Advantage™ VISION:Builder®

Advantage™ VISION:Two™ for z/OS

Getting Started

r15

Computer Associates®

MAN08092830E
B02628-1E

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2005 Computer Associates International, Inc.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1: Introducing Advantage VISION:Builder

Chapter 2: Advantage VISION:Builder Quick Reference

Chapter 3: Using COBOL Quick Start

Chapter 4: Using DB2 Quick Start

Chapter 5: Using Advantage VISION:Results Quick Start

Chapter 6: Using Advantage VISION:Inquiry Quick Start

Appendix A: ASL Examples

Index

Chapter 1

Introducing Advantage VISION:Builder

Advantage VISION:Builder

Note: Throughout this guide, the symbol ❹ is used to designate that the feature or function being described only applies to the full Advantage™ VISION:Builder® product, and not Advantage™ VISION:Two™.

Advantage VISION:Builder is a software system you can use to design, implement, and execute data processing applications. Novice users can be productive within a few hours, yet Advantage VISION:Builder produces complex applications efficiently. An experienced user can build an application in Advantage VISION:Builder in less than half the time it would take for a similar application in COBOL or other third-generation languages.

Advantage VISION:Builder is the full function product, while Advantage VISION:Two is a subset of Advantage VISION:Builder that restricts the ability to create new databases and to update existing databases using transaction files.

The philosophy in using Advantage VISION:Builder is to:

- Conceptualize how Advantage VISION:Builder functions during input, processing, and output.
- Design applications that use as many of the automatic functions of Advantage VISION:Builder as possible.

Advantage VISION:Builder provides extensive facilities for the creation and maintenance of files and databases, as well as report generation. Advantage VISION:Builder processes virtually any data source and database format available on an IBM host (for example, flat files, IMS, and DB2) without requiring the programmer to be familiar with these formats.

The Advantage VISION:Builder system is made up of the following main components:

- Advantage VISION:Builder Engine
- Advantage VISION:Builder Workbench for DOS
- Advantage VISION:Builder Workbench for ISPF
- COMLIB Library Utilities.

For more information about these components, see the *Advantage VISION:Builder for z/OS Reference Guide*.

Features

This section describes important features and benefits of Advantage VISION:Builder.

Efficient Database Access

With Advantage VISION:Builder, you can generate a virtually unlimited number of reports in a single pass of multiple databases. Other products require a separate pass of each database for each report, which significantly increase the read/write cycles and runtime.

Advantage VISION:Builder is engineered to access all IBM enterprise server databases and files automatically, simultaneously, and concurrently for maximum efficiency.

Advantage VISION:Builder frees application developers from the complexities of various database management systems, providing a logical view of data separate from the physical organization. Users preparing their applications are not required to code any calls to DB2 , IMS/DB, or VSAM database management systems. After data definitions are prepared and referenced in the program, Advantage VISION:Builder automatically manages all the input and output operations for the various databases. This gives you the staffing flexibility you need for application management projects.

Wide Variety of Report Output Formats

With Advantage VISION:Builder, you can view reports in a wide variety of formats. In addition to the traditional report formats, VISION:Builder automatically outputs report contents in HTML, TXT, or CSV.

Migrating Data

Advantage VISION:Builder can select data from existing operational data files and external data sources; and transform, summarize, and enhance the data prior to loading into a data warehouse. Advantage VISION:Builder can build the initial warehouse and be used on a continuing basis for refreshing the warehouse with updated data.

Meta Data Repository System

The meta data repository system (COMLIB Library Utilities) lets you create file definitions, table definitions, and transaction update definitions once, and then store them in the Advantage VISION:Builder library where they can be used for populating and updating data warehouses or by any other program. The library lets you store and reuse Advantage VISION:Builder code, so that commonly used procedures can be employed over and over again in different applications.

Verifying the Data

Advantage VISION:Builder offers complete transaction processing, as well as extensive editing and automated error-checking for file, database, and data warehouse updates. You can dynamically modify transactions before applying them as updates and test or modify the resulting record. Advantage VISION:Builder automatically creates an audit trail.

Handling Complex Applications Easily

Advantage VISION:Builder is designed for development organizations with the most challenging enterprise server application requirements. Many Advantage VISION:Builder applications in use today involve databases with millions of records and tables with millions of rows.

Advantage VISION:Builder helps both novice and experienced developers easily solve complex problems by automating more than 2,000 development functions. Powerful programs that meet a wide variety of objectives can be easily and quickly generated using the defaults and automatic functions of Advantage VISION:Builder.

Benefits

Advantage VISION:Builder provides you with the following benefits:

- Automates database conversions necessary for populating and maintaining data warehouses
- Provides for more efficient use of corporate data with concurrent access to multiple databases
- Simplifies migration by automating database conversions
- Minimizes data handling, optimizes runtime, and reduces storage overhead by taking advantage of summarized data files

- Automates more than 2,000 development functions to simplify development and database management for even the most complex applications
- Reduces enterprise server overhead by making database access more efficient
- Provides an efficient and easy-to-use utility to organize corporate data for application management projects

Advantage VISION:Builder Requirements

The following hardware and platforms are required for Advantage VISION:Builder:

- IBM and IBM-compatible enterprise servers — z/OS, VSE/ESA, and VM/ESA.
- VSAM support with options for supporting IMS, DB2, and ISV database managers concurrently.
- Optional PC and ISPF program development through Advantage VISION:Builder Workbench.

Environment

This section describes the packaging and environmental considerations for Advantage VISION:Builder r15.

IBM Operating Systems Supported

Separate versions of Advantage VISION:Builder exist to support the various IBM operating systems in use at an installation. The z/OS version supports all z/OS operating systems. The VSE version supports VSE/ESA. The CMS version supports VM/ESA.

Available Interfaces

You can use the following interfaces with Advantage VISION:Builder:

- Advantage VISION:Builder Workbench for DOS
- Advantage VISION:Builder Workbench for ISPF
- DB2 SQL/DS Interface
- Advantage VISION:Builder IMS Interface
- Generalized Data Base Interface
- Call other programs

SMP/E Installation Specifications

The packaging of Advantage VISION:Builder conforms to the IBM SMP/E standards. SMP/E modification control statements (MCS) along with the supporting JCL for the RECEIVE, APPLY, and ACCEPT processes are used to install the elements of the product. The Indirect File method is used to reference the unloaded product data sets during the install process. A REXX program is available to assist you in tailoring the SMP/E install job streams (JCL and control statements) to conform to your standards and conventions. Local product tailoring procedures (M4PARAMS, and so on) are performed in a manner similar to that used in previous releases of Advantage VISION:Builder. Problem fixes or post-delivery enhancements to the product are packaged to conform to SMP/E standards and processes for PTFs and APARs.

The Function Modifier Identification (FMID) for Advantage VISION:Builder r15 is CCVCF00. The FMID for Advantage VISION:Two r15 is CCVPF00.

CA LMP Licensing Specifications

To verify proper authorization to use the product at a customer site, Advantage VISION:Builder r15 uses the CA License Management Program (LMP). The Advantage VISION:Builder product family is offered as two products in the marketplace, namely Advantage VISION:Builder and Advantage VISION:Two. Both of these products provide the following two database interfaces as optional features:

- DB2 Database Interface
- Generalized Database Interface

Depending on your licensed features, there is one product code for the base system, Advantage VISION:Builder or Advantage VISION:Two, and one code for each licensed optional feature.

The assigned product codes are:

SM	Advantage VISION:Builder
SZ	Advantage VISION:Builder Interface for GDBI
S6	VISION:Builder Interface for DB2
S8	Advantage VISION:Two
S9	Advantage VISION:Two Interface for GDBI
TG	Advantage VISION:Two Interface for DB2

Advantage VISION:Builder r15 invokes LMP to verify the authorized use of either the Advantage VISION:Builder or Advantage VISION:Two products. This verification occurs during product initialization. If the CAIRIM service is not operational, the product terminates with an appropriate error message.

Additionally, Advantage VISION:Builder r15 invokes LMP to verify the authorized use of the base product or any of the three optional database interfaces. If an attempt is made to employ the base product or any of the optional interfaces within an application without proper authorization, an appropriate error message is issued but execution of the product continues.

The Purpose of This Guide

This guide introduces you to Advantage VISION:Builder Advantage VISION:Two r15 for z/OS (hereinafter referred to as Advantage VISION:Builder). By the time you have finished reading this guide, you will have an overview of the wide scope of the product and its usability will be familiar to you. It is important to us that you feel comfortable with Advantage VISION:Builder before you begin to use it.

Documentation

The documentation is delivered on a compact disc (CD). The books are in Adobe Acrobat Portable Document Format (PDF) and are designed for you to read online using the Acrobat Reader (also included on the CD).

Online books may include a table of contents, index, and underlined colored hypertext links. To go directly to the book, chapter, section, or topic being referenced, click the hypertext link.

The following documentation is available for Advantage VISION:Builder r15 and Advantage VISION:Two r15 on the documentation CD:

Name	Description
Readme file	A file containing system and installation requirements, last minute known issues, and information on contacting technical support.
Advantage VISION:Builder r15 for z/OS Getting Started	Contains fundamental descriptions of Advantage VISION:Builder, along with sections on features, benefits, hardware requirements, and documentation.

Name	Description
Advantage VISION:Builder r15 for z/OS Release Summary	Contains a brief summary of the new enhancements for the current release of Advantage VISION:Builder, and enhancements to existing features.
Advantage VISION:Builder r15 for z/OS Installation Guide	Describes how to install Advantage VISION:Builder.
Advantage VISION:Builder r15 for z/OS Reference Guide	Describes all of the concepts necessary to understand how Advantage VISION:Builder works and how applications can be developed using Advantage VISION:Builder. It covers both basic and advanced applications. Also contains information specific to the use and operation of the toolkit routines delivered with Advantage VISION:Builder. These toolkit routines may be used within an Advantage VISION:Builder application to perform functions not readily available using native Advantage VISION:Builder programming operations. Also provides information specific to the use and operation of a routine which allows user programs written in other languages to access tables stored in a Common Library.

Name	Description
Advantage VISION:Builder r15 for z/OS ASL Reference Guide	Contains information specific to the use and operation of Advanced Syntax Language (ASL). (This guide assumes that you are familiar with Advantage VISION:Builder.) Also describes the conversion from fixed form syntax to ASL.
Advantage VISION:Builder r15 for z/OS Specifications Guide (previously named *Advantage VISION:Builder Reference Summary*)	Contains all valid fixed format syntax specifications for Advantage VISION:Builder statements.
Advantage VISION:Builder r15 for z/OS Messages Guide	Lists Advantage VISION:Builder messages and codes and gives detailed explanations. Also assists programmers in identifying and resolving problems that may occur during the execution of Advantage VISION:Builder.
Advantage VISION:Builder r15 for z/OS Environment Guide	Describes detailed information for the use of Advantage VISION:Builder and the Common Library subsystem in the z/OS environment.

Name	Description
Advantage VISION:Builder Workbench for DOS Reference Guide	Contains information specific to the operation and use of Advantage VISION:Builder Workbench for DOS. It is designed to give you a basic understanding of the features, capabilities, and flow of Advantage VISION:Builder Workbench for DOS.
Advantage VISION:Builder Workbench for ISPF Reference Guide	Contains a brief introduction to Advantage VISION:Builder Workbench for ISPF, an illustrated application, and an in-depth discussion of the panels in Advantage VISION:Builder Workbench for ISPF.

The following guides only apply if you have the GDBI feature. These books are available only by special request and are not available on the documentation CD or SupportConnect.

Name	Description
GDBI Programming Guide	Covers the concepts and operation of the Generalized Data Base Interface which enables you to access and process data from virtually any Database Management System without having to be concerned about where the data is stored or in what format it is stored.
Using GDBI with ADABAS	Aids you in the design and implementation of Advantage VISION:Builder applications which access Software AG's Adaptable Data Base System (ADABAS) files using the GDBI option of Advantage VISION:Builder.

Name	Description
Using GDBI with SUPRA	Aids you in the design and implementation of Advantage VISION:Builder applications which access Cincom's SUPRA databases using the GDBI option of VISION:Builder.
Using GDBI with IDMS	Aids you in the design and implementation of Advantage VISION:Builder applications which access the Computer Associates Integrated Database Management System (IDMS) databases using the GDBI option of VISION:Builder.

CA Technology Services: Delivering Business Value On Your Terms

CA Technology Services is a global organization of highly trained, experienced professionals who are determined to provide you with the technical expertise you need, when and how you need it. From implementing a CA solution to helping you get the most out of the CA technology that you have, CA Technology Services is committed to delivering business value to you on **your** terms.

Our professionals understand your unique business needs and work closely with you to assess which technology is right for your business. Whether the assignment is large or small or you need a custom, stand-alone, or packaged solution, we tailor our efforts to meet your business demands.

By offering a broad range of flexible services, we help you maximize your investment in our technology, achieve more efficient IT performance, and better manage your infrastructure, security, storage, applications, and data. Such flexibility ensures that you reach your time-to-market goals while improving your business performance.

Why not ask your CA representative for more information about how a CA Technology Services professional can help your organization get the most out of your CA business solutions?

CA Education Services: Ready When You Are

The goal of CA Education Services is to help you realize the full potential of your CA software investment. To meet this goal, our high-quality instructors strive to understand your specific training requirements, and then deliver the right kind of training when, where, and how you need it.

All CA instructors are fully certified and offer a wealth of hands-on enterprise management experience gained in working with today's largest and most complex businesses. Whether your training is web-based, self-paced, or in the traditional classroom, you always receive the most up-to-date instruction and expertise that is available. The knowledge you gain through training prepares you to successfully leverage the capabilities of your CA software.

Why not ask your CA representative how our training and education programs can help you get more out of your CA business solutions?

Computer Associates: Commitment, Quality, Innovation

For more than a quarter century, CA has been developing and supporting software solutions that are currently used by more than 99 percent of the Fortune 500 companies in more than 100 countries. CA is committed to offering leading technologies in flexible partnerships to help you derive full value from your software investments.

At Computer Associates, we are committed to offering simple and meaningful solutions to your complex problems, and to delivering management solutions that offer security, reliability, availability, and performance. We work hard to achieve the highest levels of quality in our solutions to help you meet your changing business needs.

To meet these needs, CA's world-class solutions address all aspects of process management, information management, and infrastructure management with six focus areas:

- Enterprise management
- Security
- Storage
- Portal and business intelligence
- Database management
- Application life cycle management and application development

In addition, our innovative approach to technology is carried over into our innovative business solutions. From a revolutionary new business model to a dedicated customer relationship organization, CA is responding to your changing business needs.

We know what it takes to deliver and support valuable solutions 24 hours a day, 7 days a week, 365 days a year while maintaining the highest standards for quality and innovation:

- We are the first global enterprise software company to meet the exacting standards for worldwide ISO 9002 certification.
- We have earned over 150 patents for innovative software solutions.
- We have the highest caliber software developers and consultants in the industry.

We also know you expect us to stand by our commitments. And we do.

For More Information

After reading this *Getting Started*, you can refer to the numerous resources available to you for additional information.

Supportconnect.ca.com provides access to instructional documents that showcase your software and provide detailed explanations about the product's comprehensive, feature-rich components.

You can also obtain procedural information and answers to any questions from the supportconnect.ca.com site.

Chapter 2

Advantage VISION:Builder Quick Reference

This chapter lists some commonly used Advantage VISION:Builder applications, which are described in the other Advantage VISION:Builder product guides. This section describes where you can find information about those tasks and proposes several alternate solutions.

Use the online search function to locate the sections referenced in the following tables.

Specialized Report Formats

Task	Book	Chapter	Section
Report on preprinted forms.	Advantage VISION:Builder Reference Guide	Chapter 14	Formatted Reporting
Display data across the page instead of columnar.	Advantage VISION:Builder Reference Guide	Chapter 13	Reporting from Arrays
Generate labels.	Advantage VISION:Builder Reference Guide	Chapter 9	Report Page Layout
		Chapter 14	Formatted Reporting

Task	Book	Chapter	Section
Generate reports of non-standard sizes.	Advantage VISION:Builder Reference Guide	Chapter 14	Formatted Sectional Reporting Formatted Reporting Dynamic Report Line Modification
Display message on bottom of every page.	Advantage VISION:Builder Reference Guide	Chapter 14	Formatted Sectional Reporting

Reporting from Special Data Files

Task	Book	Chapter	Section
Summarize file data and change report information based on the use of summary data.	Advantage VISION:Builder Reference Guide	Chapter 14	Formatted Sectional Reporting
Report from multiple files.	Advantage VISION:Builder Reference Guide	Chapter 6	Coordinated Files

Task	Book	Chapter	Section
Report from multiple files when coordinating field is not contiguous.	Advantage VISION:Builder Reference Guide	Chapter 6	Coordinated Files
		Chapter 15	The Transaction Processing Step❹
Report failed transactions.❹	Advantage VISION:Builder Reference Guide	Chapter 15	Transaction Record Rejection and Type 4 Procedure Processing Flow
Report from header/trailer file.	Advantage VISION:Builder Reference Guide	Chapter 6	Coordinated Files
		Chapter 10	Transaction File and Master File Record Flows❹
		Chapter 14	Formatted Reporting
Report collating and routing to remote printers.	Advantage VISION:Builder Reference Guide	Chapter 17	Report Manager

Data Field and Record Processing

Task	Book	Chapter	Section
Sample Application.	Advantage VISION:Builder Reference Guide	Chapter 5	Sample Application Source Listing
Add fields to an existing file. ❹	Advantage VISION:Builder Reference Guide	Chapter 5	Application Cycle Overview
		Chapter 10	Transaction File and Master File Record Flows
Clear out invalid data from numeric fields.	Advantage VISION:Builder Reference Guide	Chapter 4	Invalid Fields
		Chapter 10	Transaction Definitions ❹
Perform calculations with time data.	Advantage VISION:Builder Reference Guide	Chapter 4	Time Data Conversions and Arithmetic Operations
Change the same field in every record on the file. Ⓑ	Advantage VISION:Builder Reference Guide	Chapter 6	Master File Processing Options
		Chapter 10	Transaction File and Master File Record Flows
Procedurally update fields.	Advantage VISION:Builder Reference Guide	Chapter 6	Master File Processing Options

Task	Book	Chapter	Section
Remove selected records from a file. ❹	Advantage VISION:Builder ASL Reference Guide	Appendix C	Flags
	Advantage VISION:Builder Reference Guide	Chapter 7	Deleted Master File Records
		Chapter 10	Transaction File and Master File Record Flows
	Advantage VISION:Builder Specifications Guide	Chapter 3	Flags

File Manipulation

Task	Book	Chapter	Section
Read files with different key lengths.	Advantage VISION:Builder Reference Guide	Chapter 6	Coordinated Files
Process only a subset of the file.	Advantage VISION:Builder Reference Guide	Chapter 5	Application Cycle Overview
		Chapter 16	IMS™ Processing
	Advantage VISION:Builder Specifications Guide	Chapter 2	RC Statement

Task	Book	Chapter	Section
Compare keys in multiple files to determine keys on one file and not another.	Advantage VISION:Builder Reference Guide	Chapter 6	Coordinated Files
			Sequential Coordination
Convert a file from one database manager to another (VSAM to DB2).	Advantage VISION:Builder Reference Guide	Chapter 2	Discipline of Advantage VISION:Builder
		Chapter 3	Define a Relational File
		Chapter 7	Subfiles
Audit trail when updating. ❹	Advantage VISION:Builder Reference Guide	Chapter 10	Transaction File and Sequential Master File Record Flows
		Chapter 15	Transaction Record Rejection and Type 4 Procedure Processing Flow
Read IMS and DB2 files in same run.	Advantage VISION:Builder Reference Guide	Chapter 3	Concept of Structured Files
			Define an IMS File
			Define a Relational File
		Chapter 6	Coordinated Files

Task	Book	Chapter	Section
Control reading of a file.	Advantage VISION:Builder Reference Guide	Chapter 6	Coordinated Files
		Chapter 13	Controlling Array Looping
Generate a test file from a production file.	Advantage VISION:Builder Reference Guide	Chapter 7	Subfiles
Stop processing on demand.	Advantage VISION:Builder Reference Guide	Chapter 5	Flag Fields
		Chapter 8	Branching to Control Looping
		Chapter 10	Transaction File and Master File Record Flows ❹
Process table databases.	Advantage VISION:Builder Reference Guide	Chapter 3	Define a Relational File
		Chapter 15	Relational Updating Considerations
Restrict users view of file data.	Advantage VISION:Builder Reference Guide	Chapter 3	Using a Logical Record
Check program for efficiency.	Advantage VISION:Builder Reference Guide	Chapter 20	Using PAL to Help Maintain a Program
	Advantage VISION:Builder Environment Guide		

Task	Book	Chapter	Section
Feed parameter to Advantage VISION:Builder program (EXEC linkage or user-read).	Advantage VISION:Builder Reference Guide	Chapter 6	User Coordination
		Chapter 18	entire chapter
	Advantage VISION:Builder Environment Guide	Chapter 6	entire chapter

Definition Processing

Task	Book	Chapter	Section
Use a different definition - just for this run.	Advantage VISION:Builder Reference Guide	Chapter 3	Catalog the File Definition in the Common Library
		Chapter 19	Processing with Multiple Common Libraries
How to use multiple COMLIBs.	Advantage VISION:Builder Reference Guide	Chapter 19	Instream File Definitions
	Advantage VISION:Builder Environment Guide	Chapter 13	entire chapter

COBOL and Advantage VISION:Builder

Task	Book	Chapter	Section
Access COBOL file with OCCURS DEPENDING ON followed by more fields.	Advantage VISION:Builder Reference Guide	Chapter 4	Variable Length Fields
		Chapter 18	entire chapter
Call a COBOL routine from Advantage VISION:Builder.	Advantage VISION:Builder Reference Guide	Chapter 18	entire chapter

Table Processing

Task	Book	Chapter	Section
Convert codes to descriptions automatically.	Advantage VISION:Builder Reference Guide	Chapter 12	Defining Tables
			Automatic Table Lookup

Performance Tuning

Task	Book	Chapter	Section
Optimize your program.	Advantage VISION:Builder Reference Guide	Chapter 20	entire chapter

Chapter 3

Using COBOL Quick Start

COBOL Quick Start is a batch utility that generates a skeletal COMLIB file definition from an existing COBOL file definition. After a skeletal COMLIB file definition has been created, you can specify any additional file information that is required using the Advantage VISION:Builder Workbench for DOS, Advantage VISION:Builder Workbench for ISPF, or a text editor.

You can also run COBOL Quick Start from Advantage VISION:Builder Workbench for ISPF. The IMPORT option points you to a menu for selecting the quick start utility you want to run. The subsequent panels prompt you for the information needed to run the utility. After the information is gathered, the utility is run immediately and the output is displayed for you to browse.

Note: File definitions are stored to and retrieved from a common library (COMLIB) when needed at processing time. Unless specifically stated otherwise, the term COMLIB refers to this common file definition.

COBOL Quick Start can retrieve COBOL copybooks from z/OS data sets, CA-Panvalet® libraries, and CA-Librarian® libraries. Most COMLIB file definition types are supported by COBOL Quick Start.

Flow Diagram

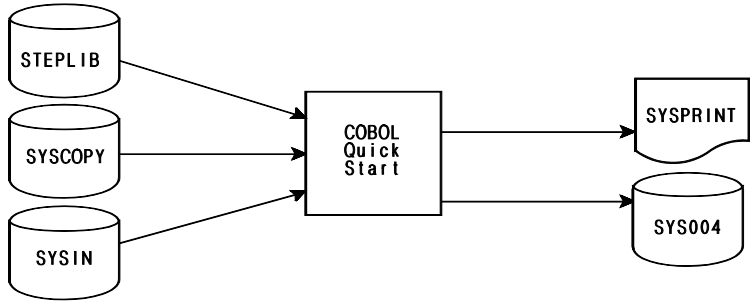

Figure 3-1 COBOL Quick Start Flow Diagram

Note: If a CA-Panvalet or CA-Librarian copybook library is used, a PANDD1 (CA-Panvalet) or MASTER (CA-Librarian) DD statement is required. For more information, see Using CA-Panvalet and CA-Librarian COBOL Copybooks.

As shown in Figure 3-1, COBOL Quick Start uses the following input data sets:

- STEPLIB Specifies the COMLIB load library.
- SYSCOPY Specifies a z/OS COBOL copybook library.
- SYSIN Provides the appropriate COBOL Quick Start control statements. You can also specify Instream COBOL field definitions here.

COBOL Quick Start produces two output files:

- SYSPRINT Contains a report on the file definition generation process that includes a listing of the COBOL statements that were processed.
- SYS004 Contains the generated COMLIB file definition source statements.

Utility Execution

COBOL Quick Start is a batch utility.The job control language (JCL) statements required to execute this utility are shown in Figure 3-2. The JCL contains an instream procedure, followed by JCL statements to execute the procedure. Note that this JCL uses sample SYSIN data. You can use this sample COBOL data to see exactly how COBOL Quick Start works.

At a minimum, you must make the following changes before submitting this JCL:

- Supply a job card.
- Supply values for the procedure variables detailed in the following table.

Variable Name	Description
CLLOAD	Specify the name of your COMLIB load library.
COPYLIB	Specify the name of the COBOL copylib that is needed.
DEFLIB	Specify the source definition library where the new file definition should be written.
MEMBER	Specify a member name for the new file definition. This name must be the same name that is specified in the NAME parameter on the FILEGEN statement. If an existing member name is specified, it is overwritten. For more information, see FILEGEN Control Statement.

- Provide a DD statement override for COBOLQS.SYSIN. This data set contains the required COBOL Quick Start input control statements used to control the generation of the COMLIB file definition. For more information, see Control Statements.

- To conform to your standards, you may need to make additional modifications.

```
//* MEMBER CLCOBQS
//*********************************************************************
//* EXECUTE THE COBOL QUICK START UTILITY.
//* THE SYSCOPY DD STATEMENT IS USED FOR MVS COPYBOOK LIBRARIES.
//* THE PANDD1  DD STATEMENT IS USED FOR PANVALET COPYBOOK LIBRARIES.
//* THE MASTER  DD STATEMENT IS USED FOR LIBRARIAN COPYBOOK LIBRARIES.
//*********************************************************************
//COBOLQS PROC CLLOAD=,
//            COPYLIB=,
//            DEFLIB=,
//            MEMBER=
//COBOLQS EXEC PGM=COBOLQS,REGION=1024K
//STEPLIB   DD DSN=&CLLOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSCOPY   DD DSN=&COPYLIB,DISP=SHR
//PANDD1    DD DSN=&COPYLIB,DISP=SHR
//MASTER    DD DSN=&COPYLIB,DISP=SHR
//SYS004    DD DSN=&DEFLIB(&MEMBER),DISP=OLD
//SYSIN     DD DUMMY
//       PEND
//*********************************************************************
//*  BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING
//*  INFORMATION:
//*    CLLOAD  - NAME OF YOUR COMLIB LOAD LIBRARY
//*   COPYLIB - NAME OF YOUR COBOL COPY LIBRARY.  THIS IS AN
//*             MVS, PANVALET, OR LIBRARIAN COPYBOOK LIBRARY.
//*    DEFLIB  - NAME OF YOUR COMLIB SOURCE DEFINITION LIBRARY
//*              THE GENERATED FILE DEFINITION IS WRITTEN TO
//*               THIS LIBRARY.
//*   MEMBER  - MEMBER NAME FOR THE DEFINITION YOU ARE GENERATING.
//*
//*  YOU MUST ALSO PROVIDE THE APPROPRIATE SYSIN DATA IN THE
//*  COBOLQS.SYSIN DD STATEMENT OVERRIDE.
//*********************************************************************
//QS      EXEC COBOLQS,
//            CLLOAD='BUILDER.CL045.LOADLIB',
//           COPYLIB='COBOL.COPYBOOK.LIBRARY',
//            DEFLIB='COMLIB.DEFLIB',
//            MEMBER='SAMPLEFD'
//COBOLQS.SYSIN DD *
  FILEGEN NAME=SAMPLEFD,TYPE=FIXED,RECSIZE=80
  SEGMENT NAME=OFFICE,NUMBER=10,LEVEL=1
  $COBOL
    01  OFFICE-DATA.
        02  OFFICE-CODE            PIC S9(3).
        02  OFFICE-ADDRESS.
            03  OFFICE-STREET         PIC X(20).
            03  OFFICE-CITY           PIC X(15).
            03  OFFICE-STATE          PIC X(2).
            03  OFFICE-ZIP.
                04  OFFICE-ZIP-FIRST-FIVE  PIC X(5).
                04  OFFICE-ZIP-LAST-FOUR   PIC X(4).
        02  OFFICE-PHONE           PIC 9(7).
        02  OFFICE-AREA-CODE       PIC X(3).
        02  SPEED-DIAL             PIC X(3).
        02  FILLER                 PIC X(18).
  $ECOBOL
/*
```

Figure 3-2 Sample Execution JCL for COBOL Quick Start

Using CA-Panvalet and CA-Librarian COBOL Copybooks

COBOL Quick Start provides direct access to COBOL copybooks that are stored in CA-Panvalet or CA-Librarian source libraries. Before using this facility, you must link edit the appropriate CA-Panvalet or CA-Librarian interface modules with the COBOL Quick Start interface modules. The required CA-Panvalet and CA-Librarian interface modules are included in your CA-Panvalet or CA-Librarian software package.

CA-Panvalet Interface

Note: The generated COMLIB field length for a GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC string is 2**n*, where *n* is the DB2 length of the string (number of DBCS characters).

Figure 3-3 contains the JCL required for link editing the CA-Panvalet interface modules with the COBOL Quick Start load module.

You may have to replace the INCLUDE LIBSYS(PAM) statement with several INCLUDE statements. For the exact INCLUDE statements that are required, see your CA-Panvalet book.

```
//* MEMBER CLCOBPL
//*********************************************************************
//* LINK EDIT PANVALET INTERFACE MODULES WITH COBOL QUICK START.
//*********************************************************************
//CLPANLK PROC CLLOAD=,
//             PANLOAD=
//LINK    EXEC PGM=IEWL,REGION=512K,PARM='LIST,MAP,LET,XREF,NCAL'
//SYSLIB    DD DUMMY
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//LIBSYS    DD DSN=&PANLOAD,DISP=SHR
//LLIB      DD DSN=&CLLOAD,DISP=SHR
//SYSLMOD   DD DSN=&CLLOAD,DISP=SHR
//         PEND
//*********************************************************************
//* BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING
//* INFORMATION:
//*   CLLOAD  - NAME OF YOUR COMLIB LOAD LIBRARY.
//*   PANLOAD - NAME OF YOUR PANVALET SYSTEM LOAD LIBRARY.
//*********************************************************************
//PANLINK EXEC CLPANLK,
//             CLLOAD='BUILDER.CL045.LOADLIB',
//             PANLOAD='PANVALET.SYSTEM.LOADLIB'
//LINK.SYSLIN  DD *
  INCLUDE LIBSYS(PAM)
  INCLUDE LLIB(COMLIBP)
  ENTRY COMLIBP
  NAME COMLIBP(R)
/*
```

Figure 3-3 JCL to Link CA-Panvalet Interface Modules

After the CA-Panvalet interface modules have been link edited, the COPYPCOBOL parameter on the SEGMENT statement can be used. For the specifications for the SEGMENT statement, see SEGMENT Control Statement in this chapter.

CA-Librarian Interface

Figure 3-4 contains a listing of the JCL required for link editing the CA-Librarian interface modules with the COBOL Quick Start interface modules.

You may have to replace the INCLUDE LIBSYS(FAIR) statement with several INCLUDE statements. For the exact INCLUDE statements that are required, see your CA-Librarian book. Unresolved external references occur when link editing the CA-Librarian interface; these are normal and you can safely ignore them.

```
//* MEMBER CLCOBLL                                                        00010000
//*********************************************************************  00020000
//* LINK EDIT LIBRARIAN INTERFACE MODULES WITH COBOL QUICK START.    *  00030000
//*********************************************************************  00040000
//CLLIBLK PROC CLLOAD=,                                                  00050000
//             LIBLOAD=                                                  00060000
//LINK    EXEC PGM=IEWL,REGION=2M,PARM='LET,LIST,MAP,NCAL'               00070000
//SYSLIB    DD DUMMY                                                     00080000
//SYSPRINT  DD SYSOUT=*                                                  00090000
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(1,1))                              00100000
//LIBSYS    DD DSN=&LIBLOAD,DISP=SHR                                     00110000
//LLIB      DD DSN=&CLLOAD,DISP=SHR                                      00120000
//SYSLMOD   DD DSN=&CLLOAD,DISP=SHR                                      00130000
//        PEND                                                           00140000
//*********************************************************************  00150000
//* BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING       *  00160000
//* INFORMATION:                                                     *  00170000
//*                                                                  *  00180000
//*   CLLOAD  - NAME OF YOUR COMLIB LOAD LIBRARY.                    *  00190000
//*   LIBLOAD - NAME OF YOUR LIBRARIAN SYSTEM LOAD LIBRARY.          *  00200000
//*                                                                  *  00210000
//*********************************************************************  00220000
//LIBLINK EXEC CLLIBLK,                                                  00230000
//             CLLOAD='BUILDER.CL045.LOADLIB',                           00240000
//             LIBLOAD='LIBRARN.SYSTEM.LOADLIB'                          00250000
//LINK.SYSLIN  DD *                                                      00250100
  INCLUDE LIBSYS(FAIRCLS)                                               00250200
  INCLUDE LIBSYS(FAIRERR)                                               00250300
  INCLUDE LIBSYS(FAIRLOC)                                               00250400
  INCLUDE LIBSYS(FAIRMOD)                                               00250500
  INCLUDE LIBSYS(FAIRNTE)                                               00250600
  INCLUDE LIBSYS(FAIROPN)                                               00250700
  INCLUDE LIBSYS(FAIRPNT)                                               00250800
  INCLUDE LIBSYS(FAIRREC)                                               00250900
  INCLUDE LIBSYS(FAIRSCAN)                                              00251000
  INCLUDE LIBSYS(FAIRSEC)                                               00251100
  INCLUDE LLIB(COMLIBL)                                                 00251200
  ENTRY COMLIBL                                                         00251300
  NAME COMLIBL(R)                                                       00252000
/*                                                                      00260000
```

Figure 3-4 JCL to Link CA-Librarian Interface Modules

After you have link edited the CA-Librarian interface modules, you can use the COPYLCOBOL parameter on the SEGMENT statement. For the specifications for the SEGMENT statement, see SEGMENT Control Statement.

Control Statements

COBOL Quick Start uses the following control statements as input:

- FILEGEN — The FILEGEN statement provides a name for the file definition that is being generated and identifies the type of file (for example, VSAM KSDS or IMS) being generated.
- SEGMENT — The SEGMENT statement is used to define the segments within a file.
- $COBOL — This statement signals that an in-stream COBOL definition follows.
- $ECOBOL — This statement signals the end of an in-stream COBOL definition.

The function and syntax of these statements are described in the following sections.

Coding Rules

When writing COBOL Quick Start control statements, you must observe the following rules:

- Each non-continued control statement must contain a control statement command that identifies the control statement type.
- Each control statement may contain keyword parameters.
  - Keyword parameters may be specified in any order. Keyword parameters must be separated by commas.
  - Embedded blanks are not allowed between parameters.
  - Each parameter must be coded unless stated otherwise.
- A comma following the last parameter on a statement causes a continuation to the next statement.
- Comments can be placed after the last parameter with an intervening blank.
- Columns 1 through 71 of the control statement are scanned. Columns 72 through 80 are ignored.

FILEGEN Control Statement

Statement Syntax

```
FILEGEN   NAME= ,
          TYPE=,
          RECSIZE=,
          RECBLK=,
          BUFFSIZE=,
          FLDPREFX=
```

Figure 3-5 FILEGEN Control Statement Format

- FILEGEN (required) — FILEGEN is a control statement command. It identifies the control statement type.
- NAME (required) — The NAME parameter specifies the name of the file definition being generated. A file name can be from 1-8 characters long. The first character must be alphabetic. The remaining characters can be a combination of alphanumeric characters. When the generated file definition is written to a partitioned data set, the file name specified here must be identical to the member name specified in the JCL.
- TYPE (required) — The TYPE parameter specifies the type of COMLIB file definition that you want to create. Valid file types are listed in the following table.

File Type	Description
DB2	DB2 Relational Database
KSDS	VSAM Key Sequenced Data Set
ESDS	VSAM Entry Sequenced Data Set
AIX	VSAM Alternate Index Data Set
DLI	IMS Database
DLIHDAM	IMS HDAM Database
ISAMFIX	ISAM Fixed Length Record Format Data Set

File Type	Description
ISAMVAR	ISAM Variable Length Record Format Data Set
FIXED	Fixed Length Record Format Data Set
VARIABLE	Variable Length Record Format Data Set
UNDEFINED	Undefined Record Format Data Set
GDBI	General Data Base Interface Mapped File

- RECSIZE (optional) — The RECSIZE parameter specifies the number of data bytes in the data portion of a record or segment. Enter a number from 1-9999. This parameter only applies to FIXED, ISAMFIX, VARIABLE, or ISAMVAR file types.
- RECBLK (optional) — The RECBLK parameter specifies the number of records in each block. Enter a number from 1-999. This parameter only applies to FIXED and ISAMFIX file types.
- BUFFSIZE (optional) — The BUFFSIZE parameter specifies the size of the buffer that is needed to process the file. Enter a value from 1-32760 or 1K-9999K.

  For DB2, DLI, DLIHDAM, and GDBI file types, enter the maximum amount of main storage required to hold a logical record. For KSDS and ESDS file types, enter the maximum record size according to the VSAM cluster definition. If the file type is AIX, enter the alternate index control interval size. For ISAMFIX, ISAMVAR, FIXED, VARIABLE, and UNDEFINED, enter the block size.

- FLDPREFX (optional) The FLDPREFX parameter specifies the 1-3 character prefix to use for generating primary field names in the COMLIB file definition. Primary field names are required in a COMLIB file definition and must be assigned a unique 1-8 character name. Since COBOL field names can be longer than 8 characters and may not be unique within the first 8 characters, COBOL Quick Start automatically generates a unique 8-character primary field using the FLDPREFX value followed by a generated field number.

  If the FLDPREFX parameter is omitted, the default prefix is F and the generated primary field names have the format Fnnnnnnn where *nnnnnnn* is a number from 0000001-9999999. For more information, see COMLIB Field Name Generation.

SEGMENT Control Statement

Statement Syntax

```
SEGMENT  NAME= ,
         NUMBER=,
         LEVEL=,
         COPYCOBOL=,
         COPYPCOBOL=,
         COPYLCOBOL=
```

Figure 3-6 SEGMENT Control Statement Format

- SEGMENT (required) — SEGMENT is a control statement command. It identifies the control statement type.
- NAME (required) — The NAME parameter assigns a name to a segment. Segment names can be from 1-8 characters. The first character must be alphabetic. The remaining characters can be a combination of alphanumeric characters.
- NUMBER (required) — The NUMBER parameter assigns a number that uniquely identifies the segment. Enter a number from 1-255. Subordinate segments must have a number larger than the parent segment and smaller than any children segments.
- LEVEL (required) — The LEVEL parameter specifies the subordination of segments. The root segment must have a level number of 1. All subordinate segments must be assigned a number from 2 to 9.
- COPYCOBOL (optional) — The COPYCOBOL parameter specifies the name of the COBOL copybook that contains the field definitions for this segment. This copybook must be located in a z/OS data set assigned to the SYSCOPY DD statement in the JCL.

- COPYPCOBOL (optional) The COPYPCOBOL parameter specifies the name of the COBOL copybook that contains the field definitions for this segment. This copybook must be located in a CA-Panvalet library assigned to the PANDD1 DD statement in the JCL.
- COPYLCOBOL (optional) The COPYLCOBOL parameter specifies the name of the COBOL copybook that contains the field definitions for this segment. This copybook must be located in a CA-Librarian library assigned to the MASTER DD statement in the JCL.

Processing Notes

COPYCOBOL, COPYPCOBOL, and COPYLCOBOL parameters are mutually exclusive. Only one of these parameters should be specified on a SEGMENT statement. Figure 3-7 shows an example of the input control statements used with the COPY parameter.

```
FILEGEN NAME=XXXX,TYPE=X
SEGMENT NAME=XXXX,NUMBER=XX,LEVEL=X,
COPYCOBOL=copybookname
```

Figure 3-7 Control Statements Used with the COPY Parameter

When using any of the COPY parameters (COPYCOBOL, COPYPCOBOL, COPYLCOBOL), you can specify a NOPRINT option to suppress the listing of the copybook on the SYSPRINT file. This option is specified as `COPYCOBOL=(copybookname,NOPRINT)`

If a COPY parameter is not specified on the SEGMENT statement, then you must provide an in-stream COBOL definition. In this case, the SEGMENT statement must be followed by a $COBOL statement, the in-stream COBOL definition, and the $ECOBOL statement which marks the end of the definition. For more information, see $COBOL and $ECOBOL Control Statements.

$COBOL and $ECOBOL Control Statements

Example of Syntax and Use

```
FILEGEN NAME=XXXX,TYPE=X
SEGMENT NAME=XXXX,NUMBER=XX,LEVEL=X
$COBOL
instream COBOL source statements
$ECOBOL
```

Figure 3-8 $COBOL and $ECOBOL Control Statements

The $COBOL and $ECOBOL control statements are used in place of the SEGMENT COPY parameter to process instream COBOL source statements. The $COBOL statement must follow a SEGMENT statement that does not contain a COPY parameter. The actual COBOL source statements to be processed must follow the $COBOL statement. The end of the in-stream COBOL source statements must be marked by the $ECOBOL statement. The in-stream COBOL source statements are used to generate field definitions for the segment.

$COBOL and $ECOBOL can be placed anywhere within columns 1-71. They must be the only command on the statement. There are no parameters for either statement.

Conversion Rules

The following sections describe how the Quick Start utility generates various file definition paramaters.

Generated COMLIB File Definition

This utility generates a skeletal COMLIB file definition which must be edited and validated by Advantage VISION:Builder Workbench for DOS, Advantage VISION:Builder Workbench for ISPF, or a text editor prior to its use. You should consider the following items when editing the generated file definition:

- Segment key assignment — Each segment must define at least one field as the key; you can also assign additional keys.
- Segment information — You should provide, as needed, additional segment information such as segment order and number of fixed occurrences.
- Field information — You can modify primary and alternate field name assignments (for more information, see COMLIB Field Name Generation). You should provide, as needed, additional field information such as rounding, editing, and automatic table lookup results.

COMLIB Field Name Generation

A unique 1-8 character name must be assigned to all fields within a COMLIB file definition. This name is referred to as the primary field name.

When building a skeletal COMLIB file definition from COBOL field definitions, a unique 1-8 character primary field name must be assigned to each field. Since COBOL field names can be longer than eight characters and may not be unique within the first eight characters, COBOL Quick Start automatically generates a unique primary field name using the FLDPREFX parameter value on the FILEGEN statement followed by a generated field number. If the FLDPREFX parameter is omitted, the default prefix is F and the generated primary field names have the format F*nnnnnnn* where *nnnnnnn* is a number from 0000001-9999999.

COBOL Quick Start uses the COBOL field name to generate the Column Heading specifications.

After the skeletal file definition has been created, you can import it into the Advantage VISION:Builder Workbench for DOS, Advantage VISION:Builder Workbench for for ISPF, and you can replace the generated primary field names with more descriptive primary field names.

Unsupported COBOL Specifications

COBOL Quick Start does not support the following COBOL field specifications:

- 66 levels. All references to the field are removed. A warning message is issued.
- 77 levels. Any field defined at level 77 has a starting location of 1. You should check the resulting definition. A warning message is issued.
- 88 levels. All references to the field are removed. A warning message is issued.
- Numeric fields with more than 9 digits to the right of the decimal place are supported for packed fields; however, for any other field type, a warning message is issued indicating that the number of decimal places has been truncated to 9.
- Binary fields larger than S9(9) are generated as character fields. A warning message is issued.
- COMP-2 data types are generated as 8-byte character fields. A warning message is issued.
- If the length of a numeric field exceeds 15 digits, the field is generated as character. A warning message is issued.
- The P edit parameter on the PICTURE clause always generates a zero SCALE value. You should check the resulting definition. A warning message is issued.
- If the size of a field exceeds 255, a warning message is issued indicating that the generated field length has been truncated to 255. The generated definition should be modified to include an additional field that defines the remaining bytes or at least the last byte of the field.
- OCCURS clause. A warning message is issued to indicate that only 1 occurrence of the item was generated. Additional occurrences must be added to the definition.

If the item on the OCCURS clause is variably occurring (DEPENDING ON clause present), then you can define the additional occurrences by defining the generated occurrence as a lower level variably occurring segment, if possible. This requires the field containing the number of occurrences to be defined as a count field in the generated file definition.

When you encounter a DEPENDING ON clause, an additional message is issued to inform you that you must adjust the field start locations of subsequent fields. This is because the generated start location of the next field will be incorrect if you define the generated occurrence as a lower level variably occurring segment. Also, the start location of the next field is calculated using the maximum number of occurrences when in reality the number of occurrences varies.

If the item on the OCCURS clause is fixed occurring (no DEPENDING ON clause), then you can define the additional occurrence by doing one of the following:

- Defining the generated occurrence as a lower level fixed occurring segment.

  If you use this method, a reference to the field initiates a loop where each occurrence of the field is automatically processed.

- Defining a separate field for each occurrence.

  If you select this method, then a separate field name exists for each occurrence.

  Note: If the field name is FILLER or blank, all references to the field are removed from the definition.

- Defining one field whose length accommodates all occurrences (for example, define a 30-byte field if you have a 3-byte field that occurs 10 times).

  If you choose this method, then each occurrence can be accessed by using dynamic partial fielding, which is similar to indexing.

Chapter

4

Using DB2 Quick Start

DB2 Quick Start is a batch utility that generates COMLIB file definitions from existing DB2 table definitions. During DB2 Quick Start processing, each DB2 table that is processed becomes a separate segment in the COMLIB file definition being created. DB2 column information is retrieved from the DB2 SYSCOLUMNS table, converted into COMLIB field specifications, and added to the segment being created. A COMLIB field statement (L0) is created for each column in the specified DB2 table.

Multiple file definitions can be generated in a single DB2 Quick Start execution. Each generated file definition is stored as a separate member in the indicated source definition library. The specified file name is used as the member name when the generated definition is saved in the source definition library. If an existing member name is specified, the existing member is replaced when the new definition is saved. If a new member name is specified, then a new member is created.

After a COMLIB file definition has been created, any additional file information that is required can be added with Advantage VISION:Builder for DOS, Advantage VISION:Builder for ISPF, or a text editor.

DB2 Quick Start can also run from Advantage VISION:Builder for ISPF. The IMPORT option points you to a menu for selecting the quick start utility you want to run. The subsequent panels prompt you for the information needed to run the utility. After the information is gathered, the utility runs immediately and the output is displayed for you to browse.

Note: File definitions are stored to and retrieved from a common library (COMLIB) when needed at processing time. Unless specifically stated otherwise, the term COMLIB refers to this common file definition.

Flow Diagram

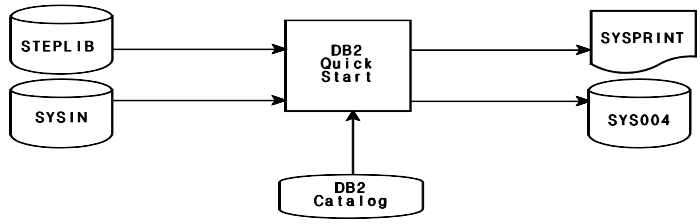

Figure 4-1 DB2 Quick Start Flow Diagram

As shown in Figure 4-1, DB2 Quick Start uses the following two input data sets:

- STEPLIB Specifies the COMLIB load library and the DB2 load library.
- SYSIN Specifies the appropriate DB2 Quick Start control statements.

DB2 Quick Start produces two output files:

- SYSPRINT Contains a summary report on the file definition generation process. You can use the PRINT parameter on the SEGMENT control statement to control the contents of this report. For more information, see SEGMENT Control Statement.
- SYS004 Generated file definitions are written to the partitioned data set pointed to by the DD name SYS004. This must be a partitioned data set.

Utility Execution

DB2 Quick Start is a batch utility. Before this utility can be executed, you must input the DB2 Quick Start Data Base Request Module (DBRM), provided on the installation tape, to a DB2 bind. This creates the DB2 plan required by DB2 Quick Start.

Figure 4-2 contains the same JCL required to execute DB2 Quick Start. This JCL contains an in-stream procedure followed by JCL statements to execute the procedure and sample DB2 Quick Start control statements. You can use the sample control statements to create a COMLIB file definition from the sample tables provided with DB2 (DSN8230.DEPT, DSN8230.EMP, and DSN8230.PROJ). If you have created a DB2 plan by binding the DB2 Quick Start DBRM during the installation process, you can use these sample control statements to see how DB2 Quick Start works.

At a minimum, you must make the following changes before submitting this JCL:

- Supply a job card
- Supply values for the following procedure variables detailed in the following table.

Variable Name	Description
CLLOAD	Specify the name of your COMLIB load library.
DB2LOAD	Specify the name of your DB2 load library.
DEFLIB	Specify the source definition library to which the new file definition should be written. DEFLIB must specify a partitioned data set.

- Provide a DD statement override for DB2QS.SYSIN. This data set contains the required DB2 Quick Start input control statements which are used to control the generation of COMLIB file definitions. For more information, see Control Statements.
- To conform to your standards, you may need to make additional modifications.

```
//* MEMBER CLDB2QS
//*********************************************************************
//* EXECUTE THE DB2 QUICK START UTILITY.
//*********************************************************************
//DB2QS   PROC CLLOAD=,
//             DB2LOAD=,
//             DEFLIB=
//DB2QS   EXEC PGM=DB2QS,REGION=1024K
//STEPLIB   DD DSN=&CLLOAD,DISP=SHR
//            DD DSN=&DB2LOAD,DISP=SHR
//SYSTERM   DD DUMMY
//SYSPRINT  DD SYSOUT=*,
//             DCB=(DSORG=PS,RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYS004    DD DSN=&DEFLIB,DISP=OLD
//SYSIN     DD DUMMY
//        PEND
//*********************************************************************
//* BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING
//* INFORMATION:
//*
//*   CLLOAD - NAME OF YOUR COMLIB LOAD LIBRARY.
//*   DB2LIB - NAME OF YOUR DB2 DSN.DSNLOAD LIBRARY.
//*   DEFLIB - NAME OF YOUR COMLIB SOURCE DEFINITION LIBRARY.
//*            THE GENERATED FILE DEFINITION IS WRITTEN TO
//*            THIS LIBRARY.
//*
//* YOU MUST ALSO PROVIDE THE APPROPRIATE SYSIN DATA IN THE
//* DB2QS.SYSIN DD STATEMENT OVERRIDE.
//*********************************************************************
//QS      EXEC DB2QS,
//             CLLOAD='BUILDER.CL045.LOADLIB',
//             DB2LOAD='DB2.SYSTEM.DSNLOAD',
//             DEFLIB='COMLIB.DEFLIB'
//*
//DB2QS.SYSIN  DD *
  DB2CNTL DB2PLAN=DB2QS,DB2SYS=DB2T
  FILEGEN NAME=DB2FD,BUFFSIZE=1024K
  SEGMENT NAME=DEPT,NUMBER=10,LEVEL=1,TABLE=DEPT,CREATOR=DSN8230,
          PRINT=ALL
  NEWPAGE
  SEGMENT NAME=EMPLOYEE,NUMBER=20,LEVEL=2,TABLE=EMP,CREATOR=DSN8230,
          PRINT=ALL
  NEWPAGE
  SEGMENT NAME=PROJECT,NUMBER=30,LEVEL=2,TABLE=PROJ,CREATOR=DSN8230,
          PRINT=ALL
/*
```

Figure 4-2 Sample Execution JCL for DB2 Quick Start

Control Statements

DB2 Quick Start uses the following control statements as input:

- DB2CNTL — Information from the DB2CNTL statement establishes the proper DB2 environment.
- FILEGEN — The FILEGEN statement triggers the start of a new file definition. The NAME parameter specified on the FILEGEN statement is used as both the file name and the member name when the generated file definition is saved.
- SEGMENT — The SEGMENT statement defines file segments. Each segment corresponds to a DB2 table. Field information for the specified DB2 table is retrieved from the DB2 SYSCOLUMNS table, converted to COMLIB field specifications, and used to create field definitions for the current segment.
- NEWPAGE — The NEWPAGE statement triggers a page eject on the DB2 Quick Start summary report.

The function and syntax of these statements are described in the following sections.

Coding Rules

When writing DB2 Quick Start control statements, you must observe the following rules:

- Each non-continued control statement must contain a control statement command that identifies the control statement type.
- Each control statement may contain keyword parameters.
  - Keyword parameters may be specified in any order.
  - Keyword parameters must be separated by commas.
  - Embedded blanks are not allowed between parameters.
- Each parameter must be coded unless stated otherwise.

- A comma following the last parameter on a statement causes a continuation to the next statement.
- Comments may be placed after the last parameter with an intervening blank.
- Blank lines within the SYSIN file are printed on the summary report output to SYSPRINT. You can use blank lines, along with the NEWPAGE control statement, to format the DB2 Quick Start summary report.
- Columns 1 through 71 of the control statement are scanned. Columns 72 through 80 are ignored.

DB2CNTL Control Statement

Statement Syntax

```
DB2CNTL   DB2PLAN=,
          DB2SYS=
```

Figure 4-3 DB2CNTL Control Statement Format

- DB2CNTL (required) — DB2CNTL is a control statement command. It identifies the control statement type. The DB2 control statement is used to establish the proper DB2 environment. DB2 Quick Start uses the TSO Call Attach Facility to connect to DB2. Static SQL is used to process information in the DB2 SYSCOLUMNS table.
- DB2PLAN (required) — The DB2PLAN parameter specifies the name of the DB2 plan that should be used when accessing DB2 tables. The DB2 plan is created during the installation process by binding the provided DB2 Quick Start DBRM.
- DB2SYS (required) — The DB2SYS parameter specifies the name of the DB2 system that is to be used.

FILEGEN Control Statement

Statement Syntax

```
FILEGEN   NAME=,
         BUFFSIZE=,
         FLDPREFX=,
         FLDNAME=,
         HEADING=,
         LOGREL=,
         LONGNAME=,
         DESCRIPT=,
         DATEFLD=
```

Figure 4-4 FILEGEN Control Statement Format

- FILEGEN (required) — FILEGEN is a control statement command. It identifies the control statement type. The FILEGEN control statement triggers the start of a new file definition.
- NAME (required) — The NAME parameter specifies the name of the file definition that is being generated. A file name can be from 1 to 8 characters. The first character must be alphabetic. The remaining characters can be a combination of alphanumeric characters.

  When the generated file definition is saved, the file name is also used as the member name. If an existing member name is specified, the existing member is overwritten. If a new member name is specified, a new member is created.
- BUFFSIZE (optional) — The BUFFSIZE parameter defines the size of the buffer needed to process a logical record. Enter a number from 1 to 32760. Multiples of 1024 can be entered as *nnnn*K where *nnnn* is a number between 1 to 9999.

- FLDPREFX (optional) — The FLDPREFX parameter specifies the 1 to 3 character prefix for generating primary field names in the COMLIB file definition. Primary field names are required in a COMLIB file definition and must be assigned a unique 1 to 8-character name. Since DB2 field names can be longer than 8 characters and may not be unique within the first 8 characters, DB2 Quick Start automatically generates a unique 8-character primary field using the FLDPREFX value followed by a generated field number.

  If the FLDPREFX parameter is omitted, the default prefix is F and the generated primary field names have the format F*nnnnnnn* where *nnnnnnn* is a number from 0000001 to 9999999. For more information, see COMLIB Field Name Generation.

- FLDNAME (optional) — The FLDNAME parameter specifies how the field name (short name) is derived. Enter GEN to specify that the name is to be generated using the FLDPREFIX parameter convention. Enter TRUNC to specify that the name is to be a truncation (first 8 characters) of the DB2 table column name.

  If the FLDNAME parameter is omitted, an entry of GEN is assumed. Note that an entry of TRUNC can result in duplicate names that you need to resolve by modifying the generated code.

- HEADING (optional) — The HEADING parameter specifies whether the Column Heading specifications (Ln statements) in the generated definition should contain the DB2 Label information for the SYSCOLUMNS table or the DB2 column name. Enter LABEL if the label information is to be used or COLUMN if the column name is to be used. If the HEADING parameter is omitted, the LABEL specification is assumed.

- LOGREL (optional) — The LOGREL parameter specifies whether DB2 Quick Start should generate LR statements for the definition. Enter YES to cause LR statements to be generated or NO to suppress the generation of LR statements. If the LOGREL parameter is omitted, an entry of YES is assumed. You must specify the foreign keys for a table to DB2 in order for this function of the utility to work correctly.
- LONGNAME (optional) — The LONGNAME parameter specifies whether DB2 Quick Start should generate LX statements for the definition. An entry of YES specifies that LX statements are to be generated containing the full column name. An entry of NO specifies that LX statements should not be generated. If the LONGNAME parameter is omitted, an entry of YES is assumed.
- DESCRIPT (optional) — The DESCRIPT parameter specifies whether DB2 Quick Start should generate D1 statements for the definition. An entry of YES specifies that D1 statements are to be generated using the Label information in the SYSCOLUMNS table. An entry of NO specifies that D1 statements should not be generated. If the DESCRIPT parameter is omitted, an entry of YES is assumed.
- DATEFLD (optional) — The DATEFLD parameter specifies how DB2 columns with a data type of DATE are to be handled. An entry of YES specifies that a DB2 data type of DATE should be defined as a Lilian Date (Type D) to Advantage VISION:Builder. An entry of NO specifies that a DB2 data type of DATE should be defined as a character string (Type C) with a length of 10 to Advantage VISION:Builder. If the DATEFLD parameter is omitted, an entry of NO is assumed.

SEGMENT Control Statement

Statement Syntax

```
SEGMENT   NAME=  ,
          NUMBER=,
          LEVEL=,
          TABLE=,
          CREATOR=,
          PRINT=
```

Figure 4-5 SEGMENT Control Statement Format

- SEGMENT (required) SEGMENT is a control statement command. It identifies the control statement type. Each COMLIB segment corresponds to a specific DB2 table.
- NAME (required) The NAME parameter assigns a name to a segment. Segment names can be from 1 to 8 characters.
- NUMBER (required) The NUMBER parameter assigns a number that uniquely identifies the segment. Enter a number from 1 to 255. Subordinate segments must have a number larger than the parent segment and smaller than any children segments.
- LEVEL (required) The LEVEL parameter specifies the subordination of segments. The root segment must have a level number of 1. All subordinate segments must be assigned a number from 2 to 9.
- TABLE (required) The TABLE parameter specifies the name of the DB2 table from which the field information for this segment should be generated.
- CREATOR (required) The CREATOR parameter qualifies the table name. Enter the authorization or creator ID for the table, or enter an asterisk (*). If an asterisk is entered, field information is generated from all tables with the specified table name, regardless of the creator ID.

- PRINT (optional) The PRINT parameter controls the information that is printed on the summary report. To print DB2 field information, enter DB2. To print COMLIB field information, enter ANSWER. To print both DB2 and COMLIB field information, enter ALL.

NEWPAGE Control Statement

Statement Syntax

```
NEWPAGE
```

Figure 4-6 NEWPAGE Control Statement Format

- NEWPAGE (required) NEWPAGE is a control statement command. It causes a page eject in the summary report that is written to SYSPRINT. There are no parameters for this control statement.

Conversion Rules

The following sections describe how the Quick Start utility generates various file definition paramaters.

Generated COMLIB File Definition

This utility generates a skeletal COMLIB file definition that, prior to its use, must be edited and validated by Advantage VISION:Builder Workbench for DOS, Advantage VISION:Builder Workbench for ISPF, or a text editor. You should consider the following items when editing the generate relational file definition:

- Segment key assignment — Each segment must define at least one field as the key; you can assign additional keys.
- Logical relationships — Logical relationships must be provided for segments defined at levels 2 to 9.
- Segment information — You should provide, as needed, additional segment information such as segment (row) order and suppress duplication.
- Field information — You can modify primary field name assignments can be modified if wanted. You should provide, as needed, additional field information such as rounding, editing, column headings, and automatic table lookup results.

COMLIB Field Name Generation

All fields within a COMLIB file definition must be assigned a unique 1- to 8- character name. This name is referred to as the primary field name.

When building a skeletal COMLIB file definition from DB2 field definitions, a unique 1 to 8-character primary field name must be assigned to each field. Since DB2 field names can be longer than 8 characters and may not be unique within the first 8 characters, DB2 Quick Start automatically generates a unique primary field name using the FLDPREFX parameter value on the FILEGEN statement followed by a generated field number. If the FLDPREFX parameter is omitted, the default prefix is F and the generated primary field names have the format F*nnnnnnn* where *nnnnnnn* is a number from 0000001 to 9999999.

DB2 Quick Start uses the DB2 column name to generate the External Column Name specification.

After the skeletal file definition has been created, it can be imported into Advantage VISION:Builder Workbench for DOS or Advantage VISION:Builder Workbench for ISPF, and you can replace the generated primary field names with more descriptive primary field names.

Generating COMLIB Field Information

The following table shows how the information in the DB2 SYSCOLUMNS table maps to the generated COMLIB field statements. The table on page 4-16 summarizes how DB2 field types and lengths are converted to COMLIB field types and lengths.

DB2 SYSCOLUMNS Field	COMLIB Field
NAME	Generates the Column Name specification.
COLTYPE	Generates the Field Type specification. See the table on page 4-16 for more information.
LENGTH	Generates the Field Length specification. See the table on page 4-16 for more information.
SCALE	Generates the Decimal Places specification. See the table on page 4-16 for more information.
KEYSEQ	Generates the Segment Key Value specification.
LABEL	Generates the Column Heading specifications.

Note: The generated COMLIB field length for a GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC string is 2*n where n is the DB2 length of the string (number of DBCS characters).

DB2		COMLIB	
Field Type	**Field Length**	**Field Type**	**Field Length**
INTEGER	4	Fixed (F)	4
SMALLINT	2	Fixed (F)	2
FLOAT	4	Floating Point (E)	4
FLOAT	8	Floating Point (E)	4 (Truncate)
DECIMAL	Up to 31 digits	Packed Decimal (P)	Up to 15 bytes - possible truncation
CHAR	Up to 254	Character (C)	Up to 255
VARCHAR	Up to maximum record size	Variable (V)	Up to 99H, possible truncation.
LONG VARCHAR	Up to maximum page size	Variable (V)	Up to 99H, possible truncation.
GRAPHIC	Up to 127	Character (C)	Up to 255
VARGRAPHIC	Up to maximum record size	Variable (V)	Up to 99H, possible truncation.
LONG VARGRAPHIC	Up to maximum page size	Variable (V)	Up to 99H, possible truncation.
DATE	4	Character (C)	10
TIME	3	Character (C)	8
TIMESTAMP	10	Character (C)	26

Chapter 5

Using Advantage VISION:Results Quick Start

The Advantage VISION:Results Quick Start utility generates COMLIB file definitions from existing Advantage VISION:Results file definitions.

Input to Advantage VISION:Results Quick Start is in the form of a sequential data set, PDS member, or z/OS, CA-Panvalet or CA-Librarian copybook containing a single Advantage VISION:Results file definition. Output from Advantage VISION:Results Quick Start is in the form of a single PDS member containing the converted file definition statements.

Advantage VISION:Results Quick Start can also run from Workbench for ISPF. The IMPORT option points you to a menu for selecting the Quick Start utility you want to run. The subsequent panels prompt you for the information needed to run the utility. When the information is gathered, the utility runs immediately and the output displayed for you to browse.

Flow Diagram

As shown in the following Advantage VISION:Results Quick Start Flow Diagram, Advantage VISION:Results Quick Start uses the following input data sets:

STEPLIB — Required. STEPLIB must specify the Advantage VISION:Builder installation load library containing the parameter module M4PARAMS and the Advantage VISION:Results Quick Start program module.

SYSIN — Required. SYSIN must specify an input file to be converted. It can be specified in various ways:

- A member in a PDS (the default in the supplied JCL).
- A sequential data set.
- An Advantage VISION:Results COPY statement for data from a z/OS copybook (SYSCOPY DD statement).
- An Advantage VISION:Results COPYP statement for data from a CA-Panvalet library (PANDD1 DD statement).
- An Advantage VISION:Results COPYL statement for data from a CA-Librarian master file (MASTER DD statement).
- As SYSIN in-stream data.

Note: Input must be a valid Advantage VISION:Results file definition beginning with a free-form FILE statement, followed by field definition statements.

For information about SYSIN data from sources other than a PDS, see DD Statement Overrides.

SYSCOPY — Optional. If SYSIN is to contain a COPY statement for a copybook from a z/OS PDS member, then the SYSCOPY DD statement must be provided, pointing to the z/OS PDS copy library.

PANDD1 Optional. If SYSIN is to contain a COPYP statement for a copybook from a CA-Panvalet library, then the PANDD1 DD statement must be provided, pointing to the CA-Panvalet library.

MASTER Optional. If SYSIN is to contain a COPYL statement for a copybook from a CA-Librarian master file, then the MASTER DD statement must be provided, pointing to the CA-Librarian master file.

Advantage VISION:Results Quick Start produces the following output data sets:

SYSPRINT The SYSPRINT data set contains a listing of the input statements, as well as any diagnostic or informational messages.

SYS004 The SYS004 file contains the converted file definition.

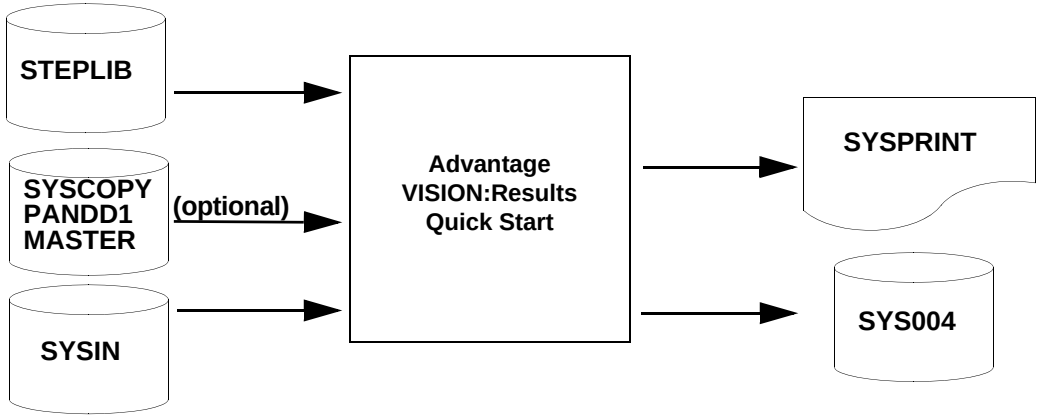

Figure 5-1 Advantage VISION:Results Quick Start Flow Diagram

Utility Execution

Advantage VISION:Results Quick Start is a batch program. You can invoke it online under the Import Main Menu option of Workbench for ISPF. You can also run it in batch mode by submitting the provided JCL member, BLXRSQ#3, in the PDS data set (...PREP.JCLCNTL). This JCL contains an in-stream procedure, followed by JCL statements to execute the procedure.

Figure 5-2 contains a listing of the sample JCL that is provided to run Advantage VISION:Results Quick Start. Review the supplied JCL carefully. At a minimum, you must make the following changes before submitting the JCL:

- Supply a job card.
- Supply appropriate variables for the procedure variables shown in the following table.

Variable Name	Description
RGN	Specify the region size for the utility execution. The default is 2 MB.
BLLOAD	Specify the Advantage VISION:Builder installation load library, which contains the M4PARAMS module.
DEFLIB	Specify the data set name of a PDS into which the converted file definition will be placed. The default is the Definition Processor Definition library PDS.
MEMBER	This is the output PDS member name that the converted definition will use. It is suggested that the member name match the file name of the file definition being converted.

Variable Name	Description
RSLTLIB	Specify the data set name of a PDS containing the Advantage VISION:Results file definition to be converted.
RSLTDEF	Specify the PDS member name of the Advantage VISION:Results file definition to be converted.

```
//* MEMBER BLXRSQ#3                                                      00010000
//*********************************************************************** 00020000
//* EXECUTE THE RESULTS QUICK START UTILITY                           * 00030000
//* ***** NOTE *****                                                  * 00040000
//* THE SYSCOPY DD STATEMENT IS USED FOR MVS COPYBOOK LIBRARIES.      * 00050000
//* THE PANDD1  DD STATEMENT IS USED FOR PANVALET COPYBOOK LIBRARIES. * 00060000
//* THE MASTER  DD STATEMENT IS USED FOR LIBRARIAN COPYBOOK LIBRARIES * 00070000
//*********************************************************************** 00080000
//RESLTQS PROC RGN=2M,                                                  00090000
//            BLLOAD=,                                                  00110000
//            DEFLIB=,                                                  00120000
//            MEMBER=,                                                  00130000
//            RSLTLIB=,                                                 00140000
//            RSLTDEF=                                                  00150000
//CONVRT EXEC PGM=RESULTQS,REGION=&RGN                                  00160000
//STEPLIB  DD DISP=SHR,DSN=&BLLOAD                                      00170000
//SYSPRINT DD SYSOUT=*                                                  00190000
//*SYSCOPY DD DISP=SHR,DSN=USER.RESULTS.COPYLIB                         00200000
//*PANDD1  DD DISP=SHR,DSN=USER.PANVALET.LIBRARY                        00210000
//*MASTER  DD DISP=SHR,DSN=USER.LIBR.MASTER                             00220000
//SYS004   DD DISP=OLD,DSN=&DEFLIB(&MEMBER)                             00230000
//SYSIN    DD DISP=SHR,DSN=&RSLTLIB(&RSLTDEF)                           00240000
//       PEND                                                           00250000
```

Figure 5-2 Sample Execution JCL for Advantage VISION:Results Quick Start (Page 1 of 2)

```
//********************************************************************* 00260000
//*  FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE. BEFORE YOU    * 00270000
//*  RUN THIS PROCEDURE, SPECIFY:                                     * 00280000
//*                                                                   * 00290000
//*    RGN     - THE REGION SIZE. DEFAULT IS 2M.                      * 00300000
//*    BLLOAD  - THE NAME OF YOUR BUILDER LOAD LIBRARY.               * 00320000
//*    DEFLIB  - THE LIBRARY(PDS) TO CONTAIN THE BUILDER DEFINITIONS. * 00330000
//*    MEMBER  - THE PDS MEMBER NAME FOR THE CONVERTED VISION:BUILDER * 00340000
//*              FILE DEFINITION IN THE DEFINITION LIBRARY.           * 00350000
//*    RSLTLIB - THE PDS CONTAINING THE VISION:Results FILE           * 00360000
//*              DEFINITION SOURCE STATEMENTS.                        * 00370000
//*    RSLTDEF - THE PDS MEMBER NAME OF THE INPUT VISION:Results      * 00380000
//*              FILE DEFINITION TO BE CONVERTED.                     * 00390000
//*                                                                   * 00400000
//* *** N O T E ***                                                   * 00410000
//*                                                                   * 00420000
//*  THIS PROCEDURE ASSUMES INPUT FROM A PDS MEMBER. OPTIONALLY, IT   * 00430000
//*  MAY ALSO COME FROM A RESULTS COPY (MVS PDS), COPYP (PANVALET),   * 00440000
//*  OR COPYL (LIBRARIAN) STATEMENT. IF SO, YOU MUST UN-COMMENT THE   * 00450000
//*  APPROPRIATE SYSCOPY (MVS PDS), PANDD1 (PANVALET), OR MASTER      * 00460000
//*  (LIBRARIAN) DD STATEMENT IN THE PROCEDURE, SPECIFYING THE        * 00470000
//*  PROPER DATA SET NAME FOR THE LIBRARY USED. PLEASE REFER TO THE   * 00480000
//*  MANUAL FOR DETAILS IN SETTING UP COPY SUPPORT.                   * 00490000
//********************************************************************* 00500000
//STEP01 EXEC RESLTQS,RGN=2M,                                         00510000
//           BLLOAD='BUILDER.BL140.LOADLIB',                          00530000
//           DEFLIB='VISION.BUILDER.DEFLIB',                          00540000
//           MEMBER=FILENAME,                                         00550000
//           RSLTLIB='VISION.RESULTS.FILEDEFS',                       00560000
//           RSLTDEF=FILENAME                                         00570000
```

Figure 5-2 Sample Execution JCL for Advantage VISION:Results Quick Start (Page 2 of 2)

DD Statement Overrides

It is possible through the use of DD overrides to change the specifications for the SYSIN data set from the default of a PDS member. You can do this in one of several ways:

- To specify a sequential data set as SYSIN input, override the SYSIN after procedure invocation with the following JCL DD override:

  ```
  //CONVRT.SYSIN DD DISP=SHR,DSN=USER.SEQ.FIELDEF
  ```

- To specify the SYSIN item to be converted as in-stream data, override SYSIN after the in-stream procedure invocation as follows:

  ```
  //CONVRT.SYSIN    DD *
  ```

  (input to be converted)

- To specify input using z/OS copybook support, un-comment the SYSCOPY DD statement in the procedure, and specify the data set name of a PDS containing the Advantage VISION:Results file definition to be copied. Then override the SYSIN DD statement with in-stream data as follows:

```
//CONVRT.SYSIN    DD *
  COPY membername
```

- To specify input using CA-Panvalet library copybook support, un-comment the PANDD1 DD statement in the procedure, and specify the data set name of the CA-Panvalet library containing the Advantage VISION:Results file definition to be copied. Then override the SYSIN DD statement with in-stream data as follows:

```
//CONVRT.SYSIN    DD *
  COPYP membername
```

- To specify input using CA-Librarian copybook support, un-comment the MASTER DD statement in the procedure, and specify the data set name of the CA-Librarian master file containing the Advantage VISION:Results file definition to be copied. Then override the SYSIN DD statement with in-stream data as follows:

```
//CONVRT.SYSIN    DD *
  COPYL membername
```

Operational Characteristics

The following sections describe optional characteristics of Advantage VISION:Results Quick Start.

Supported Statement Types

Advantage VISION:Results Quick Start processes all valid Advantage VISION:Results FILE and FIELD definition free-form statements, as well as the COPY, COPYP, and COPYL statements. Other valid Advantage VISION:Results processing and reporting statements may be included in the input, but are ignored unless they contain errors. The Advantage VISION:Results FILE statement must be the first statement in the definition.

Converted File Definition

Advantage VISION:Results Quick Start generates file definition statements of the following types:

- FD (File Definition)
- L0 (Field Definition)
- LX (Alternate Name Definition)
- LS (Segment Definition)
- Ln (Field Column Heading Definition)
- Dn (Field Description Definition)

Note that all text fields in the L0, Ln, LX, and Dn generated statements are scanned for the single quote ('), double quote ("), comma (,) and Advantage VISION:Builder system delimiter (from M4PARAMS) characters. If encountered, they are changed to either an underscore (_) or an “at” sign (@).

The generated FD statement is built from the Advantage VISION:Results FILE statement. The file name used is the same as the Advantage VISION:Results file name, and should be used as the "MEMBER" JCL PROC parameter of the PDS member name of the converted definition. If a key length and location are specified in the Advantage VISION:Results definition, it is saved and used to assign a key field to the converted FD.

If no key location is specified on the Advantage VISION:Results FILE statement, the first field output in the converted FD is arbitrarily designated as the key, and a diagnostic warning message is issued so that you can inspect and change the converted FD, if necessary. Record format conversion is accomplished where possible.

If the record format from the Advantage VISION:Results definition could not be translated, a question mark (?) is inserted in the converted record format field, and a diagnostic warning message is issued so that this field can be reviewed in the converted FD.

For record and block size, the Advantage VISION:Results characters per record and characters per block fields are specified as the record size and buffer size fields, respectively, unless the record format is FIXED or ISAM, in which case the record size is set and a records per block is calculated for the FD statement.

The generated LS statement always has a name of SEGMENT1, since Advantage VISION:Results file definitions represent flat, not hierarchical, files. It is also assigned a segment number of 1 and a level number of 1.

The generated L0 statements always contain a generated field name of a 1-byte prefix (the character F), followed by a 7-digit number that is incremented by one for each field defined. The Advantage VISION:Results field name, which can be up to 50 characters in length, is used to generate an alternate name for the field (see LX statement explanation below).

If the name is greater in length than the 30 bytes allowed for alternate names, only the first 30 characters are used, and the entire 50-byte name is used as the field description (for more information, see the Dn statement explanation below). When this happens, the field definition should be checked for duplicate names using the Duplicate Name Check function of the Workbench for ISPF IMPORT option.

Field location, length, type, output edit length, and rounding information are translated directly to the L0 statement from the Advantage VISION:Results definition. Edit codes are translated as closely as possible. A field location of 4 question marks (????) is generated if the Advantage VISION:Results definition uses 5 bytes without a leading zero to define the field location.

The generated Ln statements for column headings come from the Advantage VISION:Results field definition column heading information, if present. If no column heading is specified in the input Advantage VISION:Results definition, then the field name is used as the column heading. Since column headings are to be specified in 16-byte lines, you should check the converted definition in case the 30-byte Advantage VISION:Results column heading has been "broken up" inappropriately during the conversion.

The generated Dn statements (field descriptions) are based upon the Advantage VISION:Results field name if it was greater than 30 bytes. If not, the column heading is used as the field description.

The generated LX statements for alternate names are based upon the first 30 characters in the Advantage VISION:Results field name. Take note that if there is more than one field that is greater than 30 bytes in the Advantage VISION:Results definition, there is the possibility of duplicate names being generated in the converted FD. You should then use the Workbench for ISPF Duplicate Name Check function upon the converted FD.

Member Naming Conventions

The member name of the converted definition is always taken from the JCL "MEMBER" PROC parameter. If this name exists in the output PDS, it is overwritten. For ease of use with the Workbench for ISPF, the PDS name must always match the file name of the file definition being converted.

SYSPRINT Listing

The SYSPRINT listing contains the following information:

- A listing of the Advantage VISION:Results input statements.
- Error messages for any input statements that are invalid.
- Informational or warning messages about the converted FD statements.

COPYP and COPYL Support

In order to use the COPYP (CA-Panvalet copy) and COPYL (CA-Librarian copy) statements, additional setup must be done prior to execution of Advantage VISION:Results Quick Start. This setup involves linking the appropriate interface routines with Advantage VISION:Results Quick Start for CA-Panvalet or CA-Librarian access.

Installing the Advantage VISION:Results Quick Start Routines (Optional)

If you plan to use Advantage VISION:Results Quick Start to convert definitions that are stored in a CA-Librarian or CA-Panvalet library, the appropriate interface routine must first be link edited with Advantage VISION:Results Quick Start. This is accomplished by running one of the supplied JCL procedures in the WORKLIB PDS.

Link Edit CA-Librarian Support

The PDS dataset (...PREP.JCLCNTL) contains member BLXRSQ#1 to link edit support for accessing Advantage VISION:Results file definitions stored in a CA-Librarian master file with Advantage VISION:Results Quick Start. Figure 5-3 shows the JCL procedure. Modify the procedure variables as appropriate and run the job to activate this interface. A return code of 4, coupled with the warning message IEW0461 can be ignored.

```
//* MEMBER BLXRSQ#1                                                    00010000
//*********************************************************************00020000
//* LINK LIBRARIAN INTERFACE MODULES WITH RESULTS QUICK START.      *  00030000
//*********************************************************************00040000
//LBLNK  PROC &BLLOAD=,                                                00050000
//           LIBLOAD=                                                  00060000
//LINK   EXEC PGM=IEWL,REGION=1M,PARM='LIST,MAP,LET,NCAL'              00070000
//SYSLIB   DD DUMMY                                                    00080000
//SYSPRINT DD SYSOUT=*                                                 00090000
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))                             00100000
//LIBSYS   DD DISP=SHR,DSN=&LIBLOAD                                    00110000
//SYSLMOD  DD DISP=SHR,DSN=&BLLOAD                                     00120000
//       PEND                                                          00130000
//*********************************************************************00140000
//* BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING      *  00150000
//* INFORMATION:                                                    *  00160000
//*                                                                 *  00170000
//*   LOADLIB - NAME OF YOUR VISION:BUILDER LOAD LIBRARY.           *  00180000
//*   LIBLOAD - NAME OF YOUR LIBRARIAN SYSTEM LOAD LIBRARY.         *  00190000
//*********************************************************************00200000
//LIBLINK EXEC LBLNK,                                                  00210000
//            LOADLIB='BUILDER.BL140.LOADLIB',                         00220000
//            LIBLOAD='LIBRARN.SYSTEM.LOADLIB'                         00230000
//LINK.SYSLIN  DD *                                                    00240000
  INCLUDE LIBSYS(FAIRCLS)                                            00250000
  INCLUDE LIBSYS(FAIROPN)                                            00260000
  INCLUDE LIBSYS(FAIRREC)                                            00270000
  INCLUDE LIBSYS(FAIRMOD)                                            00280000
  INCLUDE LIBSYS(FAIRERR)                                            00290000
  INCLUDE LIBSYS(FAIRLOC)                                            00300000
  INCLUDE LIBSYS(FAIRNTE)                                            00310000
  INCLUDE LIBSYS(FAIRPNT)                                            00320000
  INCLUDE LIBSYS(FAIRSCAN)                                           00330000
  INCLUDE LIBSYS(FAIRSEC)                                            00340000
  INCLUDE SYSLMOD(DYL280LX)                                          00350000
    ENTRY DYL280L                                                    00360000
    NAME DYL280L(R)                                                  00370000
```

Figure 5-3 JCL to Link CA-Librarian Support

Link Edit CA-Panvalet Support

The PDS dataset (...PREP.JCLCNTL) contains member BLXRSQ#2 to link edit support for accessing Advantage VISION:Results file definitions stored in a CA-Panvalet library with Advantage VISION:Results Quick Start. Figure 5-4 shows the JCL procedure. Modify the procedure variables as appropriate and run the job to activate this interface.

```
//* MEMBER BLXRSQ#2                                                     00010000
//*********************************************************************00020000
//* LINK PANVALET INTERFACE MODULES WITH RESULTS QUICK START.       *  00030000
//*********************************************************************00040000
//PNLNK  PROC &BLLOAD=,                                                 00050000
//            PANLOAD=                                                  00060000
//LINK   EXEC PGM=IEWL,REGION=1M,PARM='LIST,MAP,LET,NCAL'               00070000
//SYSLIB   DD DUMMY                                                     00080000
//SYSPRINT DD SYSOUT=*                                                  00090000
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))                              00100000
//LIBSYS   DD DISP=SHR,DSN=&PANLOAD                                     00110000
//SYSLMOD  DD DISP=SHR,DSN=&BLLOAD                                      00120000
//       PEND                                                           00130000
//*********************************************************************00140000
//* BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING      *  00150000
//* INFORMATION:                                                    *  00160000
//*                                                                 *  00170000
//*   LOADLIB - NAME OF YOUR VISION:BUILDER LOAD LIBRARY.           *  00180000
//*   PANLOAD - NAME OF YOUR PANVALET SYSTEM LOAD LIBRARY.          *  00190000
//*********************************************************************00200000
//PANLINK EXEC PNLNK,                                                   00210000
//             LOADLIB='BUILDER.BL140.LOADLIB',                         00220000
//             PANLOAD='PANVALET.SYSTEM.LOADLIB'                        00230000
//LINK.SYSLIN  DD *                                                     00240000
  INCLUDE LIBSYS(PAM)                                                   00250000
  INCLUDE SYSLMOD(DYL280PX)                                             00260000
  ENTRY DYL280P                                                         00270000
  NAME DYL280P(R)                                                       00280000
```

Figure 5-4 JCL to Link CA-Panvalet Support

Messages

All diagnostic and informational messages from Advantage VISION:Results Quick Start begin with the characters DYL-. For information about diagnostic messages relating to errors in the input Advantage VISION:Results file definition, see the *Advantage* VISION:Results *Messages Guide*. Messages relating to the converted field definition are listed below. The messages are placed in the SYSPRINT data set.

DYL-1800 **If encountered, the system delimiter of '*' will be replaced with '*'.**

Informational message. This indicates which character will replace the system delimiter character if it is encountered in a name field, field description, or column heading.

DYL1801 **Unable to convert 5-byte Advantage VISION:Results record size to Advantage VISION:Builder 4-byte format.**

Warning message. This message is generated when a 5-byte Advantage VISION:Results record size is encountered. Advantage VISION:Builder allows only 4 bytes for record size, so the record size field should be checked in the converted definition. Consider leaving the record size field in FD blank, and using the buffer size field instead.

DYL1802 **Spanned record type buffer size has been estimated at 34K. Check the FD.**

Warning message. Spanned record support for definitions is specified by having a buffer size of greater than 32K, whereas spanned records in Advantage VISION:Results use a field type character of S. You should check this field in the output FD for accuracy.

DYL1803 **Unable to convert the Advantage VISION:Results record format. Check the FD.**

Warning message. The record format in the Advantage VISION:Results field definition could not be translated into Advantage VISION:Builder format. You should check the converted file definition for a valid record format specification.

DYL1804 **Unable to load module M4PARAMS. The pound sign will be assumed as the system delimiter.**

Warning message. A z/OS LOAD was issued for the M4PARAMS load module, and the module was not available in the STEPLIB concatenation. Make sure the installation load library has been made available in STEPLIB. The utility continues to run by assuming the system delimiter has not been changed from the default of #.

DYL1805 **Unable to convert 5-byte field location for field ********. Check the FD.**

Warning message. Advantage VISION:Results definitions allow for 5 bytes in the field location, and Advantage VISION:Builder only allows 4 bytes. If the Advantage VISION:Results uses all 5 bytes, and the first is not a leading zero, the utility is unable to convert the location. You should check this in the converted FD. The field name in question is inserted into the message where the asterisks appear.

DYL1806 **No key was specified: field ******** was arbitrarily designated as the key. Check the FD.**

Warning message. All definitions are required to have one key designated as a key field for the file, but Advantage VISION:Results does not allow a key specification for certain file types. When this occurs, the utility arbitrarily designates the first field converted as the key field. You should check and update the converted FD if this designation is incorrect.

DYL1807 **The key information supplied did not match a defined field. Check the FD.**

Warning message. The Advantage VISION:Results definition did have a key field length and location specified, but the utility was unable to match this with a defined field. The converted FD has not been designated with a key field. Check the converted FD and specify a key field with Workbench for ISPF.

DYL1808 **Key field ******** was designated as the key based upon the key information in the Advantage VISION:Results FD.**

Informational message. The Advantage VISION:Results definition did have a key field length and location specified, and the utility was able to match this with a defined field. The key field name is replaced in the message where the asterisks occur.

Return Codes

When run in batch, Advantage VISION:Results Quick Start returns to the operating system with one of the following return codes. These return codes appear as the COND CODE value in the operating system JCL log. Return codes of 0 or 4 mean that the utility continued executing until normal end-of-file on the SYSIN data set. A return code of 8 indicates an error from which no recovery could be made and Advantage VISION:Results Quick Start terminated without creating a converted field definition.

Return Code	Explanation
0	Normal return code. No errors were encountered, and only informational messages were issued. A converted file definition in Advantage VISION:Builder format has been created.
4	Warning return code. One or more warning messages were issued during definition conversion. A converted file definition has been created, but should be closely checked for accuracy using Workbench for ISPF. The messages in the SYSPRINT listing indicate which items should be checked.
8	Error return code. Errors were detected in the syntax or content of the input Advantage VISION:Results FILE definition statements. The statements were not converted. Correct the errors on input as indicated by the messages in the SYSPRINT listing, and rerun Advantage VISION:Results Quick Start.

Chapter 6

Using Advantage VISION:Inquiry Quick Start

Advantage VISION:Inquiry Quick Start generates COMLIB file definitions from existing Advantage VISION:Inquiry MAPGENs. During the utility processing, each Advantage VISION:Inquiry map that is processed becomes a separate file definition. The Advantage VISION:Inquiry map information is converted into file, segment, and field information in the file definition.

Note: If an existing member name is specified, the existing member is replaced when the new definition is saved. If a new member name is specified, then a new member is created.

You can invoke Advantage VISION:Inquiry Quick Start either as an import function within the Definition Processor or as a stand-alone batch execution. The remainder of this chapter describes the use of Advantage VISION:Inquiry Quick Start as a batch utility.

Multiple file definitions can be generated in a single Advantage VISION:Inquiry Quick Start batch execution. Each generated file definition is stored as a separate member in the indicated source definition library. The specified file name is used as the member name when the generated definition is saved in the source definition library.

After a file definition has been created, any additional information that is needed can be added using the Workbench for ISPF.

Advantage VISION:Inquiry Quick Start can also run from Workbench for ISPF. The IMPORT option points you to a menu for selecting the Quick Start utility you want to run. The subsequent panels prompt you for the information needed to run the utility. After the information is gathered, the utility is run immediately and the output is displayed for you to browse.

Flow Diagram

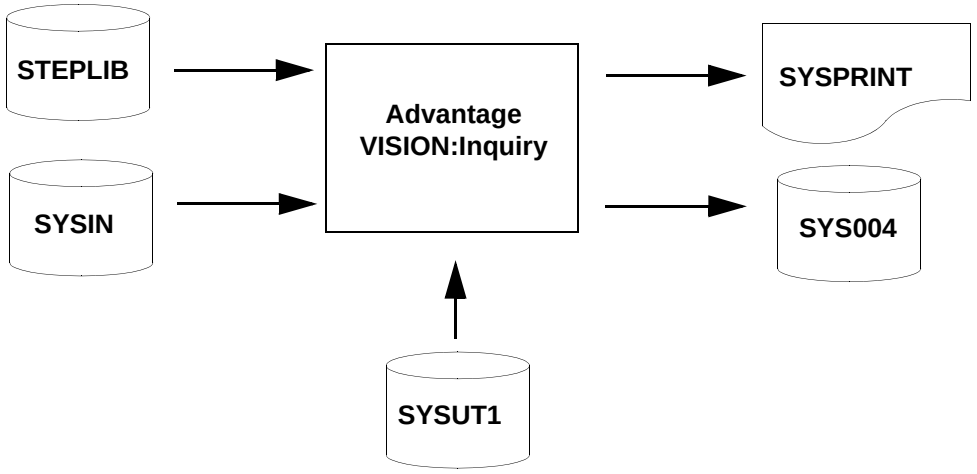

Figure 6-1 Advantage VISION:Inquiry Quick Start Flow Diagram

As shown in Figure 6-1, Quick Start uses the following three input data sets:

STEPLIB Specifies the Advantage VISION:Builder r15 load library.

SYSIN Specifies the appropriate Advantage VISION:Inquiry Quick Start control statements.

SYSUT1 Contains the unloaded form of the Advantage VISION:Inquiry system database. For information on how to create the unloaded form of the system database, see the *Advantage VISION:Inquiry Reference Guide*.

Advantage VISION:Inquiry Quick Start produces the following two output files:

SYSPRINT Contains a summary report on the file definition generation process.

SYS004 Generated file definitions are written to the partitioned data set pointed to by the ddname SYS004. This must be a partitioned data set.

Utility Execution

Figure 6-2 contains the JCL required to execute Advantage VISION:Inquiry Quick Start. This JCL contains an in-stream procedure followed by JCL statements to execute the procedure, as well as sample control statements. You can find the JCL in the PDS data set (...PREP.JCLCNTL) member BLXINQ#1.

At a minimum, you must make the following changes before submitting this JCL:

- Supply a job card.
- Supply values for the procedure variables detailed in the table on page 6-4.
- Provide a DD statement for SYSIN. This data set contains the required Advantage VISION:Inquiry Quick Start input control statements which are used to control the generation of file definitions. For more information, see FILEGEN Control Statement.
- To conform to your standards, you may need to make additional modifications.

Variable Name	Description
BLLOAD	Specifies the name of the Advantage VISION:Builder r15 load library.
ULSYSDB	Specifies the name of your Advantage VISION:Inquiry unloaded system database.
DEFLIB	Specifies the name of your definition library.

Note: If the SYSIN data set is specified as a DUMMY or if no FILEGEN statements are specified in the input stream (empty SYSIN data set), all MAPGENs in the unloaded system database are converted in one invocation of Advantage VISION:Inquiry Quick Start.

```
//* MEMBER BLXINQ#1                                                    00010000
//*********************************************************************** 00020000
//* UTILITY TO CONVERT VISION:Inquiry FILE DEFINITIONS INTO           * 00030000
//* VISION:BUILDER OR VISION:INFORM FORMAT FILE DEFINITIONS.          * 00040000
//* THE VISION:Inquiry FILE DEFINITIONS MUST COME FROM AN             * 00050000
//* VISION:Inquiry UNLOADED SYSTEM DATABASE FILE.  SEE YOUR           * 00060000
//* VISION:Inquiry TECHNICAL REFERENCE MANUAL FOR INFORMATION ON      * 00070000
//* HOW TO CREATE AN UNLOADED COPY OF THE SYSTEM DATABASE.            * 00080000
//*                                                                   * 00090000
//* THIS UTILITY MAY ALSO BE INVOKED UNDER TSO/ISPF USING THE         * 00100000
//* VISION:INFORM DEFINITION PROCESSOR IMPORT FUNCTION.               * 00110000
//*********************************************************************** 00120000
//INQRYQS PROC RGN=2M,                                                00130000
//            BLLOAD=,                                                00140000
//            ULSYSDB=,                                               00150000
//            DEFLIB=                                                 00160000
//INQRYQS EXEC PGM=INQRYQS,REGION=&RGN                                00170000
//STEPLIB   DD DISP=SHR,DSN=&BLLOAD                                   00180000
//SYSPRINT  DD SYSOUT=*                                               00200000
//SYSUT1    DD DISP=SHR,DSN=&ULSYSDB                                  00210000
//SYS004    DD DISP=OLD,DSN=&DEFLIB                                   00220000
//        PEND                                                        00230000
//*********************************************************************** 00240000
//*  FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE. BEFORE YOU    * 00250000
//*  RUN THIS PROCEDURE, SPECIFY:                                     * 00260000
//*                                                                   * 00270000
//*    RGN     - THE REGION SIZE; DEFAULT IS 2M.                      * 00280000
//*    BLLOAD  - THE BUILDER LOAD LIBRARY.                            * 00290000
//*    ULSYSDB - THE UNLOADED VISION:Inquiry SYSTEM DATABASE FILE.    * 00300000
```

Figure 6-2 Sample Execution JCL for VISION:Inquiry Quick Start (Page 1 of 2)

```
//*    DEFLIB  - THE VISION:INFORM DEFINITION LIBRARY.              * 00310000
//*********************************************************************** 00320000
//STEP01  EXEC INQRYQS,RGN=2M,                                        00330000
//             BLIQAD='BUILDER.BL135.LQADLIB',                        00340000
//             ULSYSDB='VISION.INQUIRY.UNLOADED.SYSDBASE',            00350000
//             DEFLIB='VISION.BUILDER.DEFLIB'                         00360000
//SYSIN     DD *                                                      00370000
  FILEGEN NAME=VSHPLANT,FLDPREFX=PLT                                  00380000
  FILEGEN NAME=SALARIES,FLDPREFX=SAL                                  00390000
```

Figure 6-2 Sample Execution JCL for VISION:Inquiry Quick Start (Page 2 of 2)

FILEGEN Control Statement

Advantage VISION:Inquiry Quick Start uses the following control statement as input:

FILEGEN — The FILEGEN statement triggers the start of a new file definition. The NAME parameter specified on the FILEGEN statement is used as both the file name and the member name when the generated file definition is saved.

Coding Rules

When writing Advantage VISION:Inquiry Quick Start control statements, you must observe the following rules:

- Each statement must be on a single line and must begin with the FILEGEN command name.
- Each control statement may contain keyword parameters. Keyword parameters can be specified in any order and must be separated by commas. Embedded blanks are not allowed between parameters.
- The NAME keyword is required on each statement. All other keywords are optional.
- Blank lines within the SYSIN file are not accepted.
- Columns 1 through 71 of the control statement are scanned.
- Columns 72 through 80 are ignored.

Syntax

Note: If an existing member name is specified, the existing member is replaced. If a new member name is specified, a new member is created.

FILEGEN	Required. FILEGEN is a control statement command. It identifies the control statement type. The FILEGEN control statement triggers the start of a new file definition.
NAME	Required. The NAME parameter specifies the name of the MAPGEN that is being converted. A map name can be 1–8 characters. If the corresponding MAPGEN is not found in the unloaded VISION:Inquiry system database, a message is printed indicating that no conversion took place. When the generated file definition is saved, the file name is also used as the member name unless the NEWNAME keyword is specified (for more information, see the NEWNAME keyword below).
BUFFSIZE	Optional. The BUFFSIZE parameter defines the size of the buffer needed to process a logical record for IMS. Enter a number from 1 to 32760. Multiples of 1024 can be entered as *nnnn*K where *nnnn* is a number between 1 to 9999. This parameter is only used for IMS definitions and will be ignored for all other file definition types.

FLDPREFX	Optional. The FLDPREFX parameter specifies the 1 to 3-character prefix for generating primary field names in the file definition. Primary field names are required in a file definition and must be assigned a unique 1- to 8-character name. Because Advantage VISION:Inquiry field names can be longer than 8 characters, Advantage VISION:Inquiry Quick Start automatically generates a unique 8-character primary field using the FLDPREFX value followed by a generated field number. If the FLDPREFX parameter is omitted, the default prefix is F and the generated primary field names have the format F*nnnnnnn* where *nnnnnnn* is a number from 0000001 to 9999999. For more information, see Field Name Generation.
NEWNAME	Optional. The NEWNAME parameter specifies a 1–8 character name that becomes the name of the converted file definition. This name is used as both the file definition name and the member name in the definition library. If this keyword is omitted, the existing name as specified using the NAME keyword is used as the file definition name and member name.

Conversion Rules

The following sections describe how the Quick Start utility generates various file definition paramaters.

Generated File Definition

This utility generates a file definition that can be edited and validated by the Workbench for ISPF.

File information	The Glossary specification can be added as appropriate and the buffer size specification may need to be adjusted for IMS files.
Segment information	Additional segment information such as segment (row) order and suppress duplication can be provided as needed. Segment key fields will have been identified during the conversion process based upon the information contained in the MAPGEN. If it is appropriate for a segment to have more than one key field specified, you may need to adjust the key field specifications. Segment count fields will have been identified for VSAM hierarchical files as appropriate.

Field information	Primary field name assignments can be modified if wanted (for more information, see Field Name Generation). Additional field information such as rounding, editing, column headings, and automatic table lookup results should be provided as needed. Field description information is filled in from the description information in the Advantage VISION:Inquiry MAPGEN. Since Advantage VISION:Inquiry MAPGENs do not contain column heading or equivalent information, no column heading information is present in the converted file definition.

Field Name Generation

All fields within a file definition must be assigned a unique 1–8 character name. This name is referred to as the primary field name.

When building a file definition from an Advantage VISION:Inquiry MAPGEN, a unique 1–8 character primary field name must be assigned to each field. Since Advantage VISION:Inquiry field names can be longer than 8 characters, Advantage VISION:Inquiry Quick Start automatically generates a unique primary field name using the FLDPREFX parameter value on the FILEGEN statement followed by a generated field number. If the FLDPREFX parameter is omitted, the default prefix is F and the generated primary field names have the format F*nnnnnnn* where *nnnnnnn* is a number from 0000001 to 9999999. The original name is used as the alternate name in this case.

After the file definition has been created, it can be refined using Workbench for ISPF. Column heading information can be added and other changes applied as appropriate.

Messages

The following table lists Advantage VISION:Inquiry Quick Start messages and their explanations. The Return Code for each message resulting in a return code greater than 0 is listed with the explanation. The question mark (?) in the messages below represents a parameter that is substituted when the message is issued.

? is an Invalid keyword.

Identifies the keyword of a FILEGEN statement that is invalid.
RC=16

? Statements generated.

Information message indicating the number of statements generated while converting a file definition.

Can't open file for ?

Identifies a map for which the SYS004 file cannot be opened.
RC=20

Can't open SYS004 file.

RC=20

Can't open SYSIN file.

RC=20

Can't open SYSUT1 file.

RC=20

Conversion aborted; Control Statement errors.

One or more errors where encountered while scanning the FILEGN statements. RC=16

Converting File Definition ?

Information message identifying the file definition being converted.

Definition ? is for Database Type ?

The Inquiry definition is for a database type that is not supported by Advantage VISION:Builder/Advantage VISION:Inform. RC=4

FIELD DESC. record out of sequence.

A FIELD DESCRIPTION record was encountered when not expected. RC=8

Field not found for Description ?

The related field was not found for a field description record. RC=8

FIELD record out of sequence.

A FIELD record was encountered when not expected. RC=8

File ? present but not selected.

Information message indicating that the specified definition was present on the unloaded database file, but was not selected on any FILEGEN statement.

File Definition ? not converted.

Message indicating that the specified definition was not converted, probably because the name keyword on the FILEGEN statement did not match any definition in the unloaded system database. RC=4

Insufficient memory.

There was insufficient memory available to complete the conversion of a definition. RC=12

MAP ? has too many segments.

The specified definition has more than 100 segments. RC=8

MAP ? Segment ? has too many fields.

The specified segment of the definition has more than 1000 fields. RC=8

MAPGEN Descriptor record out of sequence.

A MAPGEN Descriptor record was encountered when not expected. RC=8

Processing completed, Return code = ?

Information message indicating that the conversion program has terminated with the specified return code.

RECORD record out of sequence.

A RECORD information record was encountered when not expected. RC=8

Required NAME keyword/operand not found.

A FILEGEN statement was specified without the required NAME keyword or no operand was provided for the keyword. RC=16

Segment not found for field ?

The corresponding segment was not found for a field record. RC=8

SEGMENT record out of sequence.

A SEGMENT record was encountered when not expected. RC=8

Statement is not a FILEGEN statement.

A control statement was read that was not a FILEGEN statement. RC=16

Statement syntax error.

A FILEGEN statement syntax was in error. No operand following a keyword; keyword not followed by an equal sign; RC=16

This Database Type is not supported by Advantage VISION:Builder/ Advantage VISION:Inform.

Follow-on message to message "Definition ? is for Database Type ?". RC=4

Return Codes

Advantage VISION:Inquiry Quick Start generates the following return codes:

Return Code	Explanation
0	Successful completion. All selected definitions were converted.
4	The definition for one or more FILEGEN statements was not converted. For more information, see the print output listing regarding which conversions completed successfully and which definitions did not.
8	The records for a selected definition in the Advantage VISION:Inquiry unloaded system database were not in the expected sequence or an internal limit was exceeded. The conversion in progress at the time was terminated and conversion of the next definition (if any) was attempted. One or more definitions may have been converted successfully. For more information, see the print output listing.
12	Insufficient memory was available to complete a conversion. One or more definitions may have been converted successfully. For more information, see the print output listing.
16	A syntax error was found on one or more FILEGEN statements. Processing was terminated after all FILEGEN statements were read. No conversion was attempted.

Return Code	Explanation <$paranum
20	The SYSIN, SYSUT1, or SYS004 file did not open successfully. Processing was terminated immediately. For more information, see the print output listing.
24	The SYSPRINT file did not open successfully. Processing was terminated immediately.

Appendix

A

ASL Examples

This appendix contains a simple Advantage VISION:Builder application showing the application code, output reports, and run-time listing.

Code

```
CONTROL TERM,
        SORT INTERNAL,
        DB2 D81A INM4CALL,
        RPTMSGS NO         ;Suppress Report Info & Warning Messages
;
FILE MASTER INPUT, KEYNAME EMPNO,
     SQL "SELECT * FROM DSN8810.EMP ORDER BY EMPNO"
FILE REPO
;
MAIN: PROC
NAME:     FIELD V 22
;
; Combine Elements of Name into One Field
;
  COMBINE LASTNAME ',', STORE T.NAME
  COMBINE T.NAME FIRSTNME MIDINIT, BLANKS 1, STORE T.NAME
;
  REPORT EMPNO T.NAME BIRTHDATE
    ITEM BIRTHDATE PIC P'Wwwwwwwwwz, Mmm, DD, YYYY'
    FORMAT DATEFMT DATE, WIDTH 80,
           METHOD PLAINTEXT, DDNAME REPOUT1
    TITLE 'Report Showing Birth Dates of all Employees'
  END REPORT
END PROC
;
; Following Report Called Once at EOF
;
REPORT F.TODAY F.TODAYX F.ISDATE F.DATE,
       F.JULIAN F.JULANX F.LILIAN,
       TYPE EOF
  FORMAT HEADINGS NAME, WIDTH 80,
         METHOD PLAINTEXT, DDNAME REPOUT2
  TITLE 'Report Showing Use of All Date Flags'
END REPORT
```

Reports

JUL 29, 2004 Report Showing Birth Dates of all Employees

EMPNO	NAME	BIRTHDATE
000010	HAAS, CHRISTINE I	Monday, Aug, 14, 1933
000020	THOMPSON, MICHAEL L	Monday, Feb, 02, 1948
000030	KWAN, SALLY A	Sunday, May, 11, 1941
000050	GEYER, JOHN B	Tuesday, Sep, 15, 1925
000060	STERN, IRVING F	Saturday, Jul, 07, 1945
000070	PULASKI, EVA D	Tuesday, May, 26, 1953
000090	HENDERSON, EILEEN W	Thursday, May, 15, 1941
000100	SPENSER, THEODORE Q	Tuesday, Dec, 18, 1956
000110	LUCCHESI, VINCENZO G	Tuesday, Nov, 05, 1929
000120	O'CONNELL, SEAN	Sunday, Oct, 18, 1942
000130	QUINTANA, DOLORES M	Tuesday, Sep, 15, 1925
000140	NICHOLLS, HEATHER A	Saturday, Jan, 19, 1946
000150	ADAMSON, BRUCE	Saturday, May, 17, 1947
000160	PIANKA, ELIZABETH R	Tuesday, Apr, 12, 1955
000170	YOSHIMURA, MASATOSHI J	Friday, Jan, 05, 1951
000180	SCOUTTEN, MARILYN S	Monday, Feb, 21, 1949
000190	WALKER, JAMES H	Wednesday, Jun, 25, 1952
000200	BROWN, DAVID	Thursday, May, 29, 1941
000210	JONES, WILLIAM T	Monday, Feb, 23, 1953
000220	LUTZ, JENNIFER K	Friday, Mar, 19, 1948
000230	JEFFERSON, JAMES J	Thursday, May, 30, 1935
000240	MARINO, SALVATORE M	Wednesday, Mar, 31, 1954
000250	SMITH, DANIEL S	Sunday, Nov, 12, 1939
000260	JOHNSON, SYBIL V	Monday, Oct, 05, 1936
000270	PEREZ, MARIA L	Tuesday, May, 26, 1953
000280	SCHNEIDER, ETHEL R	Saturday, Mar, 28, 1936
000290	PARKER, JOHN R	Tuesday, Jul, 09, 1946
000300	SMITH, PHILIP X	Tuesday, Oct, 27, 1936
000310	SETRIGHT, MAUDE F	Tuesday, Apr, 21, 1931
000320	MEHTA, RAMLAL V	Thursday, Aug, 11, 1932
000330	LEE, WING	Friday, Jul, 18, 1941
000340	GOUNOT, JASON R	Monday, May, 17, 1926
200010	HEMMINGER, DIAN J	Monday, Aug, 14, 1933
200120	ORLANDO, GREG	Sunday, Oct, 18, 1942
200140	NATZ, KIM N	Saturday, Jan, 19, 1946
200170	YAMAMOTO, KIYOSHI	Friday, Jan, 05, 1951
200220	JOHN, REBA K	Friday, Mar, 19, 1948
200240	MONTEVERDE, ROBERT M	Wednesday, Mar, 31, 1954
200280	SCHWARTZ, EILEEN R	Saturday, Mar, 28, 1936
200310	SPRINGER, MICHELLE F	Tuesday, Apr, 21, 1931
200330	WONG, HELENA	Friday, Jul, 18, 1941
200340	ALONZO, ROY R	Monday, May, 17, 1926

07/29/04 Report Showing Use of All Date Flags

TODAY	TODAYX	ISDATE	DATE	JULIAN	JULANX	LILIAN
072904	07292004	20040729	JUL 29, 2004	04211	2004211	07/29/2004

Listing

```
JUL 29, 2004  18.38.41                                                                                    PAGE      1
                                  ******************************************************
                                  *    Advantage VISION:Builder for z/OS - Version 15.0   *
                                  *                                                       *
                                  * (C) 2005 Computer Associates International, Inc. (CA) *
                                  ******************************************************
VISION:Builder 15.0 GA                          Enabled for IBM Language Environment              Build Number = 104166-19.38.25.4
=============================================================================================================================
=                                                                                                                           =
=                    I N S T A L L A T I O N    P A R A M E T E R S    ( M 4 P A R A M S , M A R K L I B P )                =
=                                                                                                                           =
=  SYSTEM DELIMITER:   #          PAGE HEIGHT:       66         M4LIST WIDTH:      132        DEF WIDTH OF PAGE:  0        =
=                                                                                                                           =
=  AUTO GRAND:         N          HEADING CHAR:      -          SUBTITLE REPEAT:   N          INVALID FIELD:      *        =
=                                                                                                                           =
=  MISSING FIELD:      -          NON-EDIT FIELD:    +          PERCENT CHAR:      %          LEFT SEPARATOR:     (        =
=                                                                                                                           =
=  RIGHT SEPARATOR:    )          SINGLE SEPARATOR:  ,          SOURCE SPACING:    1          PRINT MESSAGES:     Y        =
=                                                                                                                           =
=  CONSOLE MESSAGES:   N          M4REPO BLOCKSIZE:  4,096      INPUT I/O BUFFERS: 2          OUTPUT I/O BUFFERS: 1        =
=                                                                                                                           =
=  SNGL-STEP STORAGE:  8,192      SNGL-STEP SORTSIZE: 524,288   DIGIT SELECT CHAR: 9          ZERO SUPPRESS CHAR: Z        =
=                                                                                                                           =
=  CURRENCY CHAR:      $          PLUS CHAR:         +          MINUS CHAR:        -          CHECK PROTECT CHAR: *        =
=                                                                                                                           =
=  DECIMAL CHAR:       .          GROUPING CHAR:     ,          PRIMARY PLOT CHAR: X          SECONDARY PLOT CHAR:*        =
=                                                                                                                           =
=  FIT PLOT CHAR:      .          HORIZONTAL AXIS:   _          HORIZONTAL HASH:   |          VERTICAL AXIS:      |        =
=                                                                                                                           =
=  VERTICAL HASH:      -          MINUTES/HOUR:      60         SECONDS/MINUTE:    60         TIME DELIMITER:     HH:MM:SS =
=                                                                                                                           =
=  DATE FORMAT:        MM DD, YYYY TODAY FORMAT+DELIM: MM/DD/YY ISDATE DELIMITER:   YYYY-MM-DD JULIAN DELIMITER:  YY.DDD   =
=                                                                                                                           =
=  SORT PROGRAM CODE:  2          MINCORE VALUE:     12 K       ALT M4LIST WIDTH:  132        ALT DEF W/OF PAGE:  0        =
=                                                                                                                           =
=  MAX LINES OF TRACE: 1,024      ITEM TRACKING:     0          SUPPRESS NDS REPT?: N         DEFAULT MAXGETMN:   1,024 K  =
=                                                                                                                           =
=  CONDITION CODE 1:   0          CONDITION CODE 2:  4          CONDITION CODE 3:  8          CONDITION CODE 4:   16       =
=                                                                                                                           =
=  ISAM INDEX INCORE:  N          M4LIB BLKG FACTOR: 0          M4LIB RESERVE:     0          M4LIB COMPONENT:    C5.0 B/V =
=                                                                                                                           =
=  DEFAULT F/P AMODE:  31         M4LIB COMPRESSION?: N         M4LIB COMPRESS MIN: 507                                    =
=                                                                                                                           =
=============================================================================================================================
```

```
JUL 29, 2004  18.38.41                                                                                    PAGE      2
     CONTROL TERM,                                                                                                      1
             SORT INTERNAL,                                                                                             2
             DB2 D81A INM4CALL,                                                                                         3
             RPTMSGS NO         ;Suppress Report Info & Warning Messages                                                4
     ;                                                                                                                  5
     FILE MASTER INPUT, KEYNAME EMPNO,                                                                                  6
          SQL "SELECT * FROM DSN8810.EMP ORDER BY EMPNO"                                                                7
     FILE REPO                                                                                                          8
     ;                                                                                                                  9

File Glossary derived from SQL Clause on FILE MASTER Statement:
                                Field                        Dec.  Edit
DB2 Column                      Name      Type  Length  Plcs  Len.  Location
-------------------------------------------------------------------------------
EMPNO                           SQLM0001    C       6           6       1  Key
FIRSTNME                        SQLM0002    V      12                +
MIDINIT                         SQLM0003    C       1           1       7
LASTNAME                        SQLM0004    V      15                +
WORKDEPT                        SQLM0005    C       3           3       8
```

```
PHONENO                         SQLM0006    C       4          4       11
HIREDATE                        SQLM0007    D       4         10       15
JOB                             SQLM0008    C       8          8       19
EDLEVEL                         SQLM0009    F       2          7       27
SEX                             SQLM0010    C       1          1       29
BIRTHDATE                       SQLM0011    D       4         10       30
SALARY                          SQLM0012    P       5     2   13       34
BONUS                           SQLM0013    P       5     2   13       39
COMM                            SQLM0014    P       5     2   13       44
End of Glossary; Buffer Size = 83
```

```
================================================================================
=                                                                              =
=                                  F I L E   S U M M A R Y                     =
=                                                                              =
= JOBNAME: REDJO04S                    STEPNAME: M4RUN                         =
=                                                                              =
=  FILE NAME: M4OLD                    TABLE:          .                       =
=  FILE NAME: M4LIB                    DSNAME: REDJO04.DEVEL2.M4LIB            =
=                                                                              =
================================================================================
```

```
JUL 29, 2004  18.38.41                                                PAGE      3
                                      ****************************
                                      *  PROC NAME - MAIN        *
                                      *  INPUT STREAM PROCEDURE  *
                                      ****************************
    MAIN: PROC                                                    (00040000)   10
    NAME:     FIELD V 22                                          (00130000)   11
    ;                                                             (00320000)   12
    ; Combine Elements of Name into One Field                                   13
    ;                                                                          14
      COMBINE LASTNAME ',', STORE T.NAME                                       15
      COMBINE T.NAME FIRSTNME MIDINIT, BLANKS 1, STORE T.NAME                  16
    ;                                                                          17
      REPORT EMPNO T.NAME BIRTHDATE                                            18
        ITEM BIRTHDATE PIC P'Wwwwwwwwwz, Mmm, DD, YYYY'                        19
        FORMAT DATEFMT DATE, WIDTH 80,                                         20
               METHOD PLAINTEXT, DDNAME REPOUT1                                21
        TITLE 'Report Showing Birth Dates of all Employees'                    22
      END REPORT                                                               23
    END PROC                                                      (00370000)   24
    ;                                                                          25
    ; Following Report Called Once at EOF                                      26
    ;                                                                          27
```

```
JUL 29, 2004  18.38.41                                                PAGE      4
                                      ****************************
                                      *  PROC NAME - REPT  2     *
                                      *  INPUT STREAM PROCEDURE  *
                                      ****************************
    REPORT F.TODAY F.TODAYX F.ISDATE F.DATE,                                   28
           F.JULIAN F.JULANX F.LILIAN,                                         29
           TYPE EOF                                                            30
      FORMAT HEADINGS NAME, WIDTH 80,                                          31
             METHOD PLAINTEXT, DDNAME REPOUT2                                  32
      TITLE 'Report Showing Use of All Date Flags'                             33
    END REPORT                                                                 34
```

```
JUL 29, 2004  18.38.41                                                                                                PAGE      5
                                        ******************************************
                                        *   SUMMARY OF GENERATED SQL STATEMENTS   *
                                        ******************************************

-------------------------------------------------------------------------------------------------------------------------------
FILE>SEGMENT                    SQL STATEMENT                                                    COMMENTS
-------------------------------------------------------------------------------------------------------------------------------
M4OLD>_SQLSELO     STATEMENT  1
          SELECT * FROM DSN8810.EMP ORDER BY EMPNO                          STANDARD SELECT FOR SEQUENTIAL READ OF ROOT SEGMENT.
                                                                            WHERE CLAUSE BUILT FROM: WHERE LITERAL(S). DB2
                                                                            CALL(S): 1 PREPARE(S), 1 OPEN(S), AND 42 FETCH(ES).


JUL 29, 2004  18.38.41                                                                                                PAGE      6
** MK4W204  TYPE 0  NUMBER OF MESSAGES PRINTED IS    0.                                                                  *****
                   M4OLD    -       42 RECORDS INPUT
                   M4REPO   -       68 RECORDS OUTPUT
                   M4INPUT  -       34 RECORDS INPUT
                   M4LIST   -      138 RECORDS OUTPUT
                     15 TRACKS ASSIGNED TO M4LIB     --    0 TRACKS NOT FULL. LIBRARY DIRECTORY BLOCKING FACTOR IS  694.
** MK4W209  TYPE 0   830712 BYTES OF MAIN STORAGE UNUSED DURING DECODING PHASE                                          *****
                    832096 BYTES OF MAIN STORAGE UNUSED DURING COMPILATION PHASE
                  1060864 BYTES OF MAIN STORAGE UNUSED DURING PROCESSING PHASE
```

Index

Symbols

B

C

D

E

F

H

I

J

L

N

P

R

S

T

V