL query that delegates most of the work (such as record screening, joins and aggregation) to the underlying relational engine. The interface's ability to pass the join and all tests to the RDBMS is a prerequisite to constructing an aggregated SQL query.

If aggregation cannot be passed to the RDBMS, the trace file contains an explanation. Usually, the request or the file hierarchy can be corrected in such a way that aggregation can be passed to the RDBMS.

Aggregate Awareness in an RDBMS

How to:

Set Aggregate Awareness

Reference:

Usage Notes for Aggregate Awareness

When aggregation is delegated to the RDBMS, the RDBMS can perform additional optimization of the query by using a mechanism commonly called Aggregate Awareness. Aggregate Awareness is implemented in DB2, Teradata, Oracle, and some other RDBMSs.

The implementation is similar in most RDBMSs. Based on most common types of reports performed on a database, the database administrator creates one or more objects (named SUMMARY TABLES in DB2, JOIN INDEXES in Teradata, and MATERIALIZED VIEWS or SNAPSHOTS in Oracle) that are populated with pre-aggregated data. The size of the pre-aggregated object (summary table, join index, or snapshot) is usually much smaller than the combined size of the involved tables. The RDBMS optimizer evaluates the incoming queries and, if possible, reconstructs the incoming query so that the pre-aggregated data is used for forming the answer set. This significantly reduces both CPU time and I/O operations.

The aggregated SQL generated by the relational adapter is fully suitable for an RDBMS optimizer to use pre-aggregated data. However, DB2, Teradata, and Oracle optimizers have specific rules for when to use pre-aggregated data.

Syntax: How to Set Aggregate Awareness

```
SET AGGREGATE_AWARENESS {FRESHONLY|OLD_OK|OFF}
```

where:

`FRESHONLY`

Sets different values for the parameters associated with each RDBMS.

`OLD_OK`

Sets different values for the parameters associated with each RDBMS.

`OFF`

If no option is selected, the behavior of the target RDBMS is determined by the database configuration options.There is no default for this setting.

Reference: Usage Notes for Aggregate Awareness

Adapter-specific parameters for which values are set by the FRESHONLY and OLD_OK settings:

- For DB2, CURRENT REFRESH AGE and CURRENT QUERY OPTIMIZATION.
- For Oracle, QUERY_REWRITE_INTEGRITY and QUERY_REWRITE_ENABLED.

  These values can be set in the Server profile or as a part of an RPC using Direct Passthru.
- Teradata handles aggregate awareness internally.

RDBMS manuals should be reviewed for recommendations, but the most common checklist items are:

- Updating statistics on the pre-aggregated object and the master tables: (RUNSTATS in DB2, COLLECT STATISTICS in Teradata, ANALYZE TABLE in Oracle).
- Confirming compatibility of the generated SQL query and the query used for creating the pre-aggregated object. The most common items are compatibility of:
  - Aggregators (columns functions) and aggregation objects (column function arguments)
  - GROUP BY clauses
  - Joins

APPENDIX C

Cluster Join

Topics:

- Embedded Joins
- Embedded Join Master Files

This topic describes the Master File SEGSUF attribute, which enables you to create a heterogeneous cluster joins, for example, between data sources of different types.

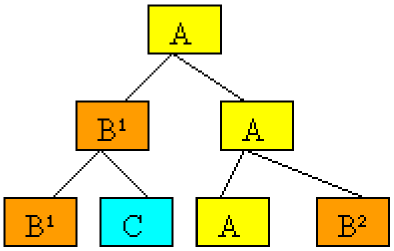
Embedded Joins

An embedded join is an equijoin that you define in a Master File. The name comes from the fact that the join definition is "embedded" in the Master File. (This is also known as a cluster join.)

You can define an embedded join between data sources:

- **Data sources of the same type.** For example, you can define an embedded join between two DB2 tables. This is known as a homogeneous embedded join.
- **Data sources of different types.** For example, you can define an embedded join between three DB2 tables, three CA-IDMS/DB records, and one IMS segment. This is known as a heterogeneous embedded join.

  In the following heterogeneous embedded join, A = a DB2 table, B = a CA-IDMS/DB record, and C = an IMS segment.



In a Master File, each segment declaration describes one RDBMS table or view, or one record. You can include up to 64 segments in an embedded join.

The entire join structure described by the Master File is referred to as a *tree*.

A *subtree* is a segment and any directly related descendant segments of the same data source type. A *root subtree* is the subtree at the top of the tree.

There are four subtrees here:

- A-A-A
- B^1-B^1
- B^2
- C

Note that B^1-B^1 and B^2 are separate subtrees because, even though they are of the same data source type, they are not directly related.

The root subtree is A-A-A.

Embedded Join Master Files

Each embedded join segment declaration describes one RDBMS table or view, or one non-RDBMS record. You can include up to 64 segments.

If several Master Files (used only with TABLE requests) include the same table, you can avoid repeating the same description multiple times. Describe the table in one of the Master Files, and use the CRFILE attribute in the other Master Files to access the existing description.

Syntax: How to Define an Equijoin in the Master File

An embedded equijoin uses the PARENT segment attribute to identify the relationship between tables and/or records.

```
FILENAME=mtname, SUFFIX=rootsuffix   [,$]
 SEGNAME=table1, SEGTYPE= {S0|KL} [,CRFILE=crfile1] [,$]
  FIELD=name,...,$
  .
  .
  .
 SEGNAME=table2, [SEGSUF=branchsuffix,] SEGTYPE=relationship,
  PARENT=table1 [,CRFILE=crfile2][,$]
   FIELD=name,...,$
  .
  .
  .
```

where:

mtname

Is the one- to eight-character name of the embedded join Master File.

rootsuffix

When the Master File describes:

- A homogeneous embedded join, *rootsuffix* specifies which adapter to use to access the data source.
- A heterogeneous embedded join, *rootsuffix* specifies which adapter to use to access the join's highest-level subtree. That is, it specifies which adapter to use to access the tables or records represented by the root segment and all directly-related descendant segments that are of the same DBMS type.

table1

Is the SEGNAME value for the root table or record in the embedded join. If this segment references a remote segment description, table1 must be identical to the SEGNAME from the Master File that contains the full definition of the remote data source.

`name`

Is any field name.

`table2`

Is the SEGNAME value for the related table or record. If this segment references a remote segment description, *table2* must be identical to the SEGNAME from the Master File that contains the full definition of the remote data source.

`SEGSUF`

Is used in a heterogeneous embedded join to identify the beginning of a new subtree (other than the root subtree), and to specify which adapter to use to access that subtree. A subtree is a segment (representing a table or record) and all directly related descendant segments of the same data source type. You specify SEGSUF in the subtree's highest-level segment.

`branchsuffix`

Specifies the value of SEGSUF.

`relationship`

Indicates the type of relationship between the table or record and its parent. Possible values are:

`S0` indicates that the related table is in a one-to-many or many-to-many (non-unique) relationship with the table or record named as its parent.

`U` indicates that the related table or record is in a one-to-one or a many-to-one (unique) relationship with the table or record named as its parent.

`KL` references a remote segment description. Indicates that the related table or record is in a one-to-many or many-to-many (non-unique) relationship with the table or record named as its parent.

`KLU` references a remote segment description. Indicates that the related table or record is in a one-to-one or a many-to-one (unique) relationship with the table or record named as its parent.

`crfile1`

References a remote segment description. Indicates the name of the Master File that contains the full definition of table1.

`crfile2`

References a remote segment description. Indicates the name of the Master File that contains the full definition of table2.

APPENDIX D

Translating COBOL File Descriptions

Topics:

This topic describes how synonyms are created from COBOL File Descriptions by the following adapters:

- C-ISAM
- Flat Files
- CICS Transactions
- IMS
- IMS Transactions
- Millennium
- VSAM and VSAM CICS

The information in this appendix is intended to supplement synonym-creation information in the individual adapter chapters.

Creating Synonyms From COBOL File Descriptions

Synonyms define unique names (or aliases) for each file that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server.

The CREATE SYNONYM command uses COBOL FD translation capabilities to translate the COBOL file definition into an internal metadata format from which a Master File is created. For adapter-specific information about synonym-creation, either from the Web Console or manually, see the following chapters:

- Chapter 6, *Using the Adapter for Informix C-ISAM, Micro Focus C-ISAM, FairCom c-tree ISAM, and Flat Files*
- Chapter 5, *Using the Adapter for CICS Transactions*
- Chapter 13, *Using the Adapter for IMS*
- Chapter 14, *Using the Adapter for IMS Transactions*
- Chapter 24, *Using the Adapter for Millennium*
- Chapter 51, *Using the Adapter for VSAM*

Controlling the Translation of a COBOL File Description

In this section:

You can take advantage of a variety of options to control the translation of a COBOL File Description to a Master File. These options are available from the Create Synonym panes in the adapters that use COBOL FDs. If customization options are not selected. default translation setting are applied.

This topic provides supplementary information for several customization options.

Customization Options for COBOL File Descriptions

You can customize how a COBOL FD is translated by selecting *Customize options* in the Synonym creation pane. The following options are added to the right pane:

Parameter	**Definition**
On Error	Choose *Continue* to continue generating the Master File when an error occurs. Choose *Abort* to stop generating the Master File when an error occurs. Choose *Comment* to produce a commented Master File when an error occurs. Continue is the default value.
Hyphens as	Choose *No* to remove all hyphens in the COBOL name from the Master File field names. Choose *Yes* to replace all hyphens in the COBOL name with the underscore character. *Yes* is the default value.
Redefines	You may treat COBOL REDEFINE fields in one of three ways: Choose *Segments* to describe REDEFINE fields as segments in the Master File. Segments is the default value. Choose *Comments* to describe REDEFINE fields as comments in the Master File. Choose *None* to exclude REDEFINE fields altogether.
Occurs as	Choose *Segments* to describe OCCURS structures as segments, or *Field* otherwise. Segments is the default value.
Alignment	Choose *Yes* to insert slack bytes into a record to ensure alignment of numeric fields. Choose *No* to generate Master Files without alignment of slack bytes. No is the default value.

Parameter	**Definition**
Number of Hyphens to skip	FD Translator removes characters from the left, up to and including the Nth hyphen (depending on the value of N chosen in the menu). A value of *0* will retain the entire COBOL name. *All* means all prefixes will be removed. *0* is the default value.
Order fields	Choose *Yes* to generate Order fields in a Master File. Choose *No* to generate a Master File without Order fields. *No* is the default value.
Level 88 as	Choose *Comment* to include COBOL Level 88 fields as comments in the Master Files. Choose *Skip* to exclude level 88 fields. *Skip* is the default value.
Zoned Numeric Fields	Sets how zoned numeric values will be stored.
Numeric Fields Edit Options	
Zeroes	Choose *Suppress* to suppress printing of the digit zero for a field whose value is zero. Choose *Display* to display leading zeroes, for example, 00124. Choose *None* for no formatting.
Negative value	Choose *Bracket* to bracket negative values, for example, (1234). Choose *Credit* to credit negative values, for example, 1234 CR. Choose *None* for no formatting.
Dollar Sign	Choose *Floating* to display a floating dollar sign and commas, for example, $1,1234. Choose *Fixed* to display a fixed dollar sign and commas, for example, $ 1,1234. Choose *None* for no formatting.

Parameter	Definition
Separate Thousands	Choose *Comma* to include commas where appropriate. Choose *None* for no formatting.

REDEFINE Fields

You may treat COBOL REDEFINE fields in one of three ways:

- Describe REDEFINEs as segments in the Master File (enter S for segments).
- Describe REDEFINEs as comments in the Master File (enter C for comments).
- Exclude REDEFINEs altogether (enter N for none).

Describing REDEFINEs as segments gives you immediate access to all first-level REDEFINEs but may still require user customization. Note that nested REDEFINEs are described as comments, even with this option. The Segment option is only available in software releases that support it. Including REDEFINEs as comments allows you to customize the Master File output to select the redefinitions of your choice.

Fields are described as comments with a dollar sign ($) in column one. If you wish to use the redefinition of a field instead of the original definition, delete the first definition, or insert dollar signs ($) in column one, and remove the dollar signs ($) from the redefinition.

Example: Generating REDEFINE Fields

Consider the following COBOL input description:

```
01  CLIENT-REC.
    02  NAME-ADDR-AREA   PIC X(57).
    02  NAME-AND-ADDR
        REDEFINES NAME-ADDR-AREA.
         03  NAME           PIC X(20).
         03  STREET         PIC X(15).
         03  CITY           PIC X(15).
         03  STATE          PIC X(2).
         03  ZIP            PIC X(5).
         03  ZIP_NUMERIC    REDEFINES ZIP PIC 9(5).
```

The following three Master Files are generated from this input, depending on the value entered for Generate REDEFINE Fields.

For a value of S:

```
FILE=REDEFS,                       SUFFIX=FIX,                          $
SEGNAME=CLIENSEG,     SEGTYPE=S0,                                       $
   GROUP=CLIENTREC,    ALIAS=E01,   USAGE=A57,      ACTUAL=A57,      $
   FIELD=NAMEADDRAREA, ALIAS=E02,   USAGE=A57,      ACTUAL=A57,      $
SEGNAME=NAMEASEG,     SEGTYPE=U,   PARENT=CLIENSEG,
OCCURS=1,             POSITION=NAMEADDRAREA,                            $
   GROUP=NAMEANDADDR,  ALIAS=E03,   USAGE=A57,      ACTUAL=A57,      $
   FIELD=NAME,         ALIAS=E04,   USAGE=A20,      ACTUAL=A20,      $
   FIELD=STREET,       ALIAS=E05,   USAGE=A15,      ACTUAL=A15,      $
   FIELD=CITY,         ALIAS=E06,   USAGE=A15,      ACTUAL=A15,      $
   FIELD=STATE,        ALIAS=E07,   USAGE=A2,       ACTUAL=A2,       $
   FIELD=ZIP,          ALIAS=E08,   USAGE=A5,       ACTUAL=A5,       $
$  FIELD=ZIP_NUMERIC,  ALIAS=E09,   USAGE=P5,       ACTUAL=Z5,       $
```

For a value of C:

```
FILE=REDEFC,                       SUFFIX=FIX,                          $
SEGNAME=CLIENSEG,     SEGTYPE=S0,                                       $
   GROUP=CLIENTREC,    ALIAS=E01,   USAGE=A57,      ACTUAL=A57,      $
   FIELD=NAMEADDRAREA, ALIAS=E02,   USAGE=A57,      ACTUAL=A57,      $
$ GROUP=NAMEANDADDR,   ALIAS=E03,   USAGE=A57,      ACTUAL=A57,      $
$   FIELD=NAME,        ALIAS=E04,   USAGE=A20,      ACTUAL=A20,      $
$   FIELD=STREET,      ALIAS=E05,   USAGE=A15,      ACTUAL=A15,      $
$   FIELD=CITY,        ALIAS=E06,   USAGE=A15,      ACTUAL=A15,      $
$   FIELD=STATE,       ALIAS=E07,   USAGE=A2,       ACTUAL=A2,       $
$   FIELD=ZIP,         ALIAS=E08,   USAGE=A5,       ACTUAL=A5,       $
$   FIELD=ZIP_NUMERIC, ALIAS=E09,   USAGE=P5,       ACTUAL=Z5,       $
```

For a value of N:

```
FILE=REDEFN,                       SUFFIX=FIX,                          $
SEGNAME=CLIENSEG,     SEGTYPE=S0,                                       $
   GROUP=CLIENTREC,    ALIAS=E01,   USAGE=A57,      ACTUAL=A57,      $
   FIELD=NAMEADDRAREA, ALIAS=E02,   USAGE=A57,      ACTUAL=A57,      $
```

LEVEL 88 as Comments

Level 88 fields in the COBOL FD associate particular values with a field name. Level 88 fields and values are not required in the Master File; however, you can include them as comments. Enter Y (for yes) to include them or N (for no) to exclude them. Including Level 88 fields lets you easily identify values for RECTYPEs for those files that require Level 88 fields. (You must manually add the RECTYPE and value. Because many Level 88 fields may occur in a COBOL FD, it is impossible to successfully determine the appropriate RECTYPE fields and values.) Since Level 88 field names typically describe the values they represent, including them as comments is also a way to document the Master File.

Example: Generating LEVEL 88 as Comments

Consider the following COBOL input description:

```
01  EMPL-REC.
    02  EMPL-SOC-SEC-NUM          PIC S9(8) COMP-3.
    02  EMPL-STATUS               PIC X(1).
        88  EMPL-ACTIVE                     VALUE 'A'.
        88  EMPL-INACTIVE                   VALUE 'I'.
        88  EMPL-RETIRED                    VALUE 'R'.
```

The following two Master Files are generated from this input, depending on the value entered for Generate LEVEL 88 as Comments.

For a value of Y:

```
FILE=LEV88Y,                       SUFFIX=FIX,                      $
SEGNAME=RECSEG,       SEGTYPE=S0,                                   $
   FIELD=SOCSECNUM,    ALIAS=E01,   USAGE=P9,      ACTUAL=P5,       $
   FIELD=STATUS,       ALIAS=E02,   USAGE=A1,      ACTUAL=A1,       $
$       ACTIVE, VALUE 'A'.                                          $
$       INACTIVE, VALUE 'I'.                                        $
$       RETIRED, VALUE 'R'.                                         $
```

The Level 88 fields are indented under the field they describe (STATUS). The values are left-justified to include the maximum information in the Master File, when many values are in the COBOL record. If the COBOL Level 88 values exceed one line in the FD, only the first line will be included in the Master File.

For a value of N:

```
FILE=LEV88N,                       SUFFIX=FIX,                      $
SEGNAME=RECSEG,       SEGTYPE=S0,                                   $
   FIELD=SOCSECNUM,    ALIAS=E01,   USAGE=P9,      ACTUAL=P5,       $
   FIELD=STATUS,       ALIAS=E02,   USAGE=A1,      ACTUAL=A1,       $
```

OCCURS as Segments

Example:

Simple Fixed OCCURS

Variable OCCURS

Nested OCCURS

COBOL FD translation supports all forms of COBOL OCCURS structures: fixed, variable, and nested. You may describe OCCURS structures as segments in the Master File or as individual, numbered fields. Enter Y (for yes) to describe OCCURS structures as segments or N (for no) otherwise.

Fixed and nested repeating groups described as segments use the OCCURS clause to describe the number of occurrences of the group. For fixed repeating groups, the OCCURS value is the number of occurrences. For variable OCCURS, the value is the name of the COBOL DEPENDING ON field. The POSITION attribute is used to describe their location in the record, which is reserved in the Master File with an internally generated POSITION field. All internal POSITION fields generated by the translation end with POSN.

Fixed repeating groups described as numbered fields have a unique number appended to the COBOL field name, once for each occurrence. Because the number of occurrences of a variably occurring group is unknown, they cannot be described individually. In fact, if any one of the repeating groups is variable, they are all described as segments, including fixed occurs, regardless of your menu selection.

Example: Simple Fixed OCCURS

Consider the following COBOL input description:

```
01  TBL-REC.
    02  TBL-ENTRY OCCURS 2 TIMES.
        03  TBL-AMT-A                   PIC S9(5).
        03  TBL-AMT-B                   PIC S9(5).
```

The translation can generate the following two Master Files from this input, depending on the value entered for Describe OCCURS as Segments.

For a value of Y:

```
FILE=OCCURSFY,                        SUFFIX=FIX,                     $
SEGNAME=TBLRESEG,     SEGTYPE=S0,                                     $
   GROUP=TBLREC,       ALIAS=E01,    USAGE=A20,       ACTUAL=A20,     $
   FIELD=TBLENSEGPOSN, ALIAS=E02,    USAGE=A20,       ACTUAL=A20,     $
SEGNAME=TBLENSEG,     SEGTYPE=S0,   PARENT=TBLRESEG,
OCCURS=2,             POSITION=TBLENSEGPOSN,                          $
   GROUP=TBLENTRY,     ALIAS=E03,    USAGE=A16,       ACTUAL=A10,     $
   FIELD=TBLAMTA,      ALIAS=E04,    USAGE=P6,        ACTUAL=Z5,      $
   FIELD=TBLAMTB,      ALIAS=E05,    USAGE=P6,        ACTUAL=Z5,      $
```

For a value of N:

```
FILE=OCCURSFN,                        SUFFIX=FIX,                     $
SEGNAME=TBLRESEG,     SEGTYPE=S0,                                     $
   GROUP=TBLREC,       ALIAS=E01,    USAGE=A32,       ACTUAL=A20,     $
   GROUP=TBLENTRY1,    ALIAS=E02,    USAGE=A16,       ACTUAL=A10,     $
   FIELD=TBLAMTA1,     ALIAS=E03,    USAGE=P6,        ACTUAL=Z5,      $
   FIELD=TBLAMTB1,     ALIAS=E04,    USAGE=P6,        ACTUAL=Z5,      $
   GROUP=TBLENTRY2,    ALIAS=E05,    USAGE=A16,       ACTUAL=A10,     $
   FIELD=TBLAMTA2,     ALIAS=E06,    USAGE=P6,        ACTUAL=Z5,      $
   FIELD=TBLAMTB2,     ALIAS=E07,    USAGE=P6,        ACTUAL=Z5,      $
```

Example: Variable OCCURS

Consider the following COBOL input description:

```
01  TABLE-REC.
    02  TABLE-COUNT            PIC S9(2) COMP.
    02  TABLE-ENTRY
        OCCURS 1 TO 10 TIMES
        DEPENDING ON TABLE-COUNT.
        03  TABLE-AMT-A        PIC S9(5).
        03  TABLE-AMT-B        PIC S9(5).
```

The translation generates the following Master Files from this input when either Y or N is entered for Describe OCCURS as Segments:

```
FILE=OCCURSVY,                       SUFFIX=FIX,                      $
SEGNAME=TABLESEG,      SEGTYPE=S0,                                    $
   FIELD=TABLECOUNT,    ALIAS=E01,   USAGE=I3,        ACTUAL=I2,      $
SEGNAME=TABLESE2,      SEGTYPE=S0,   PARENT=TABLESEG,
OCCURS=TABLECOUNT,                                                    $
   GROUP=TABLEENTRY,    ALIAS=E02,   USAGE=A16,       ACTUAL=A10,     $
   FIELD=TABLEAMTA,     ALIAS=E03,   USAGE=P6,        ACTUAL=Z5,      $
   FIELD=TABLEAMTB,     ALIAS=E04,   USAGE=P6,        ACTUAL=Z5,      $
```

The fields to be reported from (TABLEAMTA and TABLEAMTB) are within the OCCURS segment. The number of table occurrences is determined automatically by the TABLECOUNT field.

Example: Nested OCCURS

Consider the following COBOL input description:

```
01  TABLE-REC.
    02  TABLE-LEVEL-A OCCURS 2.
        03  TABLE-LEVEL-B OCCURS 3.
            04  TABLE-AMT              PIC S9(5) COMP-3.
```

The translation can generate the following two Master Files from this input, depending on the value entered for Describe OCCURS as Segments:

For a value of Y:

```
FILE=OCCURSNY,                         SUFFIX=FIX,                     $
SEGNAME=TABLESEG,      SEGTYPE=S0,                                     $
   GROUP=TABLEREC,      ALIAS=E01,    USAGE=A18,       ACTUAL=A18,     $
   FIELD=TABLESE2POSN, ALIAS=E02,    USAGE=A18,       ACTUAL=A18,     $
SEGNAME=TABLESE2,      SEGTYPE=S0,   PARENT=TABLESEG,
OCCURS=2,              POSITION=TABLESE2POSN,                          $
   GROUP=TABLELEVELA,  ALIAS=E03,    USAGE=A9,        ACTUAL=A9,      $
   FIELD=TABLESE3POSN, ALIAS=E04,    USAGE=A9,        ACTUAL=A9,      $
SEGNAME=TABLESE3,      SEGTYPE=S0,   PARENT=TABLESE2,
OCCURS=3,              POSITION=TABLESE3POSN,                          $
   GROUP=TABLELEVELB,  ALIAS=E05,    USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT,      ALIAS=E06,    USAGE=P6,        ACTUAL=P3,      $
```

The translation defines the nested table structure with nested segments. You need only refer to the final reporting field, TABLEAMT.

For a value of N:

```
FILE=OCCURSNN,                         SUFFIX=FIX,                     $
SEGNAME=TABLESEG,      SEGTYPE=S0,                                     $
   GROUP=TABLEREC,      ALIAS=E01,    USAGE=A48,       ACTUAL=A18,     $
   GROUP=TABLELEVELA1, ALIAS=E02,    USAGE=A24,       ACTUAL=A9,      $
   GROUP=TABLELEVELB1, ALIAS=E03,    USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT11,    ALIAS=E04,    USAGE=P6,        ACTUAL=P3,      $
   GROUP=TABLELEVELB2, ALIAS=E05,    USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT12,    ALIAS=E06,    USAGE=P6,        ACTUAL=P3,      $
   GROUP=TABLELEVELB3, ALIAS=E07,    USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT13,    ALIAS=E08,    USAGE=P6,        ACTUAL=P3,      $
   GROUP=TABLELEVELA2, ALIAS=E09,    USAGE=A24,       ACTUAL=A9,      $
   GROUP=TABLELEVELB4, ALIAS=E10,    USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT21,    ALIAS=E11,    USAGE=P6,        ACTUAL=P3,      $
   GROUP=TABLELEVELB5, ALIAS=E12,    USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT22,    ALIAS=E13,    USAGE=P6,        ACTUAL=P3,      $
   GROUP=TABLELEVELB6, ALIAS=E14,    USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT23,    ALIAS=E15,    USAGE=P6,        ACTUAL=P3,      $
```

ORDER Field

If you wish to select specific occurrences for reporting, add an ORDER field to the Master File as the last entry of the OCCURS segment:

```
FIELD=fieldname,    ALIAS=ORDER, USAGE=I4,       ACTUAL=I4, $
```

The value for ALIAS must be ORDER, the value for ACTUAL must be I4, the field name is arbitrary, and the value for USAGE must be an integer (I) but can be of any length from 1 to 9. The ORDER field is automatically populated.

Example: ORDER Field Sequence With Nested OCCURS

The following example adds the ORDER field SEQUENCE to the nested fixed OCCURS segments 2 and 3:

```
FILE=OCCURSNY,                         SUFFIX=FIX,                        $
SEGNAME=TABLESEG,      SEGTYPE=S0,                                        $
   GROUP=TABLEREC,      ALIAS=E01,   USAGE=A18,      ACTUAL=A18,      $
   FIELD=TABLESE2POSN, ALIAS=E02,   USAGE=A18,      ACTUAL=A18,      $
SEGNAME=TABLESE2,      SEGTYPE=S0,  PARENT=TABLESEG,
OCCURS=2,              POSITION=TABLESE2POSN,                             $
   GROUP=TABLELEVELA,  ALIAS=E03,   USAGE=A9,       ACTUAL=A9,       $
   FIELD=TABLESE3POSN, ALIAS=E04,   USAGE=A9,       ACTUAL=A9,       $
   FIELD=SEQUENCEA,    ALIAS=ORDER, USAGE=I4,       ACTUAL=I4,       $
SEGNAME=TABLESE3,      SEGTYPE=S0,  PARENT=TABLESE2,
OCCURS=3,              POSITION=TABLESE3POSN,                             $
   GROUP=TABLELEVELB,  ALIAS=E05,   USAGE=A8,       ACTUAL=A3,       $
   FIELD=TABLEAMT,     ALIAS=E06,   USAGE=P6,       ACTUAL=P3,       $
   FIELD=SEQUENCEB,    ALIAS=ORDER, USAGE=I4,       ACTUAL=I4,       $
```

Zoned Numeric Field Usage

A COBOL FD may describe alphanumeric characters as zoned decimal numeric data using the PICTURE 9(n) clause. This clause is commonly used for numeric string data on which no arithmetic operations is performed, such as phone numbers, identification numbers, and dates. You can describe these fields in packed numeric or alphanumeric USAGE format (enter the values P or A, respectively). The ACTUAL format is described as zoned (Z).

If you intend to use COBOL numeric display fields for summing or in mathematical operations, use the packed option. If you intend to use these elements for display only, use the alphanumeric option. Only simple numeric display elements of the format PICTURE 9(n) are subject to this feature. Signed elements (PICTURE S9(n)) and elements with decimal positions (PICTURE 9(n)V(m)) are always described as packed.

Example: Zoned Numeric Field Usage

Consider the following COBOL input description:

```
01 EMPL-REC.
      02 EMPL-SOC-SEC-NUM          PIC 9(8).
      02 EMPL-HIRE-DATE.
         03 EMPL-HIRE-YEAR         PIC 9(2).
         03 EMPL-HIRE-MONTH        PIC 9(2).
         03 EMPL-HIRE-DAY          PIC 9(2).
      02 EMPL-SICK-DAYS-ALLOWED    PIC S9(2).
      02 EMPL-SICK-DAYS-TAKEN      PIC 9(2).
      02 EMPL-YTD-HOURS-WORKED     PIC 9(4)V9(2).
```

The translation can generate the following two Master Files from this input, depending on the value entered for Zoned Numeric Field Usage.

For a value of P:

```
FILE=ZONEDP,                       SUFFIX=FIX,                        $
SEGNAME=RECSEG,        SEGTYPE=S0,                                     $
   GROUP=REC,           ALIAS=E01,   USAGE=A56,      ACTUAL=A24,       $
   FIELD=SOCSECNUM,     ALIAS=E02,   USAGE=P8,       ACTUAL=Z8,        $
   GROUP=HIREDATE,      ALIAS=E03,   USAGE=A24,      ACTUAL=A6,        $
   FIELD=HIREYEAR,      ALIAS=E04,   USAGE=P2,       ACTUAL=Z2,        $
   FIELD=HIREMONTH,     ALIAS=E05,   USAGE=P2,       ACTUAL=Z2,        $
   FIELD=HIREDAY,       ALIAS=E06,   USAGE=P2,       ACTUAL=Z2,        $
   FIELD=SICKDAYSALLO,  ALIAS=E07,   USAGE=P3,       ACTUAL=Z2,        $
   FIELD=SICKDAYSTAKE,  ALIAS=E08,   USAGE=P2,       ACTUAL=Z2,        $
   FIELD=YTDHOURSWORK,  ALIAS=E09,   USAGE=P7.2,     ACTUAL=Z6,        $
```

For a value of A:

```
FILE=ZONEDA,                       SUFFIX=FIX,                       $
SEGNAME=RECSEG,       SEGTYPE=S0,                                    $
   GROUP=REC,          ALIAS=E01,   USAGE=A32,      ACTUAL=A24,      $
   FIELD=SOCSECNUM,    ALIAS=E02,   USAGE=A8,       ACTUAL=Z8,       $
   GROUP=HIREDATE,     ALIAS=E03,   USAGE=A6,       ACTUAL=A6,       $
   FIELD=HIREYEAR,     ALIAS=E04,   USAGE=A2,       ACTUAL=Z2,       $
   FIELD=HIREMONTH,    ALIAS=E05,   USAGE=A2,       ACTUAL=Z2,       $
   FIELD=HIREDAY,      ALIAS=E06,   USAGE=A2,       ACTUAL=Z2,       $
   FIELD=SICKDAYSALLO, ALIAS=E07,   USAGE=P3,       ACTUAL=Z2,       $
   FIELD=SICKDAYSTAKE, ALIAS=E08,   USAGE=A2,       ACTUAL=Z2,       $
   FIELD=YTDHOURSWORK, ALIAS=E09,   USAGE=P7.2,     ACTUAL=Z6,       $
```

The two generated Master Files contains zoned decimal fields that are truly used in numeric and alphanumeric formats, but the translation cannot distinguish between the two cases. You must select the option most appropriate to your situation. You may edit the resulting Master File to change the formats to packed or alphanumeric as necessary. The USAGE length of any GROUP fields that contain these zoned fields must also be changed, according to the format conversion rules in *General Format Conversion Notes* on page D-18.

Numeric Field Edit Options

Example:

Numeric Field Edit Options

The COBOL FD translation automatically adds any edit options that you supply to all numeric fields in the generated Master File. Edit options affect how the numeric data is displayed. The options are as follows:

S - Suppresses printing of the digit zero for a field whose value is zero.

C - Includes commas where appropriate, for example, 1,234.

B - Brackets negative values, for example, (1234).

R - Credits negative values, for example, 1234 CR.

M - Displays a floating dollar sign with commas, for example, $1,234.

N - Displays a fixed dollar sign with commas, for example, $ 1,234.

L - Displays leading zeroes, for example, 001234.

You may enter up to five edit options. You may leave this option blank if you do not wish to edit numeric fields. Edit options may be entered in any order and combination except for the pairs SL, MN, and BR, which are mutually exclusive.

Example: Numeric Field Edit Options

Consider the following COBOL input description:

```
01  PAYROLL-REC.
      05  PAYROLL-NAME.
          10  PAYROLL-LAST-NAME   PIC X(20).
          10  PAYROLL-FIRST-NAME  PIC X(10).
      05  PAYROLL-AMT-FIELDS.
          10 PAYROLL-AMT-CURR-YR  PIC S9(7)V99 COMP-3.
          10 PAYROLL-AMT-PREV-YR  PIC S9(7)V99 COMP-3.
```

The translation generates the following Master Files from this input when MB is entered for Numeric Field Edit Options:

```
FILE=EDITOPT,                      SUFFIX=FIX,                      $
SEGNAME=RECSEG,       SEGTYPE=S0,                                   $
   GROUP=REC,          ALIAS=E01,   USAGE=A46,      ACTUAL=A40,     $
   GROUP=NAME,         ALIAS=E02,   USAGE=A30,      ACTUAL=A30,     $
   FIELD=LASTNAME,     ALIAS=E03,   USAGE=A20,      ACTUAL=A20,     $
   FIELD=FIRSTNAME,    ALIAS=E04,   USAGE=A10,      ACTUAL=A10,     $
   GROUP=AMTFIELDS,    ALIAS=E05,   USAGE=A16,      ACTUAL=A10,     $
   FIELD=AMTCURRYR,    ALIAS=E06,   USAGE=P11.2MB,  ACTUAL=P5,      $
   FIELD=AMTPREVYR,    ALIAS=E07,   USAGE=P11.2MB,  ACTUAL=P5,      $
```

Field Name Information

The translation generates field names in the Master File from COBOL field names using the following:

1. The Skip 'n' Hyphens option removes characters from the left, up to and including the nth hyphen (depending on the value of n you entered on the menu). It then either removes all remaining hyphens or replaces them with underscores, depending on the setting of the menu option Remove Hyphens or Use Underbars. Finally, if the Field Name Length option has been set to 12, only the 12 leftmost characters are used.

2. If the name duplicates a previously-generated field name, a qualifier is added to make it unique. For example, given the following COBOL structure

   ```
   02  EMPL-PORTION.
       03  SOC-SEC-NUM  PIC 9(9).
   02  MGR-PORTION.
       03  SOC-SEC-NUM  PIC 9(9).
   ```

   SOCSECNUM is generated from the first elementary field name and SOCSECNUM2 from the second elementary field name.

The following table illustrates the results of the possible combinations of menu options that control field name generation.

COBOL Field Name: PAYROLL-REC-IN-FICA			
Field Name Length	**Remove Hyphens or Use Underbars**	**Skip 'n' Hyphens**	**Generated Field Name**
`12`	`Remove Hyphens`	`0`	`PAYROLLRECIN`
		`1`	`RECINFICA`
		`2`	`INFICA`
		`3`	`FICA`
	`Use Underbars`	`0`	`PAYROLL_REC`
		`1`	`REC_IN_FICA`
		`2`	`IN_FICA`
		`3`	`FICA`
`30`	`Remove Hyphens`	`0`	`PAYROLLRECINFICA`
		`1`	`RECINFICA`
		`2`	`INFICA`
		`3`	`FICA`
	`Use Underbars`	`0`	`PAYROLL_REC_IN_FICA`
		`1`	`REC_IN_FICA`
		`2`	`IN_FICA`
		`3`	`FICA`

General Format Conversion Notes

Format conversion is the process of defining the ACTUAL and USAGE formats based on the COBOL format. The translation performs the conversion as described in the following table:

COBOL Format	ACTUAL	USAGE
`PICTURE  X(n)`	`An`	`An`
`PICTURE  9(n)`	`Zn (packed option)`	`Pn (packed option)`
`PICTURE  9(n)`	`An (alpha option)`	`An (alpha option)`
`PICTURE S9(n)`	`Zn`	`Pn+1`
`PICTURE  9(n)V9(m)`	`Zn+m`	`Pn+m+1.m`
`PICTURE S9(n)V9(m)`	`Zn+m`	`Pn+m+2.m`
`PICTURE  9(n) COMP (1 # n # 4)`	`I2`	`I9`
`PICTURE  9(n) COMP (5 # n # 9)`	`I4`	`I9`
`PICTURE  9(n) COMP (n > 9)`	`A8`	`A8`
`COMP-1`	`F4`	`F8`
`COMP-2`	`D8`	`D15`
`PICTURE  9(n)      COMP-3`	`P(n+2/2)`	`Pn`
`PICTURE S9(n)      COMP-3`	`P(n+2/2)`	`Pn+1`
`PICTURE  9(n)V9(m) COMP-3`	`P(n+m+2/2)`	`Pn+m+1.m`
`PICTURE S9(n)V9(m) COMP-3`	`P(n+m+2/2)`	`Pn+m+2.m`

The format conversions are subject to the following limitations:

- Alphanumeric fields are limited to 256 characters. Elementary fields that exceed 256 characters are truncated to 256 characters and have filler fields inserted to provide for the total length. Group fields that exceed 256 characters are commented.
- Unsigned zoned fields that exclude a decimal point are described with a USAGE of packed (P) or alphanumeric (A), depending on the value entered for the Zoned Numeric Field Usage option.
- COMP-4 binary fields are described the same as COMP fields.
- Binary fields that exceed 9 digits in the picture clause are described with ACTUAL format A8 and USAGE format A8.

- The maximum length of the packed USAGE format depends on the software release that uses the generated Master File. For releases that support long packed fields (16 or more digits), the maximum ACTUAL length is 16 bytes and the maximum USAGE length is 31 digits (or 32 characters including a decimal point and sign). For earlier releases, the maximum ACTUAL length is 8 bytes and the maximum USAGE length is 15 total digits, including decimal and sign. The number of decimal places is limited to one fewer than the total USAGE length.
- In general, the USAGE length of packed fields is calculated as the sum of the digits to the left of the decimal (n), the number of decimal positions (m), one for the leading minus sign if present (S), and one for the decimal (V) if present.
- The ACTUAL and USAGE formats for GROUP fields are always alphanumeric (A). The ACTUAL length is the sum of the ACTUAL lengths of its components. The USAGE length is the sum of the following:
- The USAGE lengths of all the alphanumeric (A) components.
- 16 for each long packed (P) component.
- 8 for each short packed (P) and double precision (D) component.
- 4 for each integer (I) and floating point (F) component.

Multiple Records as Input

The translation generates a new segment for each COBOL record description (01-level field) it receives as input. When multiple records are present, an additional DUMMY segment is created that is used as the common parent for the record descriptions. Whenever you submit more than one 01-level record as input, you must manually add RECTYPE fields to identify the different record types.

Example: Multiple Records as Input

Consider the following COBOL input description:

```
01  HDR-REC.
       02  HDR-KEY             PIC X(1).
           88 HDR-VALUE        VALUE 'H'.
       02  HDR-DATA            PIC X(10).
01  DET-REC.
       02  DET-KEY             PIC X(1).
           88 DTL-VALUE        VALUE 'D'.
       02  DTL-DATA            PIC X(10).
```

From this input, the following Master File with multiple segments is created:

```
FILE=MULT01,                       SUFFIX=FIX,                      $
SEGNAME=DUMMY,          SEGTYPE=S0,                                 $
   FIELD=,               ALIAS=,     USAGE=A1,       ACTUAL=A1,     $
SEGNAME=HDRRESEG,       SEGTYPE=S0,  PARENT=DUMMY,                  $
   GROUP=HDRREC,         ALIAS=E01,  USAGE=A11,      ACTUAL=A11,    $
   FIELD=HDRKEY,         ALIAS=E02,  USAGE=A1,       ACTUAL=A1,     $
   $      HDRVALUE, VALUE 'H'.                                      $
   FIELD=HDRDATA,        ALIAS=E03,  USAGE=A10,      ACTUAL=A10,    $
SEGNAME=DETRESEG,       SEGTYPE=S0,  PARENT=DUMMY,                  $
   GROUP=DETREC,         ALIAS=E04,  USAGE=A11,      ACTUAL=A11,    $
   FIELD=DETKEY,         ALIAS=E05,  USAGE=A1,       ACTUAL=A1,     $
   $      DTLVALUE, VALUE 'D'.                                      $
   FIELD=DTLDATA,        ALIAS=E06,  USAGE=A10,      ACTUAL=A10,    $
```

After editing to add the RECTYPE information, the completed Master File is:

```
FILE=MULT01,                       SUFFIX=FIX,                      $
SEGNAME=DUMMY,          SEGTYPE=S0,                                 $
   FIELD=,               ALIAS=,     USAGE=A1,       ACTUAL=A1,     $
SEGNAME=HDRRESEG,       SEGTYPE=S0,  PARENT=DUMMY,                  $
   GROUP=HDRREC,         ALIAS=E01,  USAGE=A11,      ACTUAL=A11,    $
   FIELD=RECTYPE,        ALIAS=H,    USAGE=A1,       ACTUAL=A1,     $
   $      HDRVALUE, VALUE 'H'.                                      $
   FIELD=HDRDATA,        ALIAS=E03,  USAGE=A10,      ACTUAL=A10,    $
SEGNAME=DETRESEG,       SEGTYPE=S0,  PARENT=DUMMY,                  $
   GROUP=DETREC,         ALIAS=E04,  USAGE=A11,      ACTUAL=A11,    $
   FIELD=RECTYPE,        ALIAS=D,    USAGE=A1,       ACTUAL=A1,     $
   $      DTLVALUE, VALUE 'D'.                                      $
   FIELD=DTLDATA,        ALIAS=E06,  USAGE=A10,      ACTUAL=A10,    $
```

Maximum Number of Fields

Each 01-level record input can contain up to 4,000 fields. Any number of 01-level entries can be used as input to the translation, as long as no individual record exceeds this limit. Each occurrence of an OCCURS structure counts in this limit. If the limit is exceeded, an error message is printed and program execution stops. Note that this limitation applies to the translation and not the Master File generated.

Example: Maximum Number of Fields

Consider the following COBOL input description:

```
01  REC1.
    02  FLD1  OCCURS  5000  TIMES  PIC  X(1).
```

This input generates the following message:

```
FR011E - INTERNAL TABLE OVERFLOW
```

To bypass this limitation, edit the COBOL FD and temporarily reduce the total number of OCCURS. After execution, edit the Master File and restore the original number of occurrences.

Consider the following COBOL input description:

```
01  REC1.
    02  FLD1  OCCURS  2000  TIMES  PIC  X(1).
01  REC2.
    02  FLD2  OCCURS  2000  TIMES  PIC  X(1).
```

The following Master File, with multiple segments, is created from this input:

```
FILE=MAXRECS,                     SUFFIX=FIX,                      $
SEGNAME=DUMMY,          SEGTYPE=S0,                                $
   FIELD=,               ALIAS=,      USAGE=A1,       ACTUAL=A1,       $
SEGNAME=REC1SEG,        SEGTYPE=S0,  PARENT=DUMMY,                 $
$  GROUP=REC1,           ALIAS=E01,   USAGE=A2000,    ACTUAL=A2000,    $
   FIELD=FLD1SEGPOSN,    ALIAS=E02,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E03,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E04,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E05,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E06,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E07,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E08,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E09,   USAGE=A208,     ACTUAL=A208,     $
SEGNAME=FLD1SEG,        SEGTYPE=S0,  PARENT=REC1SEG,
OCCURS=2000,            POSITION=FLD1SEGPOSN,                      $
   FIELD=FLD1,           ALIAS=E10,   USAGE=A1,       ACTUAL=A1,       $
SEGNAME=REC2SEG,        SEGTYPE=S0,  PARENT=DUMMY,                 $
$  GROUP=REC2,           ALIAS=E11,   USAGE=A2000,    ACTUAL=A2000,    $
   FIELD=FLD2SEGPOSN,    ALIAS=E12,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E13,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E14,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E15,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E16,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E17,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E18,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,         ALIAS=E19,   USAGE=A208,     ACTUAL=A208,     $
SEGNAME=FLD2SEG,        SEGTYPE=S0,  PARENT=REC2SEG,
OCCURS=2000,            POSITION=FLD2SEGPOSN,                      $
   FIELD=FLD2,           ALIAS=E20,   USAGE=A1,       ACTUAL=A1,       $
```

This description does not exceed the 4,000 field limit because each 01-level has fewer than 4,000 fields.

Year 2000

Because a COBOL FD cannot describe fields as dates, the translation cannot reliably determine which fields are dates and apply date formats to them. However, you can edit Master Files generated during synonym creation to add date formats and date defaults. These are respected in Information Builders software designed to take advantage of them.

Reference: COBOL FD Syntax Requirements

The COBOL FD translation requires a syntactically correct COBOL file description as input. While scanning the COBOL file description during synonym creation, a number of syntax checks are performed. If any of the COBOL statements are invalid, debugging messages are displayed, the program stops executing, and a Master File is not generated. Although the syntax check can identify a variety of errors, you should specify only valid COBOL file descriptions that compile successfully.

At a minimum, a valid COBOL file description must follow these guidelines:

- Level-numbers must be a one- or two-digit integer. Values can be 1 through 49, 66, 77, or 88.
- Level-numbers 01 and 77 must begin in Columns 8 through 11; all other level-numbers may begin in Columns 8 through 72.
- Comment lines are identified with an asterisk (*) in Column 7.
- Continuation lines are identified with a hyphen (-) in Column 7.
- The COBOL file description starts with a level-number 01 column and ends with the last column described.
- The COBOL field names are in upper case characters.
- Other COBOL instructions are not permitted as input.
- Embedded tabs and blank lines are not permitted as input.

Index

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:

Reader Comments