

ASG-Alliance™ **User's Guide**

Version 7.0

Publication Number: ALX0200-70

Publication Date: February 2003

The information contained herein is the confidential and proprietary information of Allen Systems Group, Inc. Unauthorized use of this information and disclosure to third parties is expressly prohibited. This technical publication may not be reproduced in whole or in part, by any means, without the express written consent of Allen Systems Group, Inc.

©2003 Allen Systems Group, Inc. All rights reserved.

All names and products contained herein are the trademarks or registered trademarks of their respective holders.

Contents

Preface	v
About this Publication	v
Related Publications	vi
ASG-Existing Systems Workbench (ASG-ESW)	vii
Invoking ESW Products	x
ESW Product Integration	xi
Example 1	xii
Example 2	xiii
Publication Conventions	xv
ASG Customer Support	xv
Intelligent Support Portal (ISP).....	xvi
Telephone Support	xvi
ASG Documentation/Product Enhancements	xviii
1 Overview: The Alliance Process	1
About Alliance	1
Step 1 - Assessing Your Task	4
Defining Your Requirements	4
Gathering Information About Related Application Components	4
Step 2 - Defining the Application	4
Allocating Your AKR.....	4
Establishing Application Boundaries	5
Step 3 - Analyzing and Refining	6
Analyzing the Application	7
Resolving Analysis Errors.....	8
Step 4 - Discovering the Nature of the Application	9
Generating Cross-reference Reports	9
Running Searches	9

Step 5 - Running Queries or Using the Query Facility	10
Performing Custom Queries	11
Searching for Specific Entities	11
Step 6 - Determining Impacted Components	11
Building List of Possible Targets	12
Selecting Items to be Analyzed	12
Determining Impacted Components and Relationships (Summary Impact Analysis) ..	13
Listing Relationships of Impacted Items	(Detailed Impact Analysis)13
Step 7 - Evaluating Components	13
Changing the Impact Generate Options	14
Refining the Base Selection List	14
Rerunning the Impact Analysis	14
Step 8 - Using Impact Analysis Results to Implement Change Request	15
Assessing Project Size	15
Determining and Assign Resources	15
Evaluating Individual Tasks	15
2 Getting Started	17
Initiating a Session	17
Verifying User Options	18
Specifying Product Parameters	18
Specifying Product Allocations	19
Processing Log and List Files	19
Allocating Script Files	22
Assigning PF Keys	23
3 Discovering the Nature of the Application	25
Generating Cross-reference Reports	26
Generating System Cross-reference Reports	26
Generating Program Cross-reference Reports	30
Running Searches	32
Understanding Synonym Relationships	32
Using Utilities	39
Searching for Application Level Synonyms	39
Searching for Program Level Synonyms	41
Searching for Homonyms	42
Saving and Printing Query/Search Results	43

4 Using the Query Facility	47
Performing Custom Queries	47
Understanding Query Language Syntax	48
Query Syntax Rules	49
The ENTITY_RELATIONSHIP BLOCK	50
The ATTRIBUTES BLOCK	52
SELECT	54
TITLE	55
WIDTH	55
SORT	56
FORMAT BLANK WHEN REPEATED	56
BREAK ON CHANGE INSERT	57
POSITION	57
DATAITEM Entity Attributes	57
DSN Entity Attributes	61
LANGUAGE Entity Attribute	62
The SELECTION BLOCK	63
Using Queries	65
Qualifying Entities in Queries	65
Aliasing Entities in Queries	66
Creating Custom Queries	67
Using the Local Synonym Query	72
Searching for Specific Entities	72
Building the List of Possible Targets (Impact List)	75
5 Determining Impacted Components	81
Selecting Items to be Analyzed	82
Locating Impacted Components and Relationships (Summary Impact Analysis)	83
Listing the Relationships of Impacted Items (Detailed Impact Analysis)	85
Understanding Detailed Analysis	87
6 Modifying the Impact Analysis	97
Changing Impact Generate Options	97
Base Impact Targets Options	99
Expanded Impact Targets Options	102
Detailed Impact Generate Options	104
Specifying New Impact Generate Options	108

Refining the Base Selection List	110
Revising an Impact List	110
Rebuilding the Base Selection List	112
Rerunning the Impact Analysis	113
Appendix A	
Valid Entities and Relationships	115
Appendix B	
Impact Analysis Entity Relationships	133
Appendix C	
Managing the AKR	181
Allocating the AKR	181
Allocating the AKR with ISPF	181
Allocating and Expanding AKRs without ISPF	184
Appendix D	
Using the Import and Export Facility	189
Importing an Application Definition	189
Exporting the Application Information	190
Exporting Application Definition Data	190
Copying Application Definition Data	192
Creating an Application Definition Template	192
Exporting Application Analysis Results	193
Using the Online Facility to Export Analysis Data	193
Using Batch JCL to Export Analysis Data	195
Appendix E	
Export Language Reference	207
Using the Output from an AKR Export	207
Entity Files	207
Relationship Files	216
Defining and Loading DB2 Databases	234
DB2 Database Description	235
Relationship Tables	242

Exporting to Transition Enablement Facility (TEF)	301
TEF Objects	302
Data Model	302
Program Control Flow.....	311
Online TEF Export Procedure.....	312
Index	315

Preface

This *ASG-Alliance User's Guide* tells you how to use ASG-Alliance (herein called Alliance). Alliance is the ASG-Existing Systems Workbench component that provides you with application-wide understanding, documentation, impact analysis, and change request assessment to help you effectively manage and maintain existing applications. Alliance allows you to quickly and accurately perform application analysis tasks. It is an accurate tool you can use to assess the resources required for application changes along with the impact of proposed system changes and failures.

Allen Systems Group, Inc. (ASG) provides professional support to resolve any questions or concerns regarding the installation or use of any ASG product. Telephone technical support is available around the world, 24 hours a day, 7 days a week.

ASG welcomes your comments, as a preferred or prospective customer, on this publication or on any ASG product.

About this Publication

This publication consists of these chapters:

- [Chapter 1, "Overview: The Alliance Process."](#) reviews the concepts of Alliance.
- [Chapter 2, "Getting Started."](#) describes how to initiate a session and validate user options.
- [Chapter 3, "Discovering the Nature of the Application."](#) gives instructions on how to use the Query and Search facilities, and describes synonym and homonym relationships.
- [Chapter 4, "Using the Query Facility."](#) provides instructions on how to perform queries and searches.
- [Chapter 5, "Determining Impacted Components."](#) describes the relational assessment process, hierarchical relationships, referential symbols and default information display options, and instructions on generating both summary and detailed impact analyses.
- [Chapter 6, "Modifying the Impact Analysis."](#) describes the impact generate options, and gives instructions on how to generate impact options and work with impact lists.

Related Publications

The documentation library for ASG-Alliance consists of these publications (where *nn* represents the product version number):

- *ASG-Application Definition and Analysis User's Guide* (ALL0200-*nn*) which provides information on defining, analyzing, and refining the application definition.
- *ASG-Alliance Installation Guide* (ALX0300-*nn*) which provides information about installing ASG-Alliance.
- *ASG-Alliance User's Guide* (ALX0200-*nn*) which provides information on the use of ASG-Alliance.
- *ASG-Center Installation Guide* (CNX0300-*nn*) which contains installation and maintenance information for ASG-Center, the common set of libraries shared by the ASG-Existing Systems Workbench suite of products.
- *ASG-ESW Enhancement Summary* (ESW1000-*nn*) highlights the new functionality for this release.

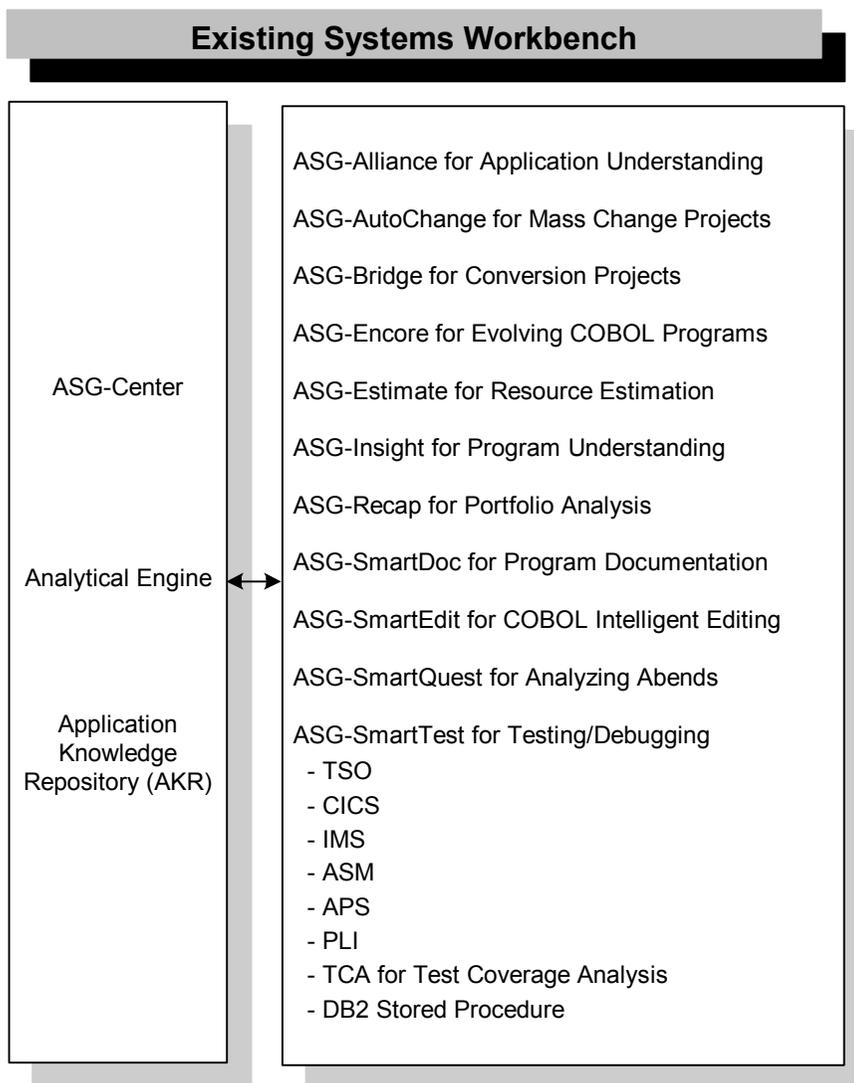
Note: _____

To obtain a specific version of a publication, contact ASG Customer Support.

ASG-Existing Systems Workbench (ASG-ESW)

ASG-ESW (herein called ESW) is an integrated suite of components designed to assist organizations in enhancing, redeveloping, or re-engineering their existing systems. ESW products use the Application Knowledge Repository (AKR) to store source program analysis information generated by the Analytical Engine. [Figure 1](#) represents the components of ESW.

Figure 1 • ASG Existing Systems Workbench



This table contains the name and description of each ESW component:

ESW Product	Herein Called	Description
ASG-Alliance	Alliance	The application understanding component that is used by IT professionals to conduct an analysis of every application in their environment. Alliance supports the analysis and assessment of the impact of change requests upon an entire application. Alliance allows the programmer/analyst to accurately perform application analysis tasks in a fraction of the time it would take to perform these tasks without an automated analysis tool. The impact analysis from Alliance provides application management with additional information for use in determining the resources required for application changes.
ASG-AutoChange	AutoChange	The COBOL code change tool that makes conversion teams more productive by enabling quick and safe changes to be made to large quantities of code. AutoChange is an interactive tool that guides the user through the process of making source code changes.
ASG-Bridge	Bridge	The bridging product that enables field expansion for program source code, without being required to simultaneously expand the fields in files or databases. Because programs are converted in smaller groups, or on a one-by-one basis, and do not require file conversion, testing during the conversion process is simpler and more thorough.
ASG-Center	Center	The common platform for all ESW products. Center provides the common Analytical Engine to analyze the source program and store this information in the AKR. This common platform provides a homogeneous environment for all ESW products to work synergistically.

ESW Product	Herein Called	Description
ASG-Encore	Encore	The program re-engineering component for COBOL programs. Encore includes analysis facilities and allows you to extract code based on the most frequently used re-engineering criteria. The code generation facilities allow you to use the results of the extract to generate a standalone program, a callable module, a complement module, and a CICS server. Prior to code generation, you can view and modify the extracted Logic Segment using the COBOL editor.
ASG-Estimate	Estimate	The resource estimation tool that enables the user to define the scope, determine the impact, and estimate the cost of code conversion for COBOL, Assembler, and PL/I programs. Estimate locates selected data items across an application and determines how they are used (moves, arithmetic operations, and compares). Time and cost factors are applied to these counts, generating cost and personnel resource estimates.
ASG-Insight	Insight	The program understanding component for COBOL programs. Insight allows programmers to expose program structure, identify data flow, find program anomalies, and trace logic paths. It also has automated procedures to assist in debugging program abends, changing a computation, and resolving incorrect program output values.
ASG-Recap	Recap	The portfolio analysis component that evaluates COBOL applications. Recap reports provide function point analysis and metrics information, program quality assessments, intra-application and inter-application comparisons and summaries, and historical reporting of function point and metrics information. The portfolio analysis information can also be viewed interactively or exported to a database, spreadsheet, or graphics package.
ASG-SmartDoc	SmartDoc	The program documentation component for COBOL programs. SmartDoc reports contain control and data flow information, an annotated source listing, structure charts, program summary reports, exception reports for program anomalies, and software metrics.

ESW Product	Herein Called	Description
ASG-SmartEdit	SmartEdit	The COBOL editing component that can be activated automatically when the ISPF/PDF Editor is invoked. SmartEdit provides comprehensive searching, inline copybook display, and syntax checking. SmartEdit allows you to include an additional preprocessor (for example, the APS generator) during syntax checking. SmartEdit supports all versions of IBM COBOL, CICS, SQL, and CA-IDMS.
ASG-SmartQuest	SmartQuest	The diagnostic tool for analyzing batch and CICS transaction abends. SmartQuest has been designed to make the maximum use of simple point-and-shoot techniques to enable fast and easy navigation through any data dump.
ASG-SmartTest	SmartTest	The testing/debugging component for COBOL, PL/I, Assembler, and APS programs in the TSO, MVS Batch, CICS (including file services), and IMS environments. SmartTest features include program analysis commands, execution control, intelligent breakpoints, test coverage, pseudo code with COBOL source update, batch connect, disassembled object code support, and full screen memory display.

Invoking ESW Products

The method you use to invoke an ESW product depends on your system setup. If you need assistance to activate a product, see your systems administrator. If your site starts a product directly, use the ISPF selection or CLIST as indicated by your systems administrator. If your site uses the ESW screen to start a product, initiate the ESW screen using the ISPF selection or CLIST as indicated by your systems administrator and then typing in the product command on the command line.

The product names can also vary depending on whether you access a product directly or through ESW. See ["ESW Product Integration" on page xi](#) for more information about using ESW.

To initialize ESW products from the main ESW screen, select the appropriate option on the action bar pull-downs or type the product shortcut on the command line.

Product Name (ESW Name)	Shortcut	ESW Pull-down Options
Alliance (Application Understanding)	AL	Understand ▶ Application
AutoChange (Conversion Set)	CC	Change ▶ Conversion Set
Bridge	BR	Change ▶ ASG-Bridge
Encore (Program Re-engineering)	EN	Re-engineer ▶ Program
Estimate	ES	Measure ▶ ASG-Estimate
Insight (Program Understanding)	IN	Understand ▶ Program
Recap (Portfolio Analysis)	RC	Measure ▶ Portfolio
SmartDoc (Program Documentation)	DC	Document ▶ Program
SmartEdit	SE	Change ▶ Program Or Change ▶ Program with Options
SmartQuest	SQV	Understand ▶ Abend/Dump
SmartTest (Testing/Debugging)	ST	Test ▶ Module/Transaction

ESW Product Integration

Because ESW is an integrated suite of products, you are able to access individual ESW products directly, or through the main ESW screen. As a result, different fields, values, action bar options, and pull-down options display on a screen or pop-up depending on how you accessed the screen or pop-up.

Certain ESW products also contain functionality that interfaces with other ESW products. Using SmartTest as an example, if Alliance is installed, SmartTest provides a dynamic link to Alliance that can be used to display program analysis information. If Insight is installed and specified during the analyze, the Insight program analysis functions are automatically available for viewing logic/data relationships and execution path. For example, the Scratchpad option is available on the Options pull-down if you have Insight installed.

[Figure 3](#) shows the Encore Primary screen that displays when you access Encore through ESW by selecting Re-engineer ► Program from the ESW action bar menu. Notice that the Primary screen name changes to ASG-ESW - Program Re-engineering when you enter Encore through ESW. Also, the Logic menu item displays if Insight is installed.

Figure 3 • ESW Encore Primary Screen

```

File View Extract Generate Search Logic List Options Help
-----
ASG-ESW - Program Re-engineering
Command ==> -----

*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****

Copyright Allen Systems Group, Inc., an unpublished work.
A proprietary product of ASG, Inc. Use restricted to authorized licensees.
Visit the ASG Support Web Site at www.asg.com

```

Example 2

[Figure 4](#) shows the File - Analyze Submit pop-up that displays when you access SmartTest directly. [Figure 5 on page xiv](#) shows the File - Analyze Submit pop-up that displays when you access SmartTest through ESW.

Figure 4 • File - Analyze Submit Screen

```

Command ==> -----
File - Analyze Submit
-----
E - Edit JCL                               S - Submit JCL

Compile and link JCL (PDS or sequential):
Data set name -----

Analyze features (Y/N):
ASG-SmartTest: Y   Extended Analysis: N

AKR data set name -----
AKR program name NEWDEMO          (if overriding PROGRAM-ID)

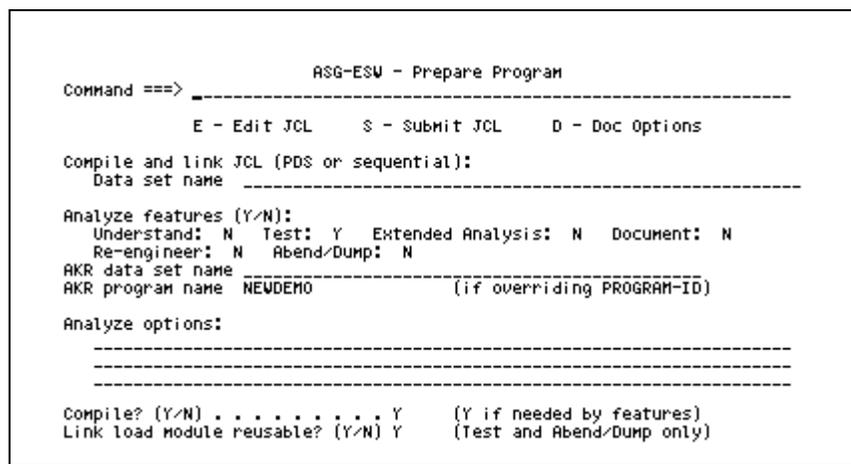
Analyze options:
-----
-----
-----

Compile? (Y/N) . . . . . Y   (Y if needed by features)
Link load module reusable? (Y/N) Y

```

The actions shown on these screens can also vary. For example, the D - Doc Options action is only available on the File Prepare Program screen (or File - Analyze Submit screen) if SmartDoc is installed on your system. In [Figure 4 on page xiii](#), the Doc Options action is not displayed.

Figure 5 • ASG-ESW - Prepare Program Screen (accessed through ESW)



Notice that the Analyze features field in [Figure 5](#) lists additional ESW products than shown on [Figure 4 on page xiii](#). This field is automatically customized to contain the ESW products you have installed on your system. These are the names of the analyze types:

Analyze Type	Analyze Type (ESW)
ASG-Encore	Re-engineer
ASG-Insight	Understand
ASG-SmartDoc	Document
ASG-SmartQuest	Abend/Dump
ASG-SmartTest	Test
Extended Analysis (ASG-SmartTest with Insight installed)	Extended Analysis

Publication Conventions

ASG uses these conventions in technical publications:

Convention	Represents
ALL CAPITALS	Directory, path, file, dataset, member, database, program, command, and parameter names.
Initial Capitals on Each Word	Window, field, field group, check box, button, panel (or screen), option names, and names of keys. A plus sign (+) is inserted for key combinations (e.g., Alt+Tab).
<i>lowercase italic monospace</i>	Information that you provide according to your particular situation. For example, you would replace <i>filename</i> with the actual name of the file.
Monospace	Characters you must type exactly as they are shown. Code, JCL, file listings, or command/statement syntax. Also used for denoting brief examples in a paragraph.
Vertical Separator Bar () with underline	Options available with the default value underlined (e.g., Y <u>N</u>).
<u>Underline</u>	Denotes a cursor-selectable field or line.

ASG Customer Support

ASG provides support throughout the world to resolve questions or problems regarding installation, operation, or use of our products. We provide all levels of support during normal business hours and emergency support during non-business hours.

ASG Third-party Support. ASG provides software products that run in a number of third-party vendor environments. Support for all non-ASG products is the responsibility of the respective vendor. In the event a vendor discontinues support for a hardware and/or software product, ASG cannot be held responsible for problems arising from the use of that unsupported version.

Intelligent Support Portal (ISP)

Online product support is available at: <http://www.asg.com/support/support.asp> via the ASG Intelligent Support Portal (ISP). Your logon information for ISP online support is:

Customer ID = *NNNNNNNNNN*

Password = *XXXXXXXXXX*

where:

NNNNNNNNNN is your customer ID supplied by ASG Product Distribution.

XXXXXXXXXX is your unique password supplied by ASG Product Distribution.

The *ASG-Intelligent Support Portal User's Guide* provides instructions on how to use the ISP and is located on the ASG Support web page.

Telephone Support

To expedite response time, please have this information ready:

- Product name, version number, and release number
- List of any fixes currently applied
- Any alphanumeric error codes or messages written precisely as displayed
- A description of the specific steps that immediately preceded the problem
- Verify whether you received an ASG Service Pack or cumulative service tape for this product. It may include information to help you resolve questions regarding installation of this ASG product. The Service Pack instructions are in a text file on the distribution media included with the Service Pack. You can access the latest software corrections and Service Packs via the ISP.
- The severity code (ASG Customer Support uses an escalated severity system to prioritize service to our clients. The severity codes and their meanings are listed below.)

Severity Codes and Expected Support Response Times

Severity	Meaning	Expected Support Response Time
1	Production down, critical situation	Within 30 minutes
2	Major component of product disabled	Within 2 hours
3	Problem with the product, but customer has work-around solution	Within 4 hours
4	"How-to" questions and enhancement requests	Within 4 hours

The Americas

	Phone	Fax	E-mail
United States and Canada	800.354.3578	1.703.464.4901	support@asg.com

Europe, Middle East, and Africa (EMEA)

During normal business hours, we recommend that you call the Central Support number first (except in South Africa).

	Phone	Fax	E-mail
Central Support	00.800.3544.3578	44.1727.812018	support.emea@asg.com
English	44.1727.736305	44.1727.812018	support.uk@asg.com
French	33.141.028590	33.141.028589	support.fr@asg.com
German	49.89.45716.200	49.89.45716.400	support.de@asg.com
Italian	39.0290450025		support.it@asg.com
Dutch	31.30.241.6133		support.nl@asg.com
Spanish	34.913.523.800	34.917.156.961	support.es@asg.com
South Africa	800.201.423		support.sa@asg.com

Asia Pacific (APAC)

	Phone	Fax	E-mail
Central Support	61.3.9645.8500	61.3.9645.8077	support.au@asg.com
Australia	800.637.947	61.3.9645.8077	support.au@asg.com
Hong Kong	800.96.2800		support.hk@asg.com
Japan	81.3.5326.3684	81.3.5326.3001	support.au@asg.com
Singapore	65.224.3080	65.224.8516	support.sg@asg.com

All Other Countries (Also for any non-working numbers)

	Phone	Fax	E-mail
All other countries	1.239.435.2201		support@asg.com

If you receive a voice mail message, follow the instructions to report a production-down or critical problem. Leave a detailed message including your name and phone number. An ASG Customer Support representative will be paged and will return your call as soon as possible. Please have available the information described previously when the ASG Customer Support representative contacts you.

ASG Documentation/Product Enhancements

Submit all product and documentation suggestions to ASG's product management team at <http://www.asg.com/asp/emailproductsuggestions.asp>.

If you do not have access to the web, FAX your suggestions to product management at (239) 263-3692. Please include your name, company, work phone, e-mail ID, and the name of the ASG product you are using. For documentation suggestions include the publication number located on the publication's front cover.

1

Overview: The Alliance Process

This chapter presents an overview of Alliance in these sections:

Section	Page
About Alliance	1
Step 1 - Assessing Your Task	4
Step 2 - Defining the Application	4
Step 3 - Analyzing and Refining	6
Step 4 - Discovering the Nature of the Application	9
Step 5 - Running Queries or Using the Query Facility	10
Step 6 - Determining Impacted Components	11
Step 7 - Evaluating Components	13
Step 8 - Using Impact Analysis Results to Implement Change Request	15

About Alliance

Alliance assesses the impact of change upon your application and helps you accurately determine the required resources. With Alliance, you have an automated, interactive tool to analyze and understand the nature and relationships of application components.

These are some of the benefits of using Alliance for your maintenance requirements:

Increased expediency. Using Alliance gives you increased expediency in modifying your application. After you define and analyze the application, Alliance enables you to spend less time manually determining the scope of a maintenance task.

Saved resources. Alliance saves resources in helping you determine how large the requirement is. You no longer need to deploy an entire team to assess the task at hand.

Reduced errors. The potential of human error is reduced. With impact analysis results generated by Alliance, you work from reliable data. As a result, the possibility of a system failure when the change is rolled into a production environment is minimized.

Increased application information. The application information analysis results Alliance provides gives you valuable maintenance tools for the application life cycle. Reuse this base of information to expedite future change requests.

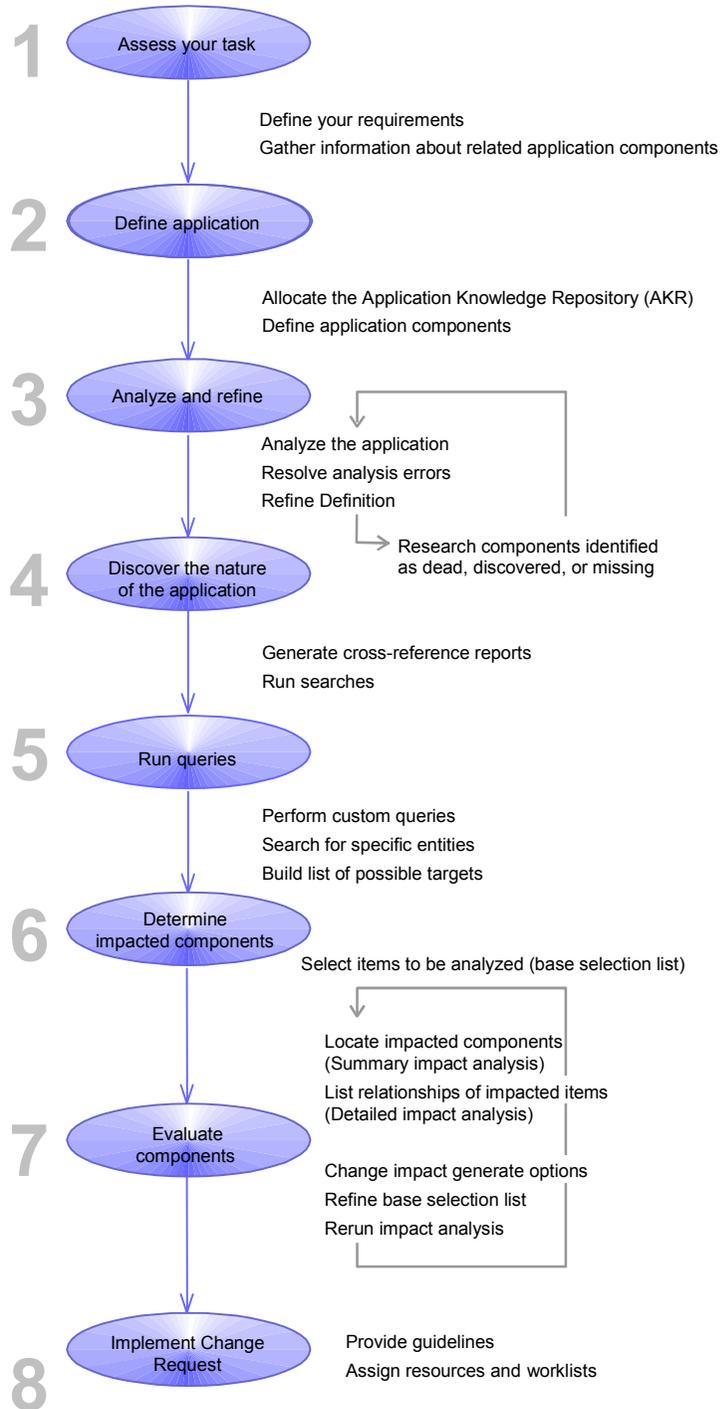
Enhanced understanding of application components. The analysis information generated by Alliance, based on your application definition, provides a complete understanding of the components that comprise your applications and the relationships between those components.

The Alliance process is relatively simple. Nevertheless, ASG recommends you use the default options until you can anticipate the result of changing those values. Altering default settings can make these procedures unnecessarily time- and resource-intensive.

As illustrated in [Figure 6 on page 3](#), you can outline the process in eight steps. These guidelines facilitate your understanding of Alliance.

This publication gives detailed instructions on performing each step in subsequent chapters, referenced under each heading. The *ASG-Application Definition and Analysis User's Guide* describes the first three steps in the Alliance process in full detail.

Figure 6 • Alliance Process Outline



Step 1 - Assessing Your Task

At this stage, define the assigned task and determine the affected applications and components.

For step-by-step instructions, see the *ASG-Application Definition and Analysis User's Guide*.

Defining Your Requirements

Identify the change and impact upon the application. For example, you can determine what components of the Manufacturing application are affected by euro conversion processing, and use that information to develop a conversion project plan.

Gathering Information About Related Application Components

Begin gathering information to define the application to Alliance. This is an iterative process. Record information about the application components and organize this information using a series of worksheets. Copy and fill out the worksheets as needed to begin the application definition process. See the *ASG-Application Definition and Analysis User's Guide* for more information.

Step 2 - Defining the Application

In Step 2, you create the initial application definition. First, allocate repository space for data storage. This is called the Application Knowledge Repository (AKR). Then, define the information gathered about the application to Alliance. This collection of information establishes the application boundaries. Alliance identifies the components you have targeted for assessment and generates the impact results within these boundaries.

Allocating Your AKR

The AKR is the repository for all of the information used by the ESW family of components. The ESW provides you with both online and batch utilities for managing the AKR. You can allocate a new AKR, or use an existing one. Each AKR can contain multiple applications, or you can allocate separate AKRs for each application.

An application is a group of programs and associated components that accomplishes a particular function. You can group all of the applications used by a business unit within an AKR. For example, typical applications within an accounting AKR could include these items:

- Accounts Payable
- Accounts Receivable
- General Ledger
- Fixed Assets
- Inventory
- Purchasing

Also, you could allocate an AKR for each major business unit in your company (for example, manufacturing, accounting, and sales) for management convenience.

After the AKR is allocated, you can start defining your application to Alliance.

For step-by-step instructions, see ["Allocating the AKR" on page 181](#).

Establishing Application Boundaries

To establish the application boundaries, create an application definition, which is an inventory of related programs and associated component definitions. Using this inventory, Alliance executes an analysis to understand the application components and relationships. Application definitions are reusable. You can open an existing one as well as create a new one. You can also import an application definition from text files.

When you create the application definition within the AKR, specify these libraries and members Alliance includes in its application analysis:

- Program source
- Load module
- JCL
- CICS
- IMS

You can also specify attributes for each library and member in the definition. For example, you could add this information:

- Analysis parameters
- Copy libraries
- Procedure libraries
- Macro libraries
- CA-IDMS information

If a library contains multiple categories of members (for example, PROGRAM and INCLUDE members), you must define the library separately for each category.

Step 3 - Analyzing and Refining

At this stage, you analyze and refine the application definition. Building an application definition is an iterative process.

These are the steps for building an application:

- 1** Run the analysis.
- 2** Review the results.
- 3** Make modifications.
- 4** Run the analysis again until you reach an accurate application definition.

When the Alliance Application Analytical Engine (AAE) analyzes the items in the application definition, it determines the application components and their relationships. The AAE also generates information about the definition. The first part of this information contains statistics and diagnostics about the components that were analyzed. The second organizes the analyzed components into three types: dead, discovered, and missing. You can use both sources to refine the application definition throughout the iterative process of defining an application.

With each analyze job, Alliance produces a summary report of analysis statistics and diagnostic messages called the VIAARPT. This report helps you to understand how the job ran, specifies the return code from the individual analysis of each member, and identifies analysis errors (for example, you might forget to define a copy library or miss a syntactic error). This file is temporary and is not stored in the AKR.

Alliance identifies discovered, missing, or dead application entities inside your application. With each analyze job, you can assess these components to determine their validity. This verifies the accuracy of the collected data and allows you to discover unaccounted aspects in the application.

Analyzing the Application

The analysis of your application is generated from the inventory of application components specified in your application definition. Alliance performs these types of analysis:

- Full analysis — analyzes every member in the application at one time, and can require a large amount of resources.
- Incremental analysis — analyzes only the members you select. This analysis allows you to perform your application analysis in phases to alleviate heavy resource demands.
- Auto analysis — similar to an incremental analysis except the analyzer selects the members that need to be analyzed. This analysis selects the members that you did not analyze that contain bad return codes, or that you modified since the last analysis.

You must completely analyze an application to obtain accurate results from an impact analysis. An analyze resource estimate table is located in the *ASG-Application Definition and Analysis User's Guide*.

The last step in the application process is Semantic linking, which determines how the application components relate to each other. Semantic linking can be a resource-intensive phase of the analyze job, and must be done before you can use the AKR for impact assessment.

If you want to analyze a large application and lack the resources to run a full analysis, disable the semantic linking option and perform an incremental analysis on sections of your application while you refine your application definition.

These are the steps for performing an incremental analysis on sections of your application:

- 1 Submit the selected components for analysis.
- 2 Review the results.
- 3 Modify the application definition.
- 4 Resubmit as necessary.

After creating an accurate application definition and analyzing all the components in the application, enable the semantic linking option and run an analysis.

Resolving Analysis Errors

Occasionally, lack of CPU time, low space allocation, and other operating system problems prevent the application analysis from completing. However, you can restart the analysis after resolving the problem, unless you change Alliance's default setting for recovery.

With each analysis, Alliance produces the VIALOG, a log file of errors, and the VIAARPT, a summary report of run-time statistics and diagnostic messages. Use these tools to find out why an error occurred, where it occurred, and to help you determine a fix. Also, you can look up explanations in the online help or see the detailed listing located in the *ASG-Application Definition and Analysis User's Guide*.

Note: _____

Semantic linking is a common cause of abends stemming from excessive CPU usage.

Refine Definition

After you resolve all analysis errors, begin refining the application definition. First, research the components the analysis detected as discovered, missing, or dead. Then, rerun the application analysis, resolve errors, and research dead, discovered, or missing components until you understand and accept these components, or there are none. A clean output ensures the accuracy of a Alliance impact assessment.

Research Components Identified as Dead, Discovered, or Missing

The analytical engine produces these three component lists to assist you in refining the application definition:

- **Dead components**—These components are in the application definition, but appear unused by the other components. You might need to search for and add a component that references the dead component. Review the dead component list to determine if the components are used in any other application. If you are sure the component is completely unused, you may want to remove it from the system. However, delete unused components cautiously. Components that appear to be dead may not actually be dead.
- **Discovered components** —These are components referenced in the application, but not included in the application definition. For example, a load module execution in your JCL that is missing from the application definition. Review the discovered components list and determine if you need to add the discovered components and attributes.
- **Missing components** —These components were present at the time of the original definition, but can no longer be found. Perhaps you renamed or deleted a library. Review the missing components list to determine if they were renamed. If a component is renamed or deleted, remove the reference to the old name. Add a renamed component to the definition with its new name.

To refine the application definition, review each identified item to determine if any action should be taken. After you refine the definition, resubmit the application for incremental analysis and semantic linking. Then the application analysis is complete and you can build the list of data items for an impact analysis.

Step 4 - Discovering the Nature of the Application

In Step 4, examine the application analysis results to familiarize yourself with aspects of the application possibly impacted by the change. Now that the process of defining and analyzing the application components has given you an understanding of the whole application, evaluate the results to identify the components within the requirement parameters.

Alliance includes a Query facility that generates cross-reference reports and Search facilities that sort out equivalent components. Queries and searches are optional, but ASG recommends you use them. They narrow the selection criteria and help you understand the entities and relationships you want the Impact facility to analyze.

You can save all queries, searches, and impact analyses results for future use in one or more sequential files, called worklists. After a worklist is open, information is automatically saved when the display changes or the current activity ends.

Generating Cross-reference Reports

System cross-reference reports display application-level data, such as the dataset name, the job name, and step name where the dataset is used. Program cross-reference reports display program-specific (local) information, such as called and calling program and copybooks in programs inside the application.

Alliance provides a set of template queries that address common questions. Run these queries to display system-level and program-level cross-reference results on the application in standard views. Later, when you are closer to determining the starting points for impact assessment, use the Alliance query language to customize these template queries, or to create new ones to show specific relationships.

For step-by-step instructions, see ["Generating Cross-reference Reports" on page 26](#).

Running Searches

The search feature is an important factor in refining the scope of your analysis. For example, use the search function to detect synonyms (components that contain the same data) to reduce the number of entities selected for analysis without diminishing the accuracy of the results. Use homonyms (components with the same name) to help identify entities that should be analyzed.

The Alliance search features enable you to conduct application-wide searches from the Query or the Impact facility. Use the search function to locate specific entities and data items matching the criteria strings you define.

These are the items Alliance evaluates when identifying synonyms:

- Assignments
- Parameters
- Renames and redefinitions
- Parents and children
- Indirect and implied dependencies
- Utilities

When identifying homonyms, Alliance detects data items with the same name.

Synonyms and homonyms exist at both application and program (local) levels.

For step-by-step instructions, see ["Running Searches" on page 32](#).

Step 5 - Running Queries or Using the Query Facility

At this stage, identify the components possibly impacted by the change and submit the list to the Alliance Impact facility. This list is called the impact entity list (or simply the impact list). The more narrow the scope of the analysis with a precise impact list, the more accurate the information obtained for impact assessment.

These are the steps for limiting the impact analysis to relevant entities:

- 1** Build an impact list consisting of pairs of entity kinds and unique names that match your defined selection criteria. The objective is to build an impact list based on the understanding gained in ["Step 4 - Discovering the Nature of the Application" on page 9](#).
- 2** Customize the Alliance standard queries to display application information about specific entities you want included in the analysis.
- 3** Use the search feature to refine the entities included in the analysis.

After determining the combination of entity kinds and unique names to include in the impact list, Alliance matches the impact list against the open application to derive a list of components for impact analysis. The selections you make from this list, also known as the selection list, are the starting point for the impact analysis evaluation of all relationships. Selecting items for the base selection list is the first action of ["Step 6 - Determining Impacted Components" on page 11](#).

Performing Custom Queries

The next step is to uncover more details by modifying existing queries or creating new ones based on the information gathered through standard queries and searches. For example, you might want to look up specific entities and their relationships.

Construct custom queries using the syntax supported by Alliance. An Alliance query consists of these two definition blocks:

- The ENTITY_RELATIONSHIP BLOCK — names the application entities (such as programs, FDs, PROCs, and JCL) included in the query and the relationships used in the search for application information.
- The ATTRIBUTES BLOCK — enables you to select data to be included in the query results and to format the display differently than the default format.

For step-by-step instructions, see ["Performing Custom Queries" on page 47](#).

Searching for Specific Entities

After extracting system and program cross-reference data from the application analysis results, use the advanced search options to locate specific entities possibly impacted by the change. You can conduct application-wide searches for specific entities, entity kinds, and literal search strings from the Impact or Query facility.

These searches enable you to track individual entities, entity groups, and the occurrence and location of literal strings. These search results contribute to the field of entities to determine the starting points for the impact list.

For step-by-step instructions, see ["Searching for Specific Entities" on page 72](#).

Step 6 - Determining Impacted Components

At this stage, generate the first round of impact analysis results based upon the list of items selected for impact assessment. The Alliance impact analysis automates the research process necessary to investigate the impact of change on an application. It evaluates the dataflow and logical and physical relationships to determine the impact of changes on an application.

Alliance produces both summary and detailed impact analysis information. The summary impact analysis identifies components impacted by the change. The detailed impact analysis explains the summary analysis results and shows the physical, logical, and dataflow relationships transversed. The summary and detailed analyses output displays on the Impact Facility screen. You can save the results of each impact analysis to a worklist.

Like the application definition, impact analysis is an iterative process. For example, after reviewing the components in the summary analysis, you might use the detailed impact analysis to examine the impact relationship of each component. Evaluating the impact results may lead you to change some settings for impact generate or display options, and modify the base selection list. Then, you would run the impact analysis again.

Building List of Possible Targets

You can build an impact list with unique entities that are affected by the change after you determine the components of interest for the change request. Unless you want to include all the entity kinds in your application in the impact list, you must define selection criteria to limit the number of entities the impact analysis targets. To use the selection criteria, specify the full name of each impacted entity kind, or use text patterns to identify entity groups.

You can generate a new impact list each time you run the impact analysis on an application, or you can save the impact list to a worklist for future use in this or other impact analyses. After you specify an impact list, you can create the base selection list the Impact facility uses to evaluate all relationships.

For step-by-step instructions, see ["Building the List of Possible Targets \(Impact List\)" on page 75](#).

Selecting Items to be Analyzed

After you specify the type of impact analysis (detailed or summary) the Impact facility matches the entities in the impact list to those in the open application to create a list of targets to select from. The components you select comprise the base selection list.

After you select entities for the impact analysis, specify the scope of the analysis by defining impact generate options. These options control how the impact analysis evaluates data flow and hierarchical relationships. If this is an initial impact analysis, ASG recommends you use the defaults for these options. After you set the impact generate options, start the impact assessment.

For step-by-step instructions, see ["Selecting Items to be Analyzed" on page 82](#).

Determining Impacted Components and Relationships (Summary Impact Analysis)

The summary impact analysis lists all of the components affected by the proposed change. It also displays the summary information for each impacted entity kind and the number of occurrences for each. From this information, you can determine the task size and the resources needed for implementation. After you review the list of impacted entities and the number of changes involved, expand the summary information through detailed analysis to see how these impacted entities were found, and how they are related. For step-by-step instructions, see "[Locating Impacted Components and Relationships \(Summary Impact Analysis\)](#)" on page 83.

Listing Relationships of Impacted Items (Detailed Impact Analysis)

The detailed impact analysis displays the paths taken to generate the summary analysis results. The display is hierarchical, with each level representing an impact relationship path to a new component. Follow the path of a component by expanding and collapsing the display. You can also display the Detailed Information pop-up that contains data about the component at the cursor location.

Alliance builds impact analysis by layering related components. The detailed analysis shows what each layer is and why the layer was applied to the member of the base selection list.

Various options control the information shown for each entity. The defaults are name, entity kind, and containing component. You can change the detailed impact options to display additional information to the right of the original entity in the Impact Facility screen.

After you generate the impact analysis results, you can evaluate the impacted components.

For step-by-step instructions, see "[Listing the Relationships of Impacted Items \(Detailed Impact Analysis\)](#)" on page 85.

Step 7 - Evaluating Components

At this stage, evaluate any components in the impact analysis results that warrant further investigation. This evaluation leads to these two choices:

- Determine if the hierarchical and data flow relationships traversed during impact analysis need to be adjusted. To do this, change the Impact Generate options.
- Alter the base selection list and/or the items in your impact list.

The second choice is recommended because you should change the Impact Generate options cautiously. Inappropriate use can cause inaccurate results and tie-up system resources.

Changing the Impact Generate Options

The impact generate options affect the way the impact analysis traverses the data flow and the hierarchical component relationships. For example, you might decide to include entities that the base selection member assigns value to or from, or to include data movement and comparison within programs for statements other than direct moves (i.e., alternate assignments).

For step-by-step instructions, see ["Changing Impact Generate Options" on page 97](#).

Refining the Base Selection List

In some cases, you may need to change the components of the base selection list to include data that was not included in a previous analysis. For example, you might find an item that is not needed in the impact assessment and decide to remove it from the base selection list. Or, you might decide to add items to the impact list you did not initially plan to analyze.

For step-by-step instructions, see ["Refining the Base Selection List" on page 110](#).

Rerunning the Impact Analysis

The last step of this iterative process is to rerun the impact analysis until you gather enough data to proceed with the change. Rerun the Alliance impact analysis after you perform these tasks:

- Finish reviewing the results of the last impact analysis.
- Finish investigating the paths traversed by components related to the change.
- Verify that the impact generate options or the base selection list require no changes.

The results of the final iteration of this process shows all application components impacted by the change.

For step-by-step instructions, see ["Rerunning the Impact Analysis" on page 113](#).

Step 8 - Using Impact Analysis Results to Implement Change Request

At this stage, begin organizing and implementing the change request. Prior to performing any other action, save the impact analysis results from the final iteration of the process to a worklist.

Assessing Project Size

Use the entity counts in the impact analysis results to provide an immediate indication of the size of the task. The capability of the available resources and the complexity of the change request must also be given consideration, but are outside the scope of this discussion.

Determining and Assign Resources

Assign resources to implement the change. You can break the worklist up and distribute it to the assigned resources so that each has an exact list of the components that they are responsible for changing.

Evaluating Individual Tasks

Prior to implementing the change, execute a detailed impact analysis against the individual components to show how each component relates to the application components. The Query facility reveals specific relationships of interest. Additionally, you can use the Search facilities to further clarify the role of the component being changed.

2

Getting Started

This chapter describes how to initiate Alliance and set user options, and consists of these sections:

Section	Page
Initiating a Session	17
Verifying User Options	18

Initiating a Session

Note: _____

The method you use to start ESW products depends upon your system configuration.

To start Alliance directly, follow this step:

- ▶ Use the ISPF selection or CLIST, as indicated by your systems administrator. After you activate the session, the Alliance Primary screen displays.

To use the ESW screen to start individual products

- 1 Initiate the ESW screen using the ISPF selection or CLIST, as indicated by your systems administrator.
- 2 Select Understand ▶ Application from the ESW Primary screen to start the product.

The ESW - Application Understanding or Alliance Primary screen displays. During the installation, ESW product options are customized for your site. If you are using the product for the first time, you might want to perform the procedures in the next sections to ensure that installation defaults meet your needs.

Verifying User Options

During installation, options that define the operating environment are set to default values. These are the options that you can modify:

- Product parameters that control Alarm settings and the use of confirmation pop-ups
- Product allocation of DASD information for the log, list, and work files
- Log and list file processing options
- Script file allocations
- PF key assignments

Specifying Product Parameters

To verify or change product parameters

- 1 Select Options ► Product parameters from the Alliance Primary screen and press Enter. The Options - Product Parameters pop-up displays (see [Figure 7](#)).

Figure 7 • Options - Product Parameters Pop-up

```
Options - Product Parameters
Command ===> _____
Specify parameter information.
Alarm . . . . . YES (Yes/No)
Confirm submit . . NO (Yes/No Confirmation displayed before Submit)
Confirm delete . . YES (Yes/No Confirmation displayed before Delete)
AMF Stats Required YES (Yes/No IF ISPF Stats required for AMF)
Confirm save . . . YES (Yes/No Save results to Work List)
```

- 2 Review the information and make any necessary changes.
- 3 Specify whether the Application Maintenance Facility (AMF) requires the ISPF statistics (time datestamp) to determine if a source member was modified.

Choose NO for the AMF Stats Required Field to allow the AMF to use a potentially expensive calculation to determine if members without ISPF statistics were modified.

See the *ASG-Application Definition and Analysis User's Guide* for more details.

- 4 Press PF3 to save changes and exit.

Specifying Product Allocations

Use product allocations options to specify the product allocations of DASD information for the Log, List, and Work files.

Alliance allocates the Log file if it encounters an internal error, such as an abend. The Log file contains ESW error messages. The List file contains printed output and is allocated when you issue a request to print.

The analysis work file is a temporary AKR file that contains intermediate results while an analyze job is running. This file should be as large as the space requirements for the application itself. Use the other work file parameters for other temporary files allocated during other processing.

To verify product allocation settings

- 1 Select Options ► Product allocations from the Alliance Primary screen and press Enter. The Options - Production Allocations pop-up displays (see [Figure 8](#)).

Figure 8 • Options - Product Allocations Pop-up

```

File Edit View Facility Options Help
-----
Options - Product Allocations
Command ==> -----

Specify allocation information.
      Log File
Generic unit . . . SRTDA
Volume serial . . . SRT801

      List File
SYSDA (Group name or unit address)
SRT801 (Blank for default volume)

      (Analysis) Work File
Generic unit . . . SYSDA
Volume serial . . . -----
Space units . . . CYLS
Primary space . . . 050
Secondary space . . 010
Alternate node . . -----

      Work File (Other)
SYSDA (Group name or unit address)
----- (Blank for default volume)
CYLS (BLKS, TRKS or CYLS)
5 (Space units)
5 (Space units)

```

- 2 Review the information and make any necessary changes.
- 3 Press PF3 to save changes and exit.

Processing Log and List Files

Use Log and List file processing options to specify the settings for processing the Log and List files. The Options - Log/List Definition pop-up also defines printing or deletion options.

To verify or change the setting for processing

- 1 Select Options ► Log/List from the Alliance Primary screen and press Enter. The Options - Log/List Definition pop-up displays (see [Figure 9](#)).

Figure 9 • Options - Log/List Definition Pop-up

```

Options - Log/List Definition
-----
Command ===> _____
Specify Log/List options. Then press PF key for action.

Options          Log          List
-----          -
Process option   . . . . . K          PD
Primary tracks   . . . . . 1          1
Secondary tracks . . . . . 2          5
Lines per page   . . . . . 56         56
Sysout class     . . . . . *          *

Process options: PK (print/keep), PD (print/delete), K, or D.

Job statement information:
//VIA123 JOB (DEVJMS,283200),
//          MSGCLASS=A
//          INSERT /*ROUTE PRINT NODE.USER' HERE IF NEEDED.
//          /*

PF4=Customized Names PF5=Process log file PF6=Process list file
    
```

- 2 Verify that the allocation and disposition choices conform to your site standards and make the necessary changes.
- 3 Enter the appropriate information in the Job statement information field, if required for printing the Log or List file.
- 4 Press PF3 to save changes and exit.

To customize the Log or List dataset name

- 1 Specify the K or PK process option on the Options - Log/List Definition pop-up.
- 2 Customize the dataset where the log or list file is allocated. By default, Alliance allocates the Log, and List files with this syntax:

`USERID.ALNnnnnn.VIAxxxxxx`

where:

`nnnnn` is a sequential number from 00001 to 99999.

`xxxxxx` is LOG for Log and LIST for List files.

If you specified a TSO prefix, the prefix appends to the beginning of the file name allocated for the Log and List files.

To define a dataset name

- 1 From the Options - Log/List Definition pop-up, press PF4. The Options - Log/List Name Customization pop-up displays (see [Figure 10](#)).

Figure 10 • Options - Log/List Name Customization Pop-up

```

Options - Log/List Name Customization
Command ==> -----
You can define a customized name when you choose option PK (print/keep) or
K (keep) by specifying U(ser). Specifying Y(es) on Prompt later lets you
define a custom name as you process each log/list file. Otherwise,
specify data set name and file mode. Then press Enter.

Options ----- Log      List
File Naming . . . . . $      $      U(ser) or S(ystem)
Prompt later for DSN . . _    -      Y(es) or N(o)

The following are needed if U(ser) and N(o) are specified above

Log Data set name ----- (Seq)
File Mode . . _          O(verwrite) or A(ppend)
List Data set name ----- (Seq)
File Mode . . _          O(verwrite) or A(ppend)

```

- 2 Type U in the File Naming field for Log and/or List to indicate a user-defined dataset name and press Enter.

If you specify N in the Prompt later for DSN field, you must enter a dataset name in the corresponding Dataset name field, and specify Overwrite or Append in the File Mode field.

If you specify Y in the Prompt later for DSN field, Alliance prompts you for the dataset name during file processing. For example, if you specify Yes in the Prompt later for DSN field for the Log file, the Log Name Customization pop-up displays (see [Figure 11](#)).

Figure 11 • Log Name Customization Pop-up

```

Log Name Customization
Command ==> -----
The current log file's data set name is shown below. To have a custom
name, specify a new sequential data set name. If it already exists, it
has to have LRECL=137 and RECFM=U; and specify the file mode. Then press
Enter.

Current Data set name : ''
Custom Data set name -----
File Mode . . _          O(verwrite) or A(ppend)

```

Allocating Script Files

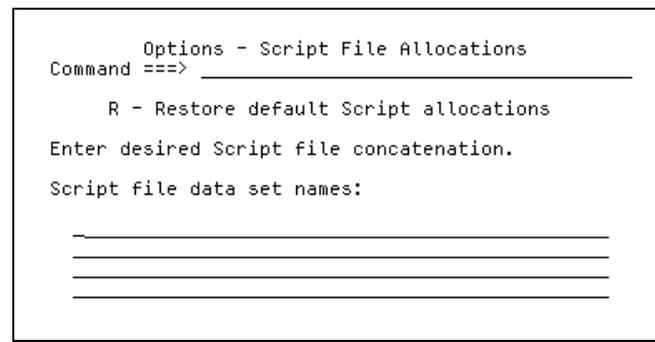
A script file automates a sequence of commands and screen interactions. Typically, you use a script file for Batch processes, such as definition changes or Batch queries.

During the installation, you specify default script files for the site. Script file allocation options specify the default dataset names or the concatenation sequence for your session.

To display or change the script files

- 1 Select Options ► Script file allocations from the Alliance Primary screen and press Enter. The Options - Script File Allocations pop-up displays (see [Figure 12](#)).

Figure 12 • Options - Script File Allocations Pop-up



- 2 Review the current default script file dataset names.

Note: _____

When initializing for defaults, the lines are filled from the bottom to allow concatenation.

- 3 Enter the dataset names for the new script files in the Script file dataset names field, if needed. The files are concatenated in the order entered.
- 4 Type R on the command line and press Enter to restore the default.
- 5 Press PF3 to save any changes and exit.

Assigning PF Keys

PF key assignment options specify PF key functions. [Figure 13](#) shows the Options - PF Key Definition pop-up with default settings. These default PF key assignments apply only if no PF key assignments are listed in the screen or pop-up. Fixed PF key assignments listed at the bottom of screens and pop-ups override these defaults.

Figure 13 • Options - PF Key Definition Pop-up with Default Settings

```

Options - PF Key (01-12) Definition
Command ==> _____
Press Enter to process changes and/or to display alternate keys.
Press PF3/15 (END) to exit.

      Number of PF keys: 12      Terminal type: 3278

PF01 HELP
PF02 SPLIT
PF03 END
PF04 RETURN
PF05 RFIND
PF06 RCHANGE
PF07 UP
PF08 DOWN
PF09 SWAP
PF10 LEFT
PF11 RIGHT
PF12 CANCEL

```

To verify the current PF key assignments

- 1 Select Options ► PF keys from the Alliance Primary screen and press Enter. The Options - PF Key Definition pop-up displays showing current PF key settings.
- 2 Type over the current setting to change the function of a PF key.
- 3 Press PF3 to save changes and exit.

3

Discovering the Nature of the Application

This chapter describes how to use the Query and Search facilities, defines synonym and homonym relationships, and contains these sections:

Section	Page
Generating Cross-reference Reports	26
Running Searches	32
Saving and Printing Query/Search Results	43

As described in the overview, Alliance includes a Query facility that allows you to generate cross-reference reports on the results of the application analysis. After you refine the definition of the application, identify components possibly impacted by the change. You can then use this information to specify selection criteria for the entities you want the Impact facility to analyze.

Save the results of each query to one or more worklists so you can access the data when building the list of possible candidates for the impact analysis. After you open a worklist, Alliance automatically saves the information when the display changes or the current activity ends.

Alliance also provides Search facilities that identify components that contain the same data, have the same name, or make use of literals. You can use this information to refine the candidates for impact assessment. Use both the Impact and the Query facility to perform searches.

Generating Cross-reference Reports

Alliance provides a set of cross-reference reports, or template queries, that answer common questions about the components in the applications. System cross-reference reports provide information about system-level components in the application (e.g., databases, jobs, and library members). Program cross-reference reports provide information about programs and the components related to and contained within programs [e.g., data items, records, and copymembers (includes)].

After you review the application components through these standard views and are closer to determining the starting points for impact assessment, use Alliance's query language to customize these template queries or to create new ones that display specific relationships.

For step-by-step instructions, see ["Using the Query Facility" on page 47](#).

Generating System Cross-reference Reports

To generate a report with system cross-reference information

- 1 Select View ► System Cross Reference Information from the Alliance Primary screen and press Enter. The System Cross Reference Information pop-up displays.

These are the available system cross-reference reports:

Report Name	Displays
CICS File cross-referenced by Program	<ul style="list-style-type: none">• CICS file DD• Program
CICS Map cross-referenced by Program	<ul style="list-style-type: none">• CICS maps• Mapsets• Program names
DSN cross-referenced by Job	<ul style="list-style-type: none">• DSN• DD• STEP name• JOB name• PROC name• PROC STEP• PROC JOB name

Report Name	Displays
DSN cross-referenced by Copybook	<ul style="list-style-type: none">• DSN• Record• Size• Copybook
IMS DBD cross-referenced by Segment	<ul style="list-style-type: none">• IMS DBD• Segment• Segment fields
IMS PSB cross-referenced by Program	<ul style="list-style-type: none">• IMS PSB• Program name
Job cross-referenced by JCL library	<ul style="list-style-type: none">• JOB name• JCL library name• Member name containing the JCL
Job cross-referenced by JOBLIB	<ul style="list-style-type: none">• JOB library DSN• JOB name
Job cross-referenced by Load Module	<ul style="list-style-type: none">• JOB name• JOB STEP name• Load Module name <p>If a PROC is executed:</p> <ul style="list-style-type: none">• PROC name• PROC STEP• PROC STEP load module• Member containing the job

Report Name	Displays
Job cross-referenced by STEPLIB	<ul style="list-style-type: none"> • JOB name • JOB STEP name • JOB STEP STEPLIB DSNs <p>If a PROC is executed:</p> <ul style="list-style-type: none"> • PROC name • PROC STEP • PROC STEP STEPLIB DSNs
Job cross-referenced by FD	<ul style="list-style-type: none"> • JOB name • JOB STEP name • JOB STEP DD name • FD name • FD within the executed program associated with the DD • Records defined to the FD within the executed program <p>If a PROC is executed:</p> <ul style="list-style-type: none"> • PROC name • PROC STEP name • PROC STEP DD name • FD within the executed program associated with the DD • Records defined to the FD within the executed program
JOBCAT cross-referenced by Job	<ul style="list-style-type: none"> • JOBCAT DSN • JOB name
Load Module cross-referenced by CSECT	<ul style="list-style-type: none"> • Load module name • CSECT name, and optionally, the CSECT source library name • Source library member name

Report Name	Displays
Load Module cross-referenced by Job	<ul style="list-style-type: none"> • Load module name • STEP name executing the load module • JOB containing STEP executing the load module <p>If the STEP executing the load module is contained in a PROC, lists:</p> <ul style="list-style-type: none"> • STEP executing the PROC • JOB containing the STEP that executed the PROC • DD and DSN used by the STEP
PROC cross-referenced by Load Module	<ul style="list-style-type: none"> • PROC name • STEP name • PROC load module executed in the STEP • PROC Library where the PROC resides
STEPCAT cross-referenced by Job	<ul style="list-style-type: none"> • STEPCAT DSN • JOB using the STEPCAT • STEP name

2 Select one of the listed reports and press Enter.

Note: _____

For step-by-step instructions on saving and printing query information, see "[Saving and Printing Query/Search Results](#)" on page 43.

3 Press PF3 to close the report and return to the previous screen.

Generating Program Cross-reference Reports

To generate a report with program cross-reference information

- 1 Select View ► Program Cross Reference Information from the Alliance Primary screen and press Enter. The Program Cross Reference Information pop-up displays.

These are the available program cross-reference reports:

Report Name	Displays
Called program cross-referenced by Caller	<ul style="list-style-type: none"> • CALLEd program entry name • CALLEd program name, and optionally, CALLing program name • Source library name where CALLEd program resides • Member name in source library
Caller program cross-referenced by Called	<ul style="list-style-type: none"> • CALLing program name • CALLEd program entry name, and optionally, CALLEd program name • Source library name where CALLing program resides • Member name in source library
CICS Transactions cross-referenced by Program	<ul style="list-style-type: none"> • Transaction name • Program name containing the transaction • Source library where program resides • Member name in source library
Copybook cross-referenced by Program	<ul style="list-style-type: none"> • Copybook name • Name of program using copybook, and optionally, the member and library where copybook resides

Report Name	Displays
Data item cross-referenced by Program	<ul style="list-style-type: none"> • Data item name • Program name containing data item definition • Source library where program resides • Member name containing the program
FD name cross-referenced by Record	<ul style="list-style-type: none"> • FD name using the record • Record name • Data item names in record • Program name containing FD
Program cross-referenced by CICS transaction	<ul style="list-style-type: none"> • Program name • CICS transaction executing the program
Program cross-referenced by File	<ul style="list-style-type: none"> • Program name • FD name within program • DD name associated with the FD in the execution JCL • DSN associated to the DD in the execution JCL
Program cross-referenced by Copybook	<ul style="list-style-type: none"> • Program name • Copybook used by program • Program source library name • Member name of program in source library
Program cross-referenced by Data item	<ul style="list-style-type: none"> • Program name • Dataname • Copybook containing the data item

- 2 Select a report and press Enter.

Note:

For step-by-step instructions on saving and printing query information, see ["Saving and Printing Query/Search Results" on page 43](#).

- 3 Press PF3 to close the report and return to the previous screen.

Running Searches

The Alliance search function familiarizes you with the application components possibly impacted by the change. You can conduct application-wide searches from the Query or the Impact facility to isolate the specific components that match the criteria you define. Save all of the search results to a worklist to use them when you determine the list of components possibly impacted by the change.

Searches help you to determine the components to use for impact analysis. To reduce the number of entities selected for impact analysis without affecting the accuracy of the change assessment, identify synonyms (components that contain the same data) before submitting the list of impacted items for analysis. Identify homonyms (components with the same name) to find components with the same use.

To find additional items requiring impact analysis, identify components that use literals related to the change request. Run searches to narrow the focus of your understanding of the application to the components surrounding the change request. For step-by-step instructions, see ["Searching for Specific Entities" on page 72](#).

Understanding Synonym Relationships

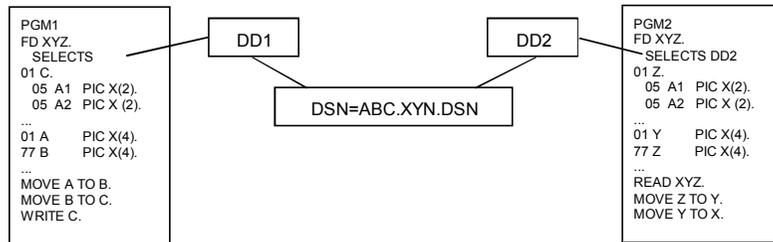
These are the relationships Alliance evaluates when identifying synonyms:

- Assignments
- Parameters
- Renames and redefinitions
- Parents and children
- Implied dependencies
- Indirect dependencies
- Utilities

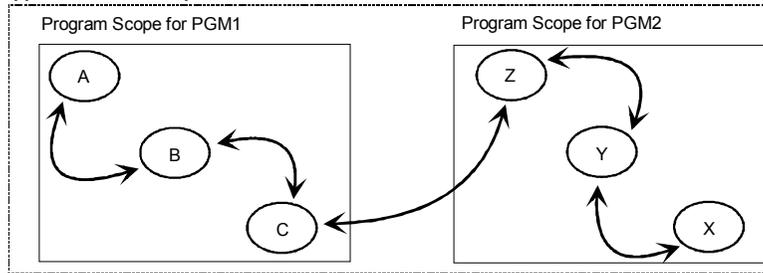
Synonyms and homonyms exist at both application and program (local or global) levels. The same relationships occur at each level and you can specify local, or both local and global levels for each search. Application level synonyms expand the scope of the search to the entire application. Use program level synonyms to limit the search to a single program. This eliminates components outside the selected program from the results. The results reference globally-visible components that have a synonym relationship within the program.

Figure 14 illustrates application and program level synonyms.

Figure 14 • Application and Program Level Synonyms



Application Level Scope



Local Synonyms for A of PGM1

B
C
Local synonyms will not include the connection from C to Z, because it is outside the scope of PGM1.

Application Level Synonyms for A of PGM1

B
C
Z
Y
X

For example, if A is the Employee Name field and B the 401K Participant field, and the data in B comes from A, A and B are synonyms. The synonym relationship is independent of the direction of the data movement between them.

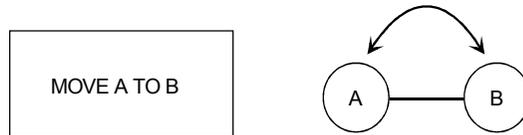
You can control how Alliance calculates synonyms by specifying options to the application analytical engine before you submit the application definition for analysis. For step-by-step instructions on changing analyze options, see the *ASG-Application Definition and Analysis User's Guide*.

Assignments

When Alliance finds a component directly assigning data to another it defines those components as synonyms.

[Figure 15](#) illustrates a synonym relationship by assignment.

Figure 15 • Synonyms - Assignment



A and B have a direct relationship, A and B are Synonyms.

Alliance recognizes these statements as data movement between components:

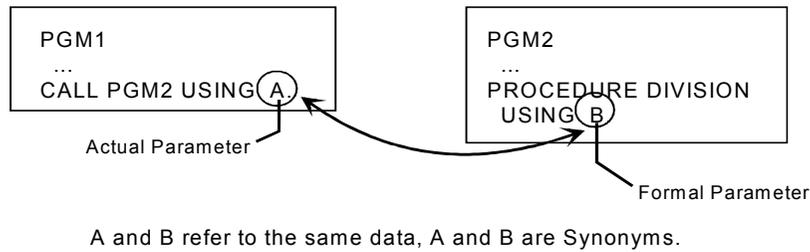
NATURAL	COBOL	PL/I	CICS	IDMS	IMS	SQL
ACCEPT	ACCEPT	ALLOCATE	SEND MAP	ACCEPT	INSERT	FETCH
COMPRESS	MOVE CORRESPONDING	ASSIGN	RECEIVE MAP	END TRANSACTION	GET	UPDATE
GET	MOVE	DECLARE	WRITE	WRITE	REPLACE	
GET SAME	READ	DELETE	READ	FIND		
GET TRANSACTION DATA	REWRITE	LOCATE	READPREV	OBTAIN		
INPUT	WRITE	READ	READNEXT	GET		
REINPUT	SORT	RETURN	REWRITE			
SEPARATE	MERGE	REWRITE				
MOVE	RETURN	WRITE				
MOVE ALL	RELEASE					
READ						
READ WORK FILE						
WRITE						

Parameters

Alliance recognizes actual and formal parameters as synonyms.

[Figure 16](#) illustrates a synonym relationship by parameters.

Figure 16 • Synonyms - Parameters



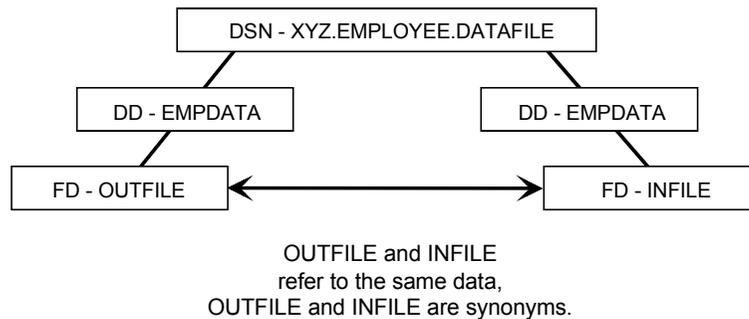
File I/O

Notice that you make the connection from the DD selected within the program to a DD defined within the execution method (e.g., a JCL job) by understanding the load module referenced in the execution method. The load module is affiliated to the program source code and the selected DD within the program is associated to the DD within the execution method.

To make this synonym relationship you must first properly define the load module for the source to the application and define the method of executing the load module to the application inventory (e.g., the execution JCL).

[Figure 17](#) illustrates a synonym relationship by affiliation.

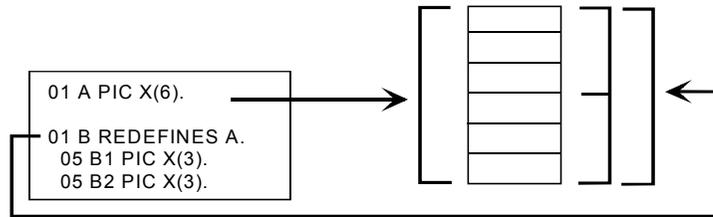
Figure 17 • Synonym Relationship by Affiliation



Renames and Redefinitions

When component A is renamed or redefined as component B, the two components share the same data and are synonyms. [Figure 18](#) illustrates a synonym relationship by rename/redefinition.

Figure 18 • Synonyms - Renames and Redefinitions



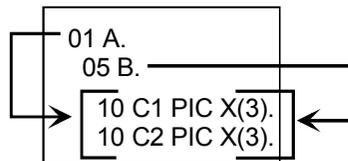
A and B will share the same data, A and B are synonyms.

Parents and Children

In a hierarchical data structure, you can define parent and child components as synonyms when the data they contain overlaps exactly. For example, if A is the parent and B is the only child of A, A and B are synonyms.

[Figure 19](#) illustrates a synonym relationship by hierarchical definition.

Figure 19 • Synonyms - Parent/child Hierarchy



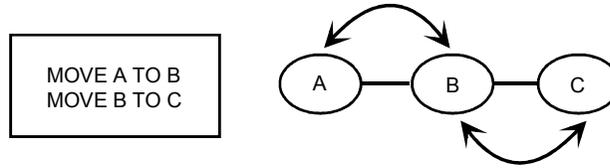
A and B refer to the same data, A and B are synonyms

Indirect Dependencies

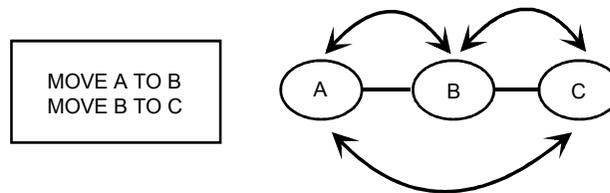
When you define two components as synonyms, the synonyms of one become the synonyms of the other. For instance, when A and B are defined as synonyms, all synonyms of A indirectly become synonyms of B.

[Figure 20](#) illustrates a synonym relationship by indirect dependency.

Figure 20 • Synonyms - Indirect Dependencies



Working from the previous example, a direct synonyms relationship exists between A and B. The second statement creates a direct synonyms relationship between B and C.

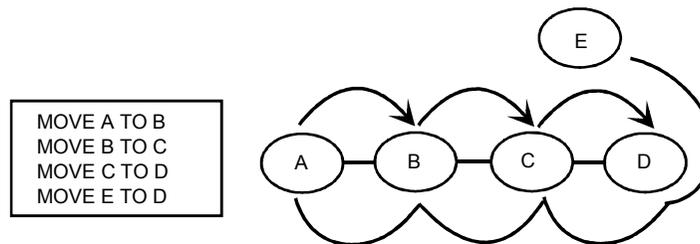


The relationship between A and B, and B and C causes an indirect dependency between A and C, through B.

When two components are identified as synonyms, synonyms of one become synonyms of the other. These relationships are indirect dependencies regardless of the direction of the assignment chain. When A and B are defined as synonyms, all synonyms of B become indirect synonyms of A.

[Figure 21](#) illustrates a multi-level synonym relationship by indirect dependency, regardless of the direction of the relationship.

Figure 21 • Indirect Dependencies - Multiple Levels



C, D and E have an indirect relationship to A.

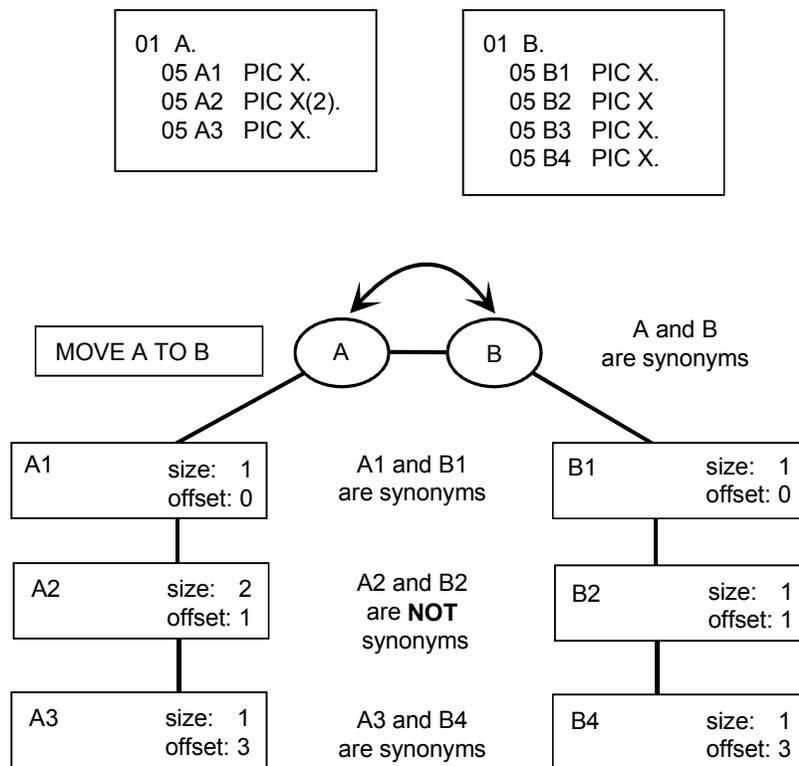
A, B, C, D and E are synonyms of each other. Note, E is indirectly related to A regardless of the direction of the data flow.

Implied Dependencies

When two components are identified as synonyms, and each component has hierarchical children, the children of each component are checked for size and offset to determine if they are synonyms of the other component's children. When A and B are defined as synonyms, A1 is checked against B1, A2 is checked against B2, and so on. For example, each child of A is compared to the children of B for size and offset to determine if the compared items are synonyms of each other.

[Figure 22](#) illustrates synonym relationships by implied dependency.

Figure 22 • Implied Synonyms from Hierarchical Children



Using Utilities

When an input file passes data to an output file through a utility program, the files are considered equivalent and are identified as synonyms.

This table lists the utilities Alliance supports and the names of their input and output files. Field rearrangement by the utilities is not supported.

Utility	DD for Input File	DD for Output File
ICEMAN	SORTIN	SORTOUT, SORTOF _{nn}
IDCAMS	(all)	(all)
IEBCOPY	SYSUT _n	SYSUT _n
IEBGENER	SYSUT _n	SYSUT _n
IERRCO00	SORTIN	SORTOUT, SORTOF _{nn}
IGHRCO00	SORTIN	SORTOUT, SORTOF _{nn}
SORT/MERGE	SORTIN	SORTOUT, SORTOF _{nn}
SYNCSORT	SORTIN	SORTOUT, SORTOF _{nn}

Searching for Application Level Synonyms

Perform searches from both the Impact and the Query facility. You can access these facilities from the Facility pull-down on the Alliance Primary screen.

To search for application-level synonyms

- 1 Select Facility ▶ Impact from the Alliance Primary screen and press Enter. The Impact Facility displays.

Or

Select Facility ▶ Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.
- 2 Select Search ▶ Synonyms and press Enter.
- 3 Select Data Item Synonyms ▶ Application-level synonyms and press Enter.

- 4 In the Dataitem field of the Synonym Criteria pop-up, choose one of these actions:
 - To only list data items with an exact name, type the complete name of the data item.
 - To list data items that have a pattern in their name, type a text pattern.
 - To list all data items in the application, leave the asterisk (*).

Note: _____

ASG recommends that you not list all synonyms, as it is time intensive.

- 5 Press Enter to initiate the search for data items that match the criteria you defined.
- 6 The Synonym Selection pop-up displays showing the list of data items. Specify the data items Alliance identifies synonyms for.
- 7 Type any non-blank character in the selection column beside the data items you want to select and press PF4 (Synonyms) to initiate the search for synonyms.
- 8 The Synonym View screen displays showing the list of synonyms Alliance identified. The data items you selected are highlighted with their synonyms listed below. Use the standard ISPF scroll keys to display the qualification information for each synonym.

Note: _____

For step-by-step instructions on saving and printing search information, see ["Saving and Printing Query/Search Results" on page 43](#).

- 9 Press PF3 and then Enter to exit the Synonym View pop-up. The search results are not saved.

Searching for Program Level Synonyms

Program-level (or local) searches identify synonyms within a selected program.

To search for program-level synonyms

- 1 Select Facility ▶ Impact from the Alliance Primary screen and press Enter. The Impact Facility displays.

Or

Select Facility ▶ Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.

- 2 Select Search ▶ Synonyms and press Enter.
- 3 Select Data Item Synonyms ▶ Program-level synonyms and press Enter.
- 4 Type any character in the selection column beside the program displayed in the Synonym - Program Selection pop-up and press Enter.
- 5 In the Dataitem field of the Synonym Criteria pop-up, choose one of these actions:
 - To only list data items with an exact name, type the complete name of the data item.
 - To list data items that have a pattern in their name, type a text pattern.
 - To list all data items in the application, leave the asterisk (*).
- 6 Press Enter.
- 7 The Synonym Selection pop-up displays showing the list of data items. Specify the data items Alliance identifies synonyms for.
- 8 Type any non-blank character in the selection column beside the data items you want to select and press PF4 (Synonyms).
- 9 The Synonym View screen displays listing the identified synonyms. The data items you selected are highlighted with their synonyms listed below them. Use the standard ISPF scroll keys to display the qualification information for each synonym.

For step-by-step instructions on saving and printing search information, see ["Saving and Printing Query/Search Results" on page 43](#).
- 10 Press PF3 and then Enter to exit the Synonym View pop-up. The search results are not saved.

Searching for Homonyms

Alliance defines homonyms as data elements that have identical names across an application. Data elements with identical names usually are used with similar intent. Identifying them enables you to ensure thoroughness in implementing change requests. You can use this information to accomplish data rationalization tasks as well. Because data elements that come from the same copybook (include) have the same name, Alliance provides an option to remove them from the homonym output.

To search for homonyms

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility displays.

Or

Select Facility ► Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.
- 2 Select Search ► Homonyms and press Enter.
- 3 In the Name field of the Homonym Criteria pop-up, choose one of these actions:
 - Type the complete name of the data item to list only data items with that exact name.
 - Type a text pattern to list data items that have that pattern in their name, as well as the number of definitions for that dataname.
 - Leave the asterisk (*) to list all data items in the application. This is a time intensive procedure and ASG does not recommend it.
- 4 Type any non-blank character in the selection column beside the Include Copybook Items field if you want Alliance to count items in copybooks as homonyms when the copybook appears in more than one program.
- 5 Press Enter. The Homonym Selection pop-up displays a data item list. Specify the data items Alliance identified homonyms for.
- 6 Type any character in the selection column beside the data items you want to select and press PF4 (Homonyms).

The Homonym View screen displays a list of homonyms Alliance identified. Each dataname definition is listed with this information:

- Homonym name
- Definition level of the dataname
- Defined picture of the dataname
- Program name
- Qualification for the dataname

- 7 Use the standard ISPF scroll keys to display the qualification information for each homonym identified.

For step-by-step instructions on saving and printing search information, see [Saving and Printing Query/Search Results](#).

- 8 Press PF3, then Enter to exit without saving the search results.

Saving and Printing Query/Search Results

To save search results information to a new worklist

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility displays.

Or

Select Facility ► Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.

- 2 Choose Worklist ► New Worklist and press Enter. The New Worklist pop-up displays.
- 3 Type a unique name and description. You may also provide an optional dataset name. Press Enter.

If you do not provide a dataset name, the Impact or Query Facility screen returns with the name of the new worklist displayed in the message area. For example:

```
<userid>.ALNnnnnn.VIAWKLIST
```

where:

<userid> is your TSO user ID or prefix.

nnnnn is a sequential number assigned by Alliance.

To save search results to an existing worklist

- 1 Select Facility ▶ Impact from the Alliance Primary screen and press Enter. The Impact Facility displays.

Or

Select Facility ▶ Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.

- 2 Select Worklist ▶ Open Worklist and press Enter. The Open Worklist pop-up displays.
- 3 Type any character in the selection column beside the name of the worklist you want to edit and press Enter.

To edit an existing worklist

- 1 Select Facility ▶ Impact from the Alliance Primary screen and press Enter. The Impact Facility displays.

Or

Select Facility ▶ Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.

- 2 Select Worklist ▶ Edit worklist and press Enter. The Open Worklist pop-up displays.
- 3 Type any character in the selection column beside the name of the worklist you want to edit and press PF4. The worklist displays on your default editor screen.

Note: _____

If SmartEdit is installed at your site, the SmartEdit editor displays. Otherwise the ISPF editor displays.

- 4 Select File ▶ Save to save your changes.
- 5 Select File ▶ Exit to return to the original screen.

To print search results

- 1** Select Facility ▶ Impact from the Alliance Primary screen and press Enter. The Impact Facility displays.

Or

Select Facility ▶ Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.

- 2** Select File ▶ Print current results and press Enter.

The search results are copied to an Alliance list file.

- 3** Select Options ▶ Log/list and press Enter. The Options - Log/list Definition pop-up displays.
- 4** Type PK (Print/Keep) or PD (Print/Delete).
- 5** Press PF6 to close and process the list file.

4

Using the Query Facility

This chapter describes how to perform queries and searches, and contains these sections:

Section	Page
Performing Custom Queries	47
Understanding Query Language Syntax	48
The ATTRIBUTES BLOCK	52
The SELECTION BLOCK	63
Using Queries	65
Building the List of Possible Targets (Impact List)	75

To refine the information collected so far, determine the selection criteria to locate the unique entity kinds and names constituting the impact list. The more narrow the scope of the analysis, the more accurate the information obtained for the impact assessment.

Use the Alliance query language to customize the template queries or to create new template queries to display application information about the entities included in the analysis. You can also run new searches to locate entities impacted by the change.

Performing Custom Queries

To locate specific entities in your application and their relationships with other entities, modify the existing queries or create new queries using the syntax supported by Alliance.

To locate specific entities in your application and their relationships with other entities

- 1 Use your system editor to create a file that contains the query source.

- 2 Modify or specify the source as needed and define the new query to Alliance.
- 3 Execute the query.

Understanding Query Language Syntax

The structure of a Alliance query consists of these definition blocks:

Definition Block	Description
ENTITY_RELATIONSHIP BLOCK	Names the application entities (such as programs, FDs, PROCs, JCL) you want to include in the query and the relationships you want to use in the search for application information.
ATTRIBUTES BLOCK	Selects data to be included in the query results and changes the format.
SELECTION BLOCK	Allows AND/OR logic between attributes or entities, and grouping of elements with the Select statement.

Query Structure Example

[Figure 23](#) shows an example of a query structure that displays all FDs used in all programs.

Figure 23 • Query Structure Example

```
000001 ENTITY_RELATIONSHIP BLOCK
000002 *
000003     FD IN PROGRAM
000004 *
000005 ATTRIBUTES BLOCK
000006 *
000007     FD
000008         WIDTH 12
000009         BLANK WHEN REPEATED
000010         SORT DESCENDING
000011         BREAK INSERT BLANK LINE
000012 *
000013     PROGRAM
000014         WIDTH 12
000015         BLANK WHEN REPEATED
000016         SORT DESCENDING
000017         BREAK INSERT PAGE
```

Query Results Example

Figure 24 shows the results of the query structure specified in Figure 23 on page 48. This table contains the specified formatting instructions:

Structural Element	Description
WIDTH <i>n</i>	Specifies user-defined column widths.
BREAK INSERT BLANK LINE	Inserts blank line between each change in the FD name.
BREAK INSERT PAGE	Inserts page break for each change in the program name.
SORT DESCENDING	Sorts results in descending order on both the FD and the program columns.

Figure 24 • Query Results Example

```

Name:   EXQUERY1 - EXAMPLE QUERY 1
FD      PROGRAM
-----
***** TOP OF DATA *****
MASTERIN          VIAIDEMO
MASTER-RPT       VIAIDEMO
-----|Page break|
DAILY TOTALS     VIAIDEM1
***** BOTTOM OF DATA *****

```

Query Syntax Rules

These are the query syntax rules:

- The first two keywords in the query definition must be ENTITY_RELATIONSHIP BLOCK in capital letters.
- Query statements can start anywhere on the line.
- Multiple statements are allowed in a single line.
- Comment lines, identified with a period (.) or asterisk (*) placed in column 1, can be located anywhere in the query definition.

For readability, start each phrase on a new line and use comment lines to separate major sections. [Figure 25](#) illustrates these syntax rules.

Figure 25 • Query Language Syntax - ENTITY_RELATIONSHIP BLOCK Example

```
000001 .This is a comment line
        ENTITY_RELATIONSHIP BLOCK
000003
000004     entity1 relationship1 entity2
000005     entity2 relationship2 entity3
000006
000007
000008     entityN relationshipN entityN+1
```

The ENTITY_RELATIONSHIP BLOCK

Use the ENTITY_RELATIONSHIP BLOCK to specify the chain of entities displayed. Entities linked by relationships are also specified in the ENTITY_RELATIONSHIP BLOCK. The valid links are specified in ["Valid Entities and Relationships," on page 115](#). Internally, this block specifies the analysis data from the AKR that Alliance uses to derive the query results.

If the ENTITY_RELATIONSHIP BLOCK specifies, for example:

```
FD IN PROGRAM
```

then the default query results show all FDs that occur in all PROGRAMs. This is the way the display is formatted:

```
FD                                PROGRAM
-----
```

On the other hand, if the ENTITY_RELATIONSHIP BLOCK specifies, for example:

```
FD IN PROGRAM
PROGRAM IN MEMBER
```

then the query results show all FDs that occur in all programs and the LIBRARY MEMBERS containing the PROGRAMs. This is the way the display is formatted:

```
FD                                PROGRAM                                MEMBER
-----
```

With each additional entity specified, Alliance adds a new column with default entity titles and field widths to the display.

In the previous example, you might be inclined to change the display to show all FDs and the library MEMBERS containing FDs by using the statement FD IN MEMBER, but this is not a valid entity-relationship combination.

Note: _____

To prevent the display of PROGRAM when the column width of PROGRAM is set to zero, use the ATTRIBUTES BLOCK statement. For more information, see ["The ATTRIBUTES BLOCK" on page 52](#).

These are the ENTITY_RELATIONSHIP BLOCK syntax rules:

- The first entity specified in the ENTITY_RELATIONSHIP BLOCK is called the root entity for the query (for example, entity1 in Query language syntax - ENTITY_RELATIONSHIP BLOCK example). This is any valid application entity.
- All remaining links must begin by naming an entity specified in a previous line. Otherwise, the syntax does not form a valid entity.

You can perform these tasks:

- Specify a continuous search path through the application information by repeating an entity that occurred at the end of the preceding line.

For example:

- Entity2 in the Query Language Syntax - ENTITY_RELATIONSHIP BLOCK Example displays at the end of the first line and the beginning of the second.
- Entity1 is related to entity2, entity2 is related to entity3, and so on.

This forms the continuous search path.

- Begin a new search path through the application information by repeating the first entity that occurs in one line on several subsequent lines. To ensure correct interpretation, ASG recommends that you change the format of the output to delineate the paths. [Figure 26](#) and [Figure 27 on page 52](#) show a query result with multiple paths.

Figure 26 • Example of Multiple Paths in a Query

```

000001 ENTITY_RELATIONSHIP BLOCK
000002 COPYMEMBER IN PROGRAM
000003 COPYMEMBER IN MEMBER
000004 MEMBER IN LIBRARY
000005
000006 ATTRIBUTES BLOCK
000007 COPYMEMBER
000008 ON CHANGE INSERT BLANK LINE

```

The ENTITY_RELATIONSHIP BLOCK syntax shown in [Figure 27](#) specifies a report showing all COPYMEMBERS and the PROGRAMs using them. In addition, this query requests the listing of all library MEMBERS containing the COPYMEMBER and the LIBRARY(s) containing those MEMBERS.

Figure 27 • Query Results Showing Multiple Paths Example

```
Name:      VIABQCPS - Copybook / Program cross-reference

COPYMEMBER  PROGRAM      MEMBER      LIBRARY
-----
COPY0005    VIAI040      n/a         n/a          <==COPYMEMBER IN PRORAM
COPY0005    n/a         COPY00005  ASG.VIA.SOURCE <==COPYMEMBER IN MEMBER
                                                MEMBER IN LIBRARY

COPY0010    VIAI040      n/a
COPY0010    n/a         ASG.VIA.SOURCE
```

For example:

- The COPYMEMBER is in the PROGRAM and also in the library MEMBER.
- The library MEMBER is in the LIBRARY.
- The PROGRAM is not in the MEMBER or in the LIBRARY.
- The MEMBER and LIBRARY columns are for the COPYMEMBER and not for the PROGRAM.

To avoid ambiguity, the display shows separate output on different lines with one line for the COPYMEMBER IN PROGRAM relationship, and one line for the COPYMEMBER IN MEMBER and the MEMBER IN LIBRARY relationship. The letters n/a display where there is no relationship.

The ATTRIBUTES BLOCK

After entities and their relationships are defined through the ENTITY_RELATIONSHIP BLOCK, the ATTRIBUTES BLOCK lets you perform these tasks:

- Specify attributes of entities and actions on attributes.
- Apply selection criteria to the generated results.
- Modify the format of the results.
- Request additional information for specific entities.

The ATTRIBUTES BLOCK and all of its statements are optional. Specify attributes for any, all, or none of the entities named in the ENTITY_RELATIONSHIP BLOCK. [Figure 28](#) is an example of the ATTRIBUTES BLOCK.

Figure 28 • Query Language Syntax - ATTRIBUTES BLOCK Example

```
ENTITY_RELATIONSHIP BLOCK
  PROGRAM HAS DATA
ATTRIBUTES BLOCK
  PROGRAM
    SORT DESCENDING
    BLANK WHEN REPEATED
  DATAITEM
    SORT ASCENDING
    BLANK WHEN REPEATED
    TITLE "DATA NAMES"
    WIDTH 35
```

These are the ATTRIBUTES BLOCK syntax rules:

- Keywords can be placed anywhere on the line.
- Multiple statements are allowed in a single line.
- The first item in the ATTRIBUTES BLOCK must be an entity named in the ENTITY_RELATIONSHIP BLOCK. Any number of attributes statements can follow this entity name. All statements apply to this entity until another entity is named.
- Optional keywords within each statement are shown between braces ({and}).
- The vertical bar (|) denotes an OR condition, indicating that only one of the keywords can be selected.

For readability, each statement should be indented and separated. These are the valid attributes and actions for the ATTRIBUTES BLOCK:

- SELECT
- TITLE
- WIDTH
- SORT
- FORMAT BLANK WHEN REPEATED
- BREAK ON CHANGE INSERT
- POSITION
- DATAITEM entity attributes
- DSN entity attributes
- LANGUAGE entity attribute

SELECT

The SELECT attribute selects or discards data from the query result producing a subset of the data from the AKR. The SELECT clause is followed by the condition including an operator and selection criterion. Depending on the entity ATTRIBUTE, you might enclose the selection criterion in quotes.

You may specify multiple selection criteria for an entity kind. When you specify more than one selection criterion, each is joined by AND or OR.

This is the syntax of the SELECT clause:

```
SELECT <entity> WHEN <attribute> <operator> <value>
```

where:

entity is PROGRAM, DATAITEM, etc.

attribute is NAME, SIZE, PICTURE, etc.

The WHEN keyword is optional. Parentheses are not allowed. Use an asterisk (*) as a wildcard in the 'text' string to specify characters, for example:

'ABC*' Indicates any name beginning with ABC.

'*ABC' Indicates any name ending with ABC.

'*ABC*' Indicates any name containing ABC.

As the query retrieves data during its execution, the entity value is compared to the string, and then selected or discarded, depending on the operator. You may enter only one select statement for each entity. If you specify, for example:

```
RECORD  
  SELECT WHEN EQ "*MASTER*" AND NE "PAY-MASTER"
```

then only RECORD entities that contain the text MASTER anywhere in their name and that are not named PAY-MASTER are included in the query result.

SELECT criteria are evaluated sequentially, by order of appearance. Every SELECT criterion is evaluated. When all SELECT criteria have been evaluated, the data selection indicator is tested to determine whether the data should be included in the results. Use multiple SELECT criteria carefully to ensure that the results contain the data you expect. If you specify, for example:

```
RECORD
  SELECT WHEN EQ "*MASTER*" AND NE "PAY-MASTER"
  OR EQ "*TRANSACTION*" AND NE "HOLD-TRANS"
```

then the query result includes:

- RECORD entities that contain MASTER anywhere in their name and are not named PAY-MASTER
- RECORD entities that contain TRANSACTION anywhere in their name and whose name does not begin with HOLD-TRANS

TITLE

The TITLE attribute changes the column title. The default column title is the entity kind name exactly as it is specified in the ENTITY_RELATIONSHIP BLOCK (including capitalization). For example, use this attribute to change the FD entity title to File Description. The title itself can be any text and must be enclosed in single or double quotes, provided that the ending quote matches the starting quote. This feature enables you to embed quotes in the title, as in "Job's PROC".

WIDTH

The WIDTH attribute changes the width of the column. The default width is wide enough to prevent truncation. You must specify a positive numeric integer for the column width. If the WIDTH attribute specifies, for example:

```
WIDTH 10
```

then the column width for the entity is 10 characters.

To hide a column, specify a width of zero (0). Although this prevents entities from being displayed, they are still required in the query syntax to define the search path for the retrieval of application data from the AKR.

For example, you could modify the query defined in [Figure 26 on page 51](#) to include a WIDTH 0 statement for the entity MEMBER, as shown in [Figure 29](#).

Figure 29 • Example of WIDTH 0 Attribute in a Query

```
000001 ENTITY_RELATIONSHIP BLOCK
000002 COPYMEMBER IN PROGRAM
000003 COPYMEMBER IN MEMBER
000004 MEMBER IN LIBRARY
000005
000006 ATTRIBUTES BLOCK
000007 COPYMEMBER
000008 ON CHANGE INSERT BLANK LINE
000009
000010 MEMBER
000011 WIDTH 0
```

[Figure 30](#) shows the result of the modified query.

Figure 30 • Example of Query Results Showing WIDTH 0

```
Name: VIABQCPS - Copybook / Program cross-reference
COPYMEMBER PROGRAM LIBRARY
-----
COPY0005 VIAI040 n/a
COPY0005 n/a ASG.VIA.SOURCE
COPY0010 VIAI040 n/a
COPY0010 n/a ASG.VIA.SOURCE
```

SORT

The SORT attribute sorts the query result in ascending or descending order. You may choose a different sort order for each entity in the result.

This statement must be used for any entity that has a BREAK or a FORMAT BLANK WHEN REPEATED attribute specified.

FORMAT BLANK WHEN REPEATED

The FORMAT BLANK WHEN REPEATED attribute causes blanks to print in the place of the text value of an entity if that text is the same as the preceding line. This statement is used to enhance the readability of the output. The FORMAT keyword is optional, but you must specify the SORT attribute.

BREAK ON CHANGE INSERT

The BREAK ON CHANGE INSERT attribute causes a blank line or a page break to be inserted between two lines in the query result when the value of an entity changes on two consecutive lines. Use this statement to enhance the readability of the output.

On the screen, a page break is denoted by a line of dashes (- - -). For printed reports, a physical page break occurs. The BREAK keyword is optional.

POSITION

The POSITION attribute is used to place a column of the output before or after another. This statement does not affect the order data is retrieved. Specify the target position by name or by column number.

For example, if the only selected entities are FD and PROGRAM, you could specify the FD attribute POSITION AFTER PROGRAM or POSITION AS 2 to display the PROGRAM column first. Specifying the FD attribute POSITION BEFORE PROGRAM is unnecessary because that is the default order.

For example, to position a column after a dataitem attribute such as LEVEL, type this statement:

```
"POSITION AFTER DATAITEM: LEVEL"
```

DATAITEM Entity Attributes

Include extended entity attributes in the ATTRIBUTES BLOCK for the entity kind DATAITEM. These entity attributes provide additional columns in the query result to display more information about a data item entity besides its default NAME attribute.

In addition to DATAITEM attributes, the CRUD attributes (Create, Read, Update, and Delete) are now valid on these entities:

- FD
- DD
- IDMS record
- DB2 table
- IMS segment

Include multiple entity attributes in a query, as shown in [Figure 31](#). The entity ATTRIBUTE keyword is required for each attribute included in the query. All attributes can be used with entity attributes.

Figure 31 • Example of Multiple Entity Attributes in a Query

```

000026  ATTRIBUTES BLOCK
000027      DATAITEM
000028          TITLE "DATA ITEM"
000029          SORT ASCENDING
000030          FORMAT BLANK WHEN REPEATED
000031          ATTRIBUTE REF
000032          ATTRIBUTE SIZE
000033          ATTRIBUTE LEVEL
000034          ATTRIBUTE OFFSET
000035          ATTRIBUTE POSITION
000036          ATTRIBUTE REDEFINES
    
```

This table lists valid entity attributes in alphabetical order. [Figure 32 on page 60](#) and [Figure 33 on page 61](#) show a query where these attributes are used.

Attribute Name	Description	SELECT Clause Must Specify
ARITH	Specifies if a data item is part of an arithmetic statement.	Positive number or zero, not enclosed in quotes
COND	Specifies if a data item is part of a conditional statement.	Positive number or zero, not enclosed in quotes
CREATE	Specifies if a data item is part of a create activity, as defined by ESW.	Y or N
DELETE	Specifies if a data item is part of a delete activity, as defined by ESW.	Y or N
LANGUAGE	Specifies the language that generates the entity kind. See " LANGUAGE Entity Attribute " on page 62 for more details.	COBOL, PL/I, ASM, or NATURAL
LEVEL	Indicates a level number of the data item, as coded in source module.	Positive number or zero, not enclosed in quotes
MOD	Indicates the number of modifications of data item, as defined by ESW. This includes occurrences of the dataname where its value is being tested or used.	Positive number or zero, not enclosed in quotes

Attribute Name	Description	SELECT Clause Must Specify
OCCURS	Specifies if a data item is the subject of, or is subordinate to the subject of an OCCURS clause. The resulting display shows the number of occurrences, as specified in the OCCURS clause. A blank or zero result indicates that the data item has no associated OCCURS clause.	Positive number not enclosed in quotes
OFFSET	Indicates the distance (in bytes) from the start of the record to the data item entity (that is, how many bytes in the record precede the data item).	Positive number or zero, not enclosed in quotes
PICTURE	Lists the PICTURE clause of the data item. The PICTURE column does not identify packed fields. To determine whether field is packed, use entity attribute SIZE in conjunction with PICTURE.	Alphanumeric character enclosed in quotes
POSITION	Indicates the first byte of the data item relative to the owning record (that is, the position of the first byte of the data item).	Positive number, not enclosed in quotes
READ	Specifies if the data item is part of a read activity, as defined by ESW.	Y or N
REDEFINES	Lists the object of a REDEFINES clause or a data item that is subordinate to the object of a REDEFINES clause.	Alphanumeric character enclosed in quotes
REF	Indicates the number of references to the data item, as defined by ESW. This includes occurrences of the dataname where its value is being tested or used.	Positive number or zero, not enclosed in quotes
SIZE	Lists length (in bytes) of the data item field.	Positive number or zero, not enclosed in quotes
UPDATE	Specifies if the data item is part of an update activity, as defined by ESW.	Y or N

Attribute Name	Description	SELECT Clause Must Specify
USE	Indicates the number of uses of the data item, as defined by ESW. This includes occurrences of the dataname where its value is being tested or used.	Positive number, not enclosed in quotes
VALUE	Lists the content of a VALUE clause for the data item.	Alphanumeric character enclosed in quotes

You can include these in the statement, MOVE A TO B:

- USE and REF with data item A
- MOD and REF with data item B

You can include these in the statement, IF A = B:

- USE, REF and COND with data item A
- USE, REF and COND with data item B

In [Figure 32](#), the query includes the entity attributes LEVEL, SIZE, PICTURE, and REDEFINES. Notice that the entity attribute REDEFINES uses the WIDTH attribute.

Figure 32 • Example of DATAITEM Entity Attributes in a Query

```

ENTITY_RELATHIPSHIP BLOCK
  DATAITEM DEFINED-IN PROGRAM

ATTRIBUTES BLOCK
  DATAITEM
    TITLE "Data item"
    SORT ASCENDING
    FORMAT BLANK WHEN REPEATED
    ATTRIBUTE LEVEL
    ATTRIBUTE SIZE
    ATTRIBUTE PICTURE
    ATTRIBUTE REDEFINES
      WIDTH 15

PROGRAM
  TITLE "PROGRAM"
  SORT ASCENDING
  FORMAT BLANK WHEN REPEATED
  
```

[Figure 33](#) shows the result of the modified query.

Figure 33 • Example of Query Results with DATAITEM Entity Attributes

Query Facility				RW-ONLY
Command ==>	-----			Scroll ==> CSR
Name: DTNMATTR - SAMPLE QUERY TO SHOW ATT				1 of 71 >>>
Data item	level	size	picture	redefines
CCVS-C-1	01		119	
CCVS-C-2	01		120	
CCVS-E-1	01		120	
CCVS-E-2	01		100	
CCVS-E-2-2	02		48	
CCVS-E-3	01		120	
CCVS-E-4	01		50	
CCVS-E-4-1	02		3 XXX	
CCVS-E-4-2	02		3 XXX	
CCVS-H-1	01		120	
CCVS-H-2	01		120	
CCVS-H-3	01		120	
CCVS-PGM-ID	01		5 X(5)	
CM-18V0	03		20	COMPUTED-A

DSN Entity Attributes

Include extended entity attributes in the ATTRIBUTES BLOCK to provide information about a dataset name entity for the entity kind DSN, in addition to its default attribute name.

The entity attribute DISPOSITION specifies the characteristics of a DSN, as established in the JCL member of reference. The SELECT clause for the DISPOSITION attribute must contain NEW, OLD, MOD, or SHR. The DISPOSITION entity attribute is valid for DSN only.

[Figure 34](#) is an example of the syntax for a query containing a DSN entity with the DISPOSITION attribute.

Figure 34 • Example of a DSN Entity with the DISPOSITION Attribute in a Query

```

000001 ENTITY_RELATIONSHIP BLOCK
000002 DSN IN DD
000003 ATTRIBUTES BLOCK
000004     DSN
000005     TITLE 'DATASET NAME'
000006     ATTRIBUTE DISPOSITION

```

[Figure 35](#) shows the results of a query containing a DSN entity with the DISPOSITION attribute.

Figure 35 • Example of Query Results with a DSN Entity and the DISPOSITION Attribute

```

Query Facility
Command ==> ----- Scroll ==> CSR
Name: DSNWDISP - A SAMPLE QUERY TO SHOW DSN WITH DISP 1 of 44
-----
DATASET NAME disposition DD
-----
DEV.LOADLIB SHR STEPLIB
DEV.F7AP SHR F7AP
DEV.CDS.MONITOR SHR CDMONRAN
DEV.INPUT.TOCDS SHR CDPAYRIN
DEV.CDS.VLDTRNS NEW CDPYROUT
DEV.LOADLIB SHR STEPLIB
DEV.F7AP SHR F7AP
DEV.CDS.REFERENCE SHR TEXTREF
DEV.CDS.MONITOR SHR CDMONRAN
DEV.CDS.VLDTRNS SHR CDPAYRIN
DEV.CDS.CDCHQREF NEW CDCHQREF
DEV.CDS.BACSFIL NEW BACSTAPE
DEV.LOADLIB SHR STEPLIB
DEV.CDS.REFERENCE SHR TEXTREF
    
```

LANGUAGE Entity Attribute

The entity attribute LANGUAGE displays the language that generated the entity kind. The languages include COBOL, PL/I, Assembler, and NATURAL. The LANGUAGE attribute is valid for these entities:

- PROGRAM/PROCEDURE/FUNCTION
- ENTRY
- FD/FILEVAR/DCB
- RECORD/STRUCTURE/DSECT
- DATAITEM
- CSECT

[Figure 36](#) shows a query that includes the entity attribute LANGUAGE.

Figure 36 • Example of the LANGUAGE Attribute in a Query

```

000001 ENTITY_RELATIONSHIP BLOCK
000002 PROGRAM HAS DATAITEM
000003
000004 ATTRIBUTES BLOCK
000005 DATAITEM WIDTH 12
000006 ATTRIBUTE LANGUAGE WIDTH 6 TITLE "LANG"
000007 ATTRIBUTE SIZE WIDTH 4
000008 ATTRIBUTE OFFSET WIDTH 4 TITLE "OFST"
000009 ATTRIBUTE CREATE WIDTH 1 TITLE "C"
000010 ATTRIBUTE READ WIDTH 1 TITLE "R"
000011 ATTRIBUTE UPDATE WIDTH 1 TITLE "U"
000012 ATTRIBUTE DELETE WIDTH 1 TITLE "D"
000013 ATTRIBUTE POSITION WIDTH 3 TITLE "POS"
000014 ATTRIBUTE LEVEL WIDTH 3 TITLE "LVL"
000015 ATTRIBUTE OCCURS WIDTH 3
000016 ATTRIBUTE PICTURE WIDTH 10
000017 ATTRIBUTE VALUE WIDTH 5
    
```

[Figure 37](#) shows the results of a query that includes the entity attribute LANGUAGE.

Figure 37 • Example of Query Results with the LANGUAGE Attribute

Query Facility											ABBEY-AL		
Command ==>										Scroll ==>	CSR		
Name: PGMWLANG - A SAMPLE QUERY TO SHOW PROGRAM WITH LAN										1 of 200...			
PROGRAM	DATAITEM	LANG	SIZE	OFST	C	R	U	D	POS	LVL	URS	PICTURE	VALUE
CD05A384	BACS-DATE	COBOL	6	0	N	N	N	N	1	01			
CD05A384	BACS-DD	COBOL	2	0	N	N	N	N	1	05		99	
CD05A384	BACS-MM	COBOL	2	2	N	N	N	N	3	05		99	
CD05A384	BACS-YY	COBOL	2	4	N	N	N	N	5	05		99	
CD05A384	BACS-DATE-XX	COBOL	4	0	N	N	N	N	1	01			
CD05A384	BACS-DATE-XX	COBOL	3	1	N	N	N	N	2	03		XXX	
CD05A384	NEXT-BACS-DA	COBOL	4	0	N	N	N	N	1	05		S9(8)	
CD05A384	WA-ACCUMULAT	COBOL	15	0	N	N	N	N	1	01			
CD05A384	WA-NO-OF-WOR	COBOL	3	0	N	N	N	N	1	05		S9(5)	+0
CD05A384	WA-POUT-REC-	COBOL	3	3	N	N	N	N	4	05		S9(5)	+0
CD05A384	WA-POUT-REC-	COBOL	3	6	N	N	N	N	7	05		S9(5)	+0
CD05A384	WA-POUT-CASH	COBOL	6	9	N	N	N	N	10	05		S9(9)V99	+0
CD05A384	WD-DYNAMIC-C	COBOL	80	0	N	N	N	N	1	01			
CD05A384	DATE8	COBOL	8	0	N	N	N	N	1	05		X(8)	'SSM6

The SELECTION BLOCK

After you define the selection criteria in the ATTRIBUTES BLOCK, the SELECTION BLOCK lets you append additional syntax to a SELECT clause, enhancing query capabilities to allow these functions:

- AND/OR logic between attributes
- AND/OR logic between entities
- Grouping of elements with parentheses '(and)'

The SELECTION BLOCK and all of its statements are optional. Specify attributes for any, all, or none of the entities named in the ENTITY_RELATIONSHIP BLOCK.

[Figure 38](#) is an example of the SELECTION BLOCK.

Figure 38 • Query Language Syntax - SELECTION BLOCK Example

```
ENTITY_RELATIONSHIP BLOCK
  PROGRAM HAS DATAITEM
ATTRIBUTES BLOCK
  PROGRAM
    SORT ASCENDING
  DATAITEM
    SORT ASCENDING
    BLANK WHEN REPEATED
    TITLE "DATA NAMES"
    WIDTH 35
SELECTION BLOCK
  SELECT DATA ITEM WHEN (NAME = '*YEAR*') AND
                        (SIZE > 1)
  SELECT PROGRAM WHEN (NAME = 'ABC*')
```

These are the SELECTION BLOCK syntax rules:

- The first item in the SELECTION BLOCK must be an entity named in the ENTITY_RELATIONSHIP BLOCK. Any number of attribute statements can follow this entity name. All statements apply to this entity until another entity is named.
- Optional keywords within each statement are shown between braces ({and}).
- The vertical bar (|) denotes an OR condition, indicating that you can select only one of the keywords.
- Keywords can be placed anywhere on the line.
- Multiple statements are allowed in a single line.
- For readability, each keyword should be indented and separated.

Using Queries

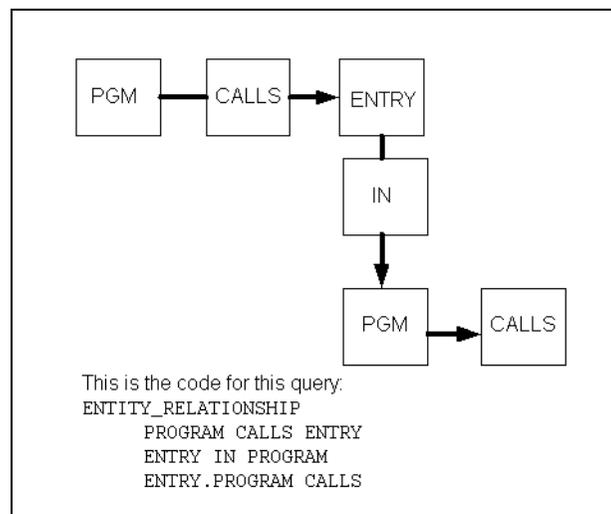
Qualifying Entities in Queries

When an entity is used multiple times within a query, you must qualify it to ensure that the desired search path is created. For example, in a query designed to display program-calling chains, you need to specify the programs and the entries that they call. Then, the query contains the program containing the called entry and any entries that program calls. Repeat this process until the calling chain is exhausted.

In [Figure 39](#), the query contains two references to the entity PROGRAM. The second reference to PROGRAM must be qualified by ENTRY to correctly inform the query processor that ENTRIES called by the second reference to program are to be generated.

If the PROGRAM name was not qualified in this line, it would be unclear whether PROGRAM refers to the calling program or the called program. Without qualification, the first occurrence of PROGRAM is assumed.

Figure 39 • Example of Qualified Entity Names in a Query



Aliasing Entities in Queries

Some queries require additional entity qualification. For example, extensive entity name qualification is necessary to provide the complete search path for the query in order for it to trace the flow of CALLS from program to subprogram on many levels.

[Figure 40](#) shows an example of extensive entity qualification in the ENTITY_RELATIONSHIP BLOCK of a query.

Figure 40 • Example of Extensive Entity Qualification in a Query

```
000001 ENTITY_RELATIONSHIP BLOCK
000002 *
000003 PROGRAM CALLS ENTRY
000004 PROGRAM.ENTRY IN PROGRAM
000005 PROGRAM.ENTRY.PROGRAM CALLS ENTRY
000006 PROGRAM.ENTRY.PROGRAM.ENTRY IN PROGRAM
000007 PROGRAM.ENTRY.PROGRAM.ENTRY.PROGRAM CALLS ENTRY
000008 PROGRAM.ENTRY.PROGRAM.ENTRY.PROGRAM.ENTRY IN PROGRAM
```

In this query, entity qualification quickly becomes difficult to follow. Entity aliasing provides a way to simplify the entity qualification required to complete the query.

To use entity aliasing, follow this step:

- ▶ Specify any name in parentheses after the entity name.

The name in the parentheses becomes the alias for the entity immediately preceding the alias name, including the complete qualification for the entity, as shown in [Figure 41](#).

Figure 41 • Example of Entity Aliasing in the ENTITY_RELATIONSHIP BLOCK of a Query

```
000100 ENTITY_RELATIONSHIP BLOCK
000200 *
000300 PROGRAM CALLS ENTRY
000400 PROGRAM.ENTRY (PROG-ENT-1) IN PROGRAM
000500 (PROG-ENT-1).PROGRAM CALLS ENTRY
000600 (PROG-ENT-1) PROGRAM.ENTRY (PROG-ENT-2) IN PROGRAM
000700 (PROG-ENT-2) PROGRAM CALLS ENTRY
```

In this example:

- PROG-ENT-1 becomes the alias for PROGRAM.ENTRY.
- PROG-ENT-2 becomes the alias for (prog-ent-1).PROGRAM.ENTRY, that is, PROGRAM.ENTRY.PROGRAM.ENTRY.

In both cases, Alliance interprets the actual query statement as if it contains the substituted name.

Use entity aliasing in the ATTRIBUTES BLOCK with the same properties that you used in the ENTITY_RELATIONSHIP BLOCK. [Figure 42](#) shows an example of a query with entity aliasing in the ATTRIBUTES BLOCK.

Figure 42 • Example of Entity Aliasing in the ATTRIBUTES BLOCK of a Query

```

000001 ATTRIBUTES BLOCK
000002 *
000003 PROGRAM
000004 TITLE 'A calls'
000005 SORT ASCENDING
000006 WIDTH 10
000007 PROGRAM
000008 TITLE "Start Pgm"
000009 SORT ASCENDING
000010 (prog-ent-1)
000011 WIDTH 0
000012 (prog-ent-1).PROGRAM
000013 TITLE "A calls"
000014 WIDTH 10

```

For an alphabetical listing of all entities recognized by Alliance, see ["Valid Entities and Relationships" on page 115](#).

Creating Custom Queries

Whether modifying standard queries, editing an open query, or creating a new query, follow these same procedures to generate custom queries.

- Create a new query language source file (see [page 70](#)).
- Define the query to Alliance (see [page 70](#)).
- Execute the query (see [page 71](#)).

Creating New Query Language Source Files

To modify standard queries or to construct entirely new displays, create new files containing the query language source needed to produce the display. These files define the displays that show when you make a selection from the System Cross Reference Information pop-up.

These are the system cross-reference source files:

File Name	Report Name
VIABQCDD	CICS file cross-referenced by program
VIABQMAP	CICS map cross-referenced by program
VIABQDSC	DSN cross-referenced by copybook
VIABQDSN	DSN cross-referenced by job
VIABQDSG	IMS DBD cross-referenced by segment
VIABQSEG	IMS PSB cross-referenced by program
VIABQJBS	Job cross-referenced by JCL library
VIABQJBL	Job cross-referenced by JOBLIB
VIABQLMO	Job cross-referenced by load module
VIABQSTL	Job cross-referenced by STEPLIB
VIABQREC	Job cross-referenced by FD
VIABQJBC	JOBCAT cross-referenced by job
VIABQLMP	Load Module cross-referenced by CSECT
VIABQLMD	Load Module cross-referenced by job
VIABQPRC	PROC cross-referenced by load module
VIABQSTC	STEPCAT cross-referenced by job
VIABQDBF	NATURAL database cross-referenced by Adabas EDT
VIABQDFF	Adabas FDT cross-referenced by FDT field
VIABQDDM	NATURAL database cross-referenced by Natural DDM
VIABQDMF	NATURAL DDM cross-referenced by DDM field

These are the program cross-reference source files:

File Name	Report Name
VIABQCLD	Called program cross-referenced by caller
VIABQCLS	Caller program cross-referenced by called
VIABQTRN	CICS transactions cross-referenced by program
VIABQCPS	Copybook cross-referenced by program JCL INCLUDEs cross-referenced by JCL
VIABQDAT	Dataitem cross-referenced by program
VIABQFD	FD name cross-referenced by record
VIABQPTR	Program cross-referenced by CICS transaction
VIABQFIL	Program cross-referenced by file
VIABQCPY	Program cross-referenced by copybook JCLs cross-referenced by JCL INCLUDEs
VIABQDCP	Program cross-referenced by data item
VIABQPDM	Program cross-referenced by NATURAL DDM
VIABQWKF	Program cross-referenced by NATURAL workfile

To modify a template query source file

- 1 Locate the name of the member containing one of the standard queries.

The files containing the query language source are installed in this dataset:

ASG.CENTER.CNTL

where *ASG* and *CENTER* are defined by the installer at your site.

- 2 Use your system editor to modify the source file as needed. For query language syntax information, see ["Understanding Query Language Syntax" on page 48](#).
- 3 To preserve the original source file, save the changed file under a new member name. Otherwise, the changed version replaces the original query source.

To edit an existing query source file while the query results are displayed

- 1** Select Facility ► Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.
- 2** Select Edit ► Query and press Enter. The system editor screen displays showing the query source file.
- 3** Modify the source file definition as needed.
- 4** Type CREATE to create a new member in the same or a different dataset.
- 5** Type CANCEL to avoid modifying the original query source.

To create a new query source file

- 1** Open your system editor.
- 2** Create a source file.

The file must be sequential or a member of a PDS, and contain 80-byte fixed length records.

Use the information in ["Understanding Query Language Syntax" on page 48](#) to specify the contents of the file.

- 3** Save the file under a new member name in the default dataset (ASG.CENTER.CNTL) or in a different one.

Defining a Query to Alliance

After you create a new query language source file, you must define the query to Alliance.

To define the new query

- 1** Select Facility ► Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.
- 2** Select Facility ► Query and press Enter.
- 3** Select File ► New query and press Enter. The New Query pop-up displays.

- 4 Type the name, description, and dataset name of the file containing the query in the appropriate fields. If the dataset is a PDS, include the member name in the dataset name.

Note: _____

If you enter a dataset member that does not exist, you receive a Member Not Found message. To create the member, press PF4 to display a blank query editor screen .

- 5 To edit the query, press PF4. To immediately generate results, press PF5.
- 6 Press PF3 to record and open the query, or press PF12 to cancel.

Executing a Custom Query

Generate query results from the Query facility after you create a new source file and define the query to Alliance.

For step-by-step instructions on saving and printing query information, see ["Saving and Printing Query/Search Results" on page 43](#).

To execute the query when the query is not open

- 1 Select Facility ► Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.
- 2 Select File ► Open query and press Enter.
- 3 Type any character in the selection area to the left of the query name on the Open query pop-up and press Enter.
- 4 Press PF5 to generate the query. The Query Facility screen displays the query results.

Note: _____

The message QUERY OPEN FAILED displays if a syntax error is encountered. Press PF1 to display more information about the error.

- 5 Press PF3 to close the query and return to the previous screen.

To execute the query when the query is already open

- 1 Select Facility ► Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.
- 2 Select File ► Open query and press Enter. Select a query from the list on the Open Query screen.
- 3 Select Generate ► Query Results and press Enter. The Query Facility screen displays the results of the query.

Note:

The message QUERY OPEN FAILED displays if a syntax error is encountered. Press PF1 to display more information about the error.

- 4 Press PF3 to close the query and return to the previous screen.

Using the Local Synonym Query

The query facility in Alliance enables you to query program-level local synonyms using the LOCALSYNONYM relationship. Issue the Entity-Relationship DATAITEM LOCALSYNONYM DATAITEM within your query to have Alliance generate local synonyms in the given program. The relationship SYNONYM continues to work as before. This generates global synonyms across an application.

Searching for Specific Entities

As the information collected brings you closer to building a list of possible targets, use more involved search options to focus on specific entities possibly impacted by the change. Besides the identification of synonyms and homonyms, use the search function to conduct application-wide searches for the location of specific entities, entity kinds, and literal strings. Run searches from the Impact or the Query facility. For step-by-step instructions on synonym/homonym identification, see ["Running Searches" on page 32](#).

To search for specific entities

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.

Or

Select Facility ► Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.

- 2 Select Search ► Entities and press Enter.

- 3 In the text field of the Search Entities Selection Criteria pop-up, perform one of these actions:
 - To list only entities with an exact name (a maximum of 40 characters), type the complete name of the entity.
 - To list all entities that match a pattern, type the partial name of the entity using wildcards (*).
- 4 Select one or more entity kinds and press Enter. The Search pop-up displays showing the list of entities.
- 5 Type any character in the selection column beside the entities you want to select and press PF4.

The Search View screen displays with the list of entities Alliance identified. Each entity definition is listed with this information:

- Entity name
- Entity kind
- Qualification for the entity name

For step-by-step instructions on saving and printing search information, see ["Saving and Printing Query/Search Results" on page 43](#).

- 6 To exit the Search View pop-up without saving the search results, press PF3 and then Enter.

[Figure 43](#) shows an example of entity search results in the Search View screen.

Figure 43 • Example of Entity Search Results

```

File Edit View Worklist Options Help
-----
Search View                                DE27269
Command ==> _                               Scroll ==> CSR
                                           1 of 176
                                           >>>

SEARCH RESULTS
Base Target      : *
Requested Entities : DATAITEM

Name             Kind      Qualification
-----
A-CHILD1         DATAITEM A-PARENT OF A-RECORD OF MOVE1 OF VIASO
A-CHILD2         DATAITEM A-PARENT OF A-RECORD OF MOVE1 OF VIASO
A-DATE           DATAITEM MOVE1 OF VIASOFT.CUST.COBOL(DE27269)
A-PARENT         DATAITEM A-RECORD OF MOVE1 OF VIASOFT.CUST.COBOL
A-RECORD         DATAITEM MOVE1 OF VIASOFT.CUST.COBOL(DE27269)
B-DATAITEM       DATAITEM MOVE1 OF VIASOFT.CUST.COBOL(DE27269)
B-DATE           DATAITEM MOVE1 OF VIASOFT.CUST.COBOL(DE27269)
C-DATAITEM       DATAITEM MOVE1 OF VIASOFT.CUST.COBOL(DE27269)
C-DATE           DATAITEM MOVE1 OF VIASOFT.CUST.COBOL(DE27269)

```

To search for literal strings

1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.

Or

Select Facility ► Query from the Alliance Primary screen and press Enter. The Query Facility screen displays.

2 Select Search ► Literals and press Enter.

3 In the text field of the Literal Criteria pop-up, type a text search pattern to list entities having that pattern in their name (a maximum of 38 characters). Press Enter. The Literal Selection pop-up displays showing the list of entities.

4 Type any character in the selection column beside the entities you want to select and press PF4. The Literals View screen displays with the list of entities Alliance identified. Each literal is listed with this information:

- Literal name
- Location of use

For step-by-step instructions on saving and printing search information, see ["Saving and Printing Query/Search Results" on page 43](#).

5 To exit the Literals View pop-up without saving the search results, press PF3 and then press Enter.

[Figure 44](#) shows an example of entity search results in the Literals View screen.

Figure 44 • Example of Literal Search Results

```

File Edit View Worklist Options Help
-----
Command ==> _ Literals View ABBEY-AL
                                     scroll ==> CSR
                                     1 of 8
                                     >>>

Literal Search Results:

Literal                               Location of Use
-----
***** TOP OF DATA *****
+08      VIAAL10.ABBEY.COBOL(CD180PGM)
+100     VIAAL10.ABBEY.COBOL(CD20APGM)
+12      VIAAL10.ABBEY.COBOL(CD10BPGM)
          VIAAL10.ABBEY.COBOL(CD20BPGM)
          VIAAL10.ABBEY.COBOL(CD05APGM)
          VIAAL10.ABBEY.COBOL(CD180PGM)
          VIAAL10.ABBEY.COBOL(CD15APGM)
+1223   VIAAL10.ABBEY.COBOL(CD110PGM)
***** BOTTOM OF DATA *****

```

Building the List of Possible Targets (Impact List)

The impact list is the tool Alliance uses to derive the base selection list, which is the starting point for the impact assessment. An impact list consists of the unique entities (names and kinds) from an application. Use the impact list to find potential impact analysis targets when you generate an impact analysis. You can save the impact list to a worklist for future use in this or in other impact analyses. In addition, you can create an impact list from components in one application and use it to select targets in other applications.

When you have gathered enough cross-reference and synonym/homonym data, start the impact analysis by building the impact list. To build the impact list, you define selection criteria to which Alliance compares all of the entities in the application to identify possible candidates. For step-by-step instructions on starting the impact analysis and generating a base selection list, see ["Selecting Items to be Analyzed" on page 82](#).

[Figure 45](#) shows an example of the Impact Entity List screen.

Figure 45 • Example of an Impact Entity List

```

File Edit View Worklist Options Help
-----
Command ==> _ Literals View ABBEY-AL
Scroll ==> CSR
1 of 8
>>>

Literal Search Results:

Literal          Location of Use
-----
***** TOP OF DATA *****
+08             VIAAL10.ABBEY.COBOL(CD180PGM)
+100            VIAAL10.ABBEY.COBOL(CD20APGM)
+12             VIAAL10.ABBEY.COBOL(CD10BPGM)
                VIAAL10.ABBEY.COBOL(CD20BPGM)
                VIAAL10.ABBEY.COBOL(CD05APGM)
                VIAAL10.ABBEY.COBOL(CD180PGM)
                VIAAL10.ABBEY.COBOL(CD15APGM)
+1223          VIAAL10.ABBEY.COBOL(CD110PGM)
***** BOTTOM OF DATA *****

```

Note: _____

If you do not specify the impact list, Alliance submits all of the entity kinds in the application for analysis.

To create an impact list

- 1 Select File ► Open application from the Alliance Primary screen and press Enter. The File - Open Application pop-up displays (see [Figure 46](#)).

Figure 46 • File - Open Application Pop-up

```
File - Open Application
Command ==> -----
Type AKR information and application name. Then press Enter.
Application Knowledge Repository (AKR):
Data set name . . . 'ASG.ALLSAMP.AKR'
Application name . . . GCCTS-PYBL (blank or pattern for list)
Volume serial . . . ----- (if not cataloged)
Password . . . . . (if password protected)
```

- 2 Type the AKR dataset name in the Dataset name field on the File - Open Application pop-up.
- 3 If you know the name of the application you want to open, type the name in the Application name field and press Enter. Otherwise, leave the field blank.

Or

To choose an application from the list, enter a text pattern and press Enter. The File - AKR Application Director pop-up displays.

- 4 Select Facility ► Impact and press Enter.
- 5 Select File ► New impact list and press Enter.
- 6 Type a unique name and description for the impact list on the New Impact pop-up and press PF4. The Impact Entity List pop-up displays. Since this is a new impact list, there are no defined entities listed. You can also use this pop-up to add and delete entities.

- 7 Press PF4 on the Impact Entity List pop-up. The Impact Entities Selection Criteria screen displays (see [Figure 47](#)).

Figure 47 • Impact Entities Selection Criteria Screen

```

Impact Entities Selection Criteria
Command ==>> _____ Scroll ==>> CSR
Text: *
Entity kind (Default = All) 1 of 51
-----
- AREA
- CATALOG
- CICS-TRANSACTION
- COLUMN
- COPYMEMBER/INCLUDE
- CSD
- CSECT
- CURSOR
- DATABASE
- DATAITEM
- DBD
- DD
- DSN
PF5=Select all PF6=Unselect all

```

- 8 On the Impact Entities Selection Criteria screen, perform one of these actions:
- Type the full name of an entity.
 - Type a pattern for selection in the Text field.

Note: _____

If you enter a pattern and want to limit the entity kinds selected, specify one or more entity kinds by entering any character in the selection column to the left of the entity name.

- 9 Press Enter. The Add Impact List Entity screen (see [Figure 48](#)) displays showing the entity names that matched the specifications on the Impact Entities Selection Criteria screen. The list is scrollable.

Figure 48 • Add Impact List Entity Screen

```

Command ==> _____ Add Impact List Entity _____ Scroll ==> CSR
Select desired entries then press the Add key.                      1 of 3053
Previously selected entries are marked with an asterisk.

Text: *
-----
Text                               Entity Kind
-----
- 'CD05B'                           ENTRY
- 'CD05C'                           ENTRY
- 'CD10A'                           ENTRY
- 'CD10B'                           ENTRY
- 'CD15A'                           ENTRY
- 'CD15B'                           ENTRY
- 'CD20A'                           ENTRY
- 'CD20B'                           ENTRY
- 'CD20C'                           ENTRY
- 'CD20D'                           ENTRY

PF5=Select all  PF6=Unselect all  PF10=Add
    
```

- 10 Type any character in the selection column beside the desired entity name. You can select as many entities as needed.
- 11 When finished selecting entries, press PF10 to add the entries and return to the Impact Entity List screen (see [Figure 49](#)).

Figure 49 • Impact Entity List Screen

```

Command ==> _____ Impact Entity List _____ Scroll ==> CSR
Press PF key for actions. Delete requires selection                 1 of 2695
of entry(s).

Current Sort Key: NAME
-----
Entity Kind                        Name
-----
- PROGRAM/PROCEDURE/FUNCTION      .BACS-DATE
- PROGRAM/PROCEDURE/FUNCTION      .CD05C-ERROR-CODES
- PROGRAM/PROCEDURE/FUNCTION      .DATE6-YY
- PROGRAM/PROCEDURE/FUNCTION      .US-DATE8-YY
- PROGRAM/PROCEDURE/FUNCTION      .US-8-YY
- ENTRY                            'CD05B'
- ENTRY                            'CD05C'
- ENTRY                            'CD10A'
- ENTRY                            'CD10B'
- ENTRY                            'CD15B'

PF4=Add entity  PF5=Delete entity  PF6=Change Sort
    
```

- 12 To add entries, press PF4.
- Or**
- To delete entries, select the entry and press Enter.

- 13** Press PF3 to return to the New Impact pop-up.
- 14** Press PF3 again to return to the Impact Facility screen.
- 15** Select File ▶ Options to save or delete the impact list.

Or

Select Impact Facility ▶ Edit to modify the option list. Use this impact list with any open application.

Press Enter.

5

Determining Impacted Components

This chapter describes the impact analysis and impacted components, and contains these sections:

Section	Page
Selecting Items to be Analyzed	82
Locating Impacted Components and Relationships (Summary Impact Analysis)	83
Listing the Relationships of Impacted Items (Detailed Impact Analysis)	85

As described in the overview, at this stage you identify and list the components possibly impacted by the change and then submit the list to the Alliance Impact facility. To assess the impact of change on the application, Alliance evaluates the flow of data and hierarchical and entity relationships. Alliance uses the list of items selected for impact assessment to derive a base selection list of components and to produce summary and detailed analysis results.

The summary impact analysis identifies and locates the components impacted by the change. The detailed impact analysis retraces the results of the summary analysis by showing the path of each component included in those results to the members of the base selection list. Based on your review of the first round of results, research any unknown entity discovered and then modify the initial parameters to rerun the analysis. Step-by-step instructions on how to investigate the results of the impact analysis are located in ["Modifying the Impact Analysis" on page 97](#).

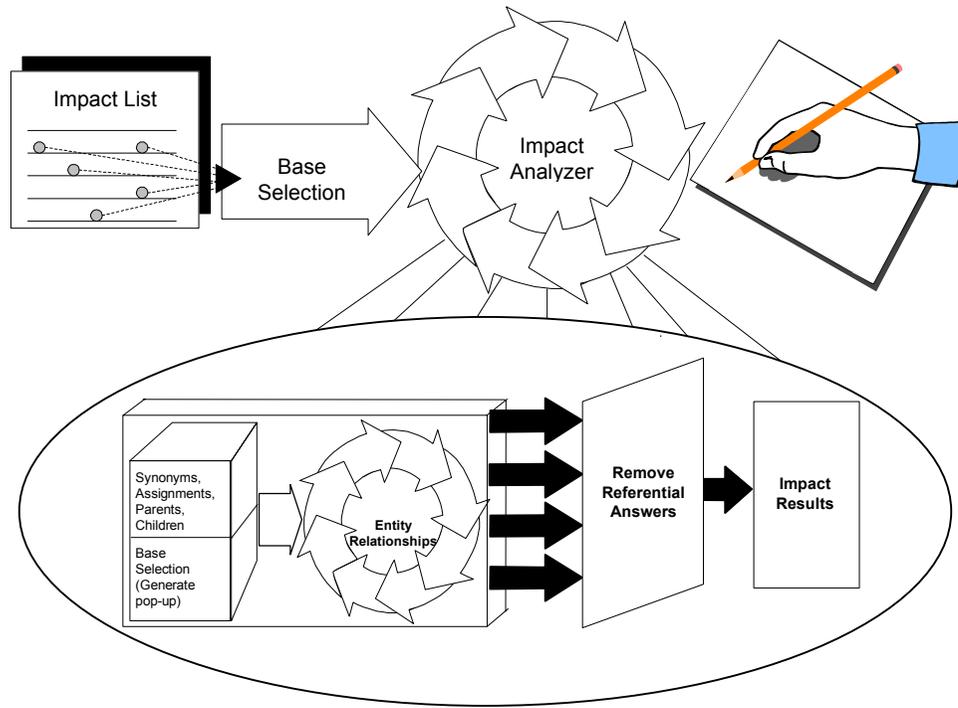
Save each round of results to a worklist. When the impact assessment is completed, you might use results as one of these:

- A guide for the detailed investigation of individual programs, JCL, database definitions, and other components.
- Input to ASG's ESW or other editing tool to implement the required changes.
- A scheduling tool to manage people and resources required to implement the proposed changes.

Selecting Items to be Analyzed

As shown in [Figure 50](#), Alliance automatically creates a base selection list from the entities you select from the impact list during the first summary analysis. If a worklist is open, the base selection list is saved to that worklist.

Figure 50 • The Impact Facility Process



To generate a base impact analysis

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.
- 2 Select File ► Open impact.
Or
Select File ► New impact.

Press Enter.
- 3 Select or enter impact name and press PF5. The Select Impact Generation Entities screen displays.

- 4 Select the items to be analyzed. The list of selected items is called the base selection list (see [Figure 51](#)). It is comprised of the occurrences of entities in the open application that match the open impact list.

Figure 51 • Example of a Base Selection List

```

Summary Impact - Select Impact Generation Entities
Command ==> _____ Scroll ==> CSR
Select the items from which the impact list will      1 of 5435
be generated.

  Text                Entity Kind                Qualification >
-----
- 'CD05B'             ENTRY                YIAAL10.ABBEY.C
- 'CD05B'             ENTRY                YIAAL10.ABBEY.C
- 'CD05C'             ENTRY                YIAAL10.ABBEY.C
- 'CD10A'             ENTRY                YIAAL10.ABBEY.C
- 'CD10B'             ENTRY                YIAAL10.ABBEY.C
- 'CD10B'             ENTRY                YIAAL10.ABBEY.C
- 'CD10B'             ENTRY                YIAAL10.ABBEY.C
- 'CD10B'             ENTRY                YIAAL10.ABBEY.C

PF2=Options  PF4=Generate  PF5=Select all  PF6=Hide entity kind

```

- 5 Use PF2 to review or designate the impact options to be applied to the impact analysis.
- 6 Press PF4 to generate the impact analysis.
- 7 Review the analysis results, change base selection list, and refine impact options, then rerun the impact analysis.

Locating Impacted Components and Relationships (Summary Impact Analysis)

The summary analysis constitutes the first level of impact assessment results. After selecting the items for the base selection list, specify the options that control how the analysis is generated and execute the summary analysis. After reviewing the summary analysis results, generate a detailed analysis to expand the information.

To generate a summary impact analysis

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.
- 2 Select Generate ► Summary and press Enter. The Summary Impact - Select Impact Generation Entities pop-up displays, showing all occurrences of entities in the open application matching those in the open impact list. The type of each entity is listed along with qualification information that provides details on the occurrence location.
- 3 Type any character in the selection column beside the data items you want analyzed and press Enter.

Or

Press PF5 to select all of the entities.

- 4 Press PF2 to view or change the impact generate options (see [Figure 52](#)).

Figure 52 • Impact Generate Options Screen

```

Command ==> Impact Generate Options
-----
Select desired Impact Generate Options.
- Include Base Selection in results
- Include Copy Detail in results

Base Impact Targets:
Targets
-----
/ Base Selection /
/ Parents of Base Selection /
- Children of Base Selection -

Expanded Impact Targets:
Targets
-----
/ Base Impact Targets /
/ Parents of Base Impact Targets /
- Children of Base Impact Targets -

Application Level Synonyms To From Alternates
Application Level Synonyms To From Alternates

PF4=Restore ASG Default Options PF5=Detailed Options
    
```

These options control the type and format of the information displayed as the result of impact analysis, as well as the components of the base selection list. If this your first time performing the impact analysis, ASG recommends that you use the option defaults.

Note:

For step-by-step instructions on modifying default impact generate options, see ["Specifying New Impact Generate Options" on page 108](#).

- 5 Press PF4 to select default options.

- 6 Press PF3 to return to the Summary Impact.
- 7 Press PF4 to generate the summary impact analysis. Analysis results display on the Impact Facility screen (see [Figure 53](#)).
- 8 Save, append, and print the summary impact analysis results to a worklist.
- 9 Use the standard ISPF scroll keys to display the qualification information for each entity identified.

Figure 53 • Impact Analysis Results - Impact Facility Screen

```

File Edit View Search Generate Worklist Options Help
-----
Impact Facility                                ABBEY-AL
Command ==> _____ Scroll ==> CSR
Name: ABBEY-AL - DATE IMPACT FOR TESTING      1 of 1229
                                             >>>

Impact Base Target List

Name-----Kind-----Parent-----
'CD05B'      ENTRY      VIAAL10.ABBEY.CO
'CD05B'      ENTRY      VIAAL10.ABBEY.CO
'CD05C'      ENTRY      VIAAL10.ABBEY.CO

Impact Generation Options

Include Base Selection in results
Include Copy Detail in results

Base Impact Targets:      Application  Assignments
Targets                  Level Sunonums To  From  Alternates

```

Listing the Relationships of Impacted Items (Detailed Impact Analysis)

Alliance builds the impact analysis by layering related components. The detailed analysis shows what each layer in the summary analysis is and why the layer was applied to the member of the base selection list. With detailed analysis, retrace all paths taken to generate the summary analysis results.

Various information options control the information shown for each entity. The defaults are name, entity kind, and containing component. Change the impact generate options to modify the default display. After you have a base of summary and detailed analysis results, look up unknown components to refine the input options and rerun the impact analysis. For step-by-step instructions on modifying default impact generate options and investigating unknown components, see ["Modifying the Impact Analysis" on page 97](#).

Detailed impact analysis results display hierarchically, with each level representing an impact relationship path to a new component. When a single component is related to several entities, these paths are represented at the same offset. The first component to the left is the base selection entity. [Figure 54](#) is an example of detailed analysis results.

Figure 54 • Example of Detailed Impact Analysis Results

```
File Edit View Search Generate Worklist Options Help
-----
Command ==> _____ Impact Facility _____ Appl
Scroll ==> CSR
Name: An Impact - Description 1 of nn >>>
***** TOP OF DATA *****
(Selection One)
|
+- (Related Component)
| |
| +- (Related Component)
| |
| ..
| |
| - (Related Component)
| |
| - (Related Component)
| ..
| |
| - (Related Component)
| |
| - (Related Component)
| ..
|
- (Related Component)
..
```

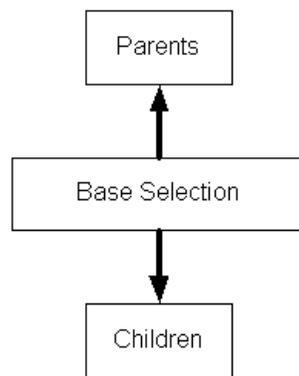
Understanding Detailed Analysis

The impact generate options determine the types of relationships assessed. The order in which the assessment is performed is constant.

To perform a relationship assessment

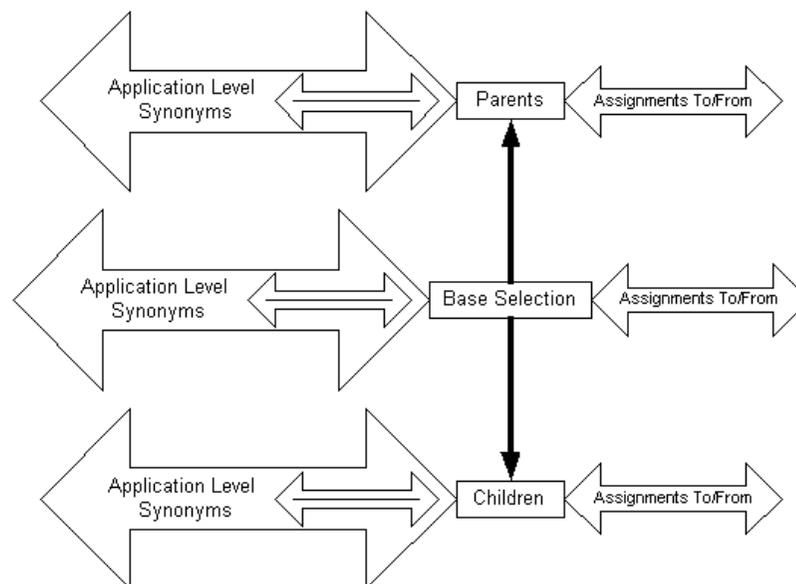
- 1 Alliance identifies components hierarchically-related to base selection entities and makes independent groups. Hierarchically-related entities are parents and children of the application components you selected (see [Figure 55](#)).

Figure 55 • Step 1 - Parents and Children of Base Selection Entities



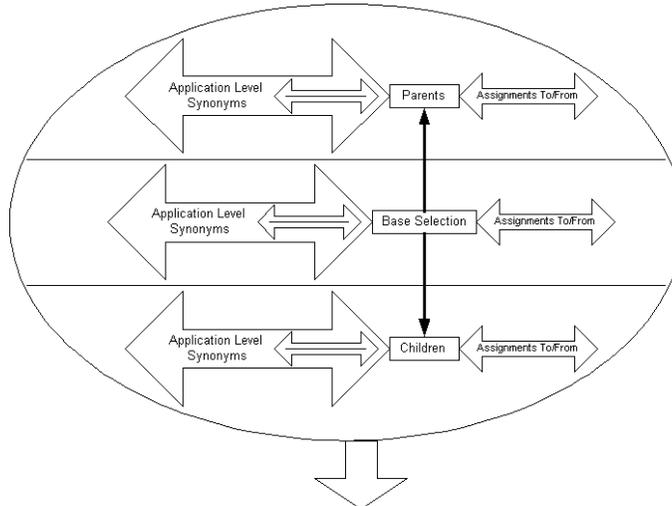
- 2 Alliance finds synonyms and local assignments (entities within the scope of the target component that have direct and indirect data movement to and from one another) for each group (see [Figure 56](#)).

Figure 56 • Step 2 - Synonyms and assignments



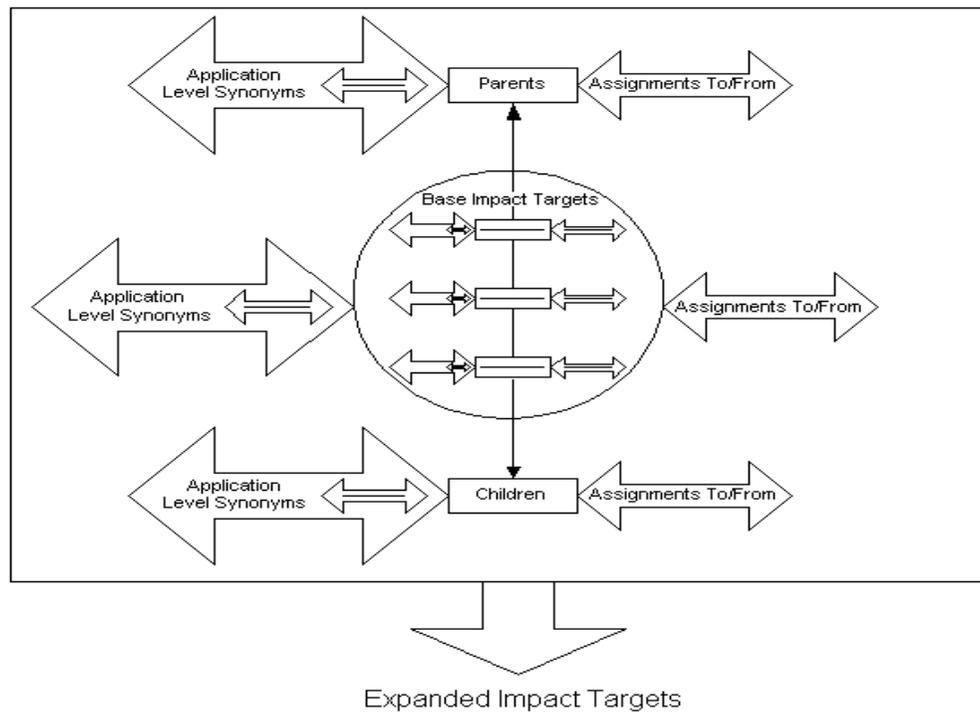
- 3 Alliance groups accumulated components into a single base impact target group (see [Figure 57](#)).

Figure 57 • Step 3 - Base Impact Targets



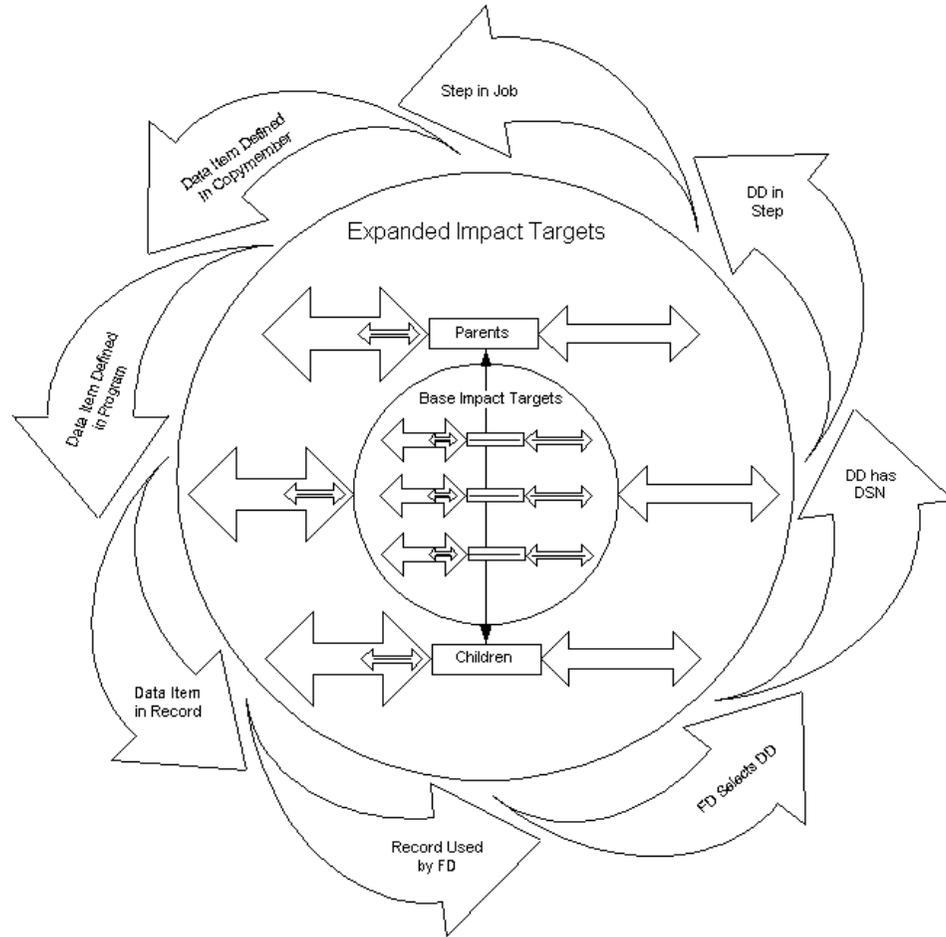
- 4 Alliance creates the expanded impact target group by identifying the parents and children and the local assignments of the base impact target group (see [Figure 58](#)).

Figure 58 • Step 4 - Expanded Impact Targets



- 5 Alliance repeats [step 3](#) and [step 4](#) until no new component is discovered (see [Figure 59](#)).

Figure 59 • Step 5 - Example of Entity Relationships



Hierarchical Relationships

Hierarchically-related components (parents and children) vary with the definition of the entity. This table lists the parents and children of specific entity kinds recognized by Alliance:

Entity Kind	Parents	Children
CICS mapset	none	CICS map, CICS map field
COBOL dataitem including rename, redefine, and 88 level items	Containing group components until level 01, 66, or 77 (i.e. up to the record level)	All within the group name

Entity Kind	Parents	Children
IDMS area	IDMS subschema, IDMS schema	IDMS logical record, IDMS record, IDMS map
IDMS logical record	none	IDMS record
IDMS schema	none	IDMS subschema, IDMS area, IDMS logical record, IDMS record, IDMS map
IDMS subschema	IDMS schema	IDMS area, IDMS logical record, IDMS record, IDMS map
IMS DynAllocMacro	none	DD, DSN
IMS segment	none	IMS segment
JCL job	none	JCL step, JCL DD
JCL INCLUDE	none	JCL step, JCL DD
JCL PROC	none	JCL step, JCL DD
JCL step	none	JCL DD
Segment	none	Segment
Senseg	none	Senseg
SQL column	The owning Table or View	none
SQL database	SQL Stogroup	SQL tablespace, SQL table, SQL view, SQL column
SQL Stogroup	none	SQL database, SQL tablespace, SQL table, SQL view, SQL column
SQL table or view	none	SQL columns within the table or view
All others	none	none

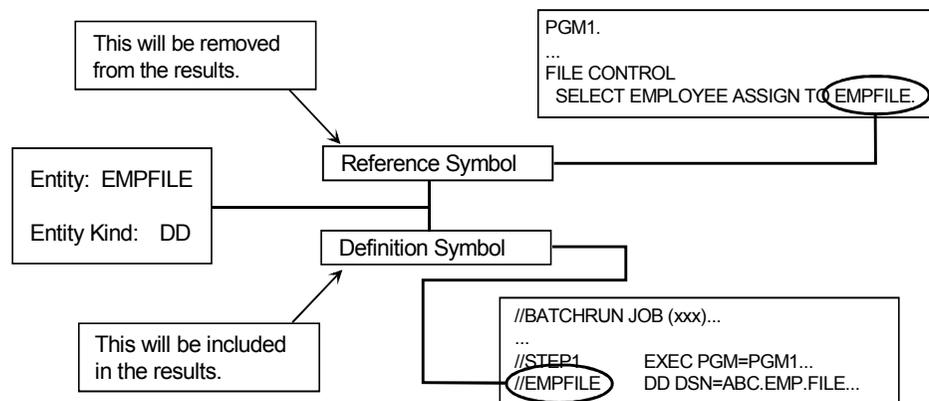
Referential Symbols

As the Application Analytical Engine (AAE) analyzes individual components within the application, it encounters references to entities that are defined elsewhere. The AAE marks each of these references to the entity to connect it to the definition of the entity, and to the other references to the entity. These references are known as referential symbols. The connecting process is the semantic linking phase of the analysis.

As the impact analysis traverses relationships (that is, parents, children, assignments, synonyms, and entity relationships) to generate results, it uses referential symbols to step from entity to entity in the relationship chains. Each referential symbol found during the impact analysis is substituted by the appropriate definition symbol and removed from the results.

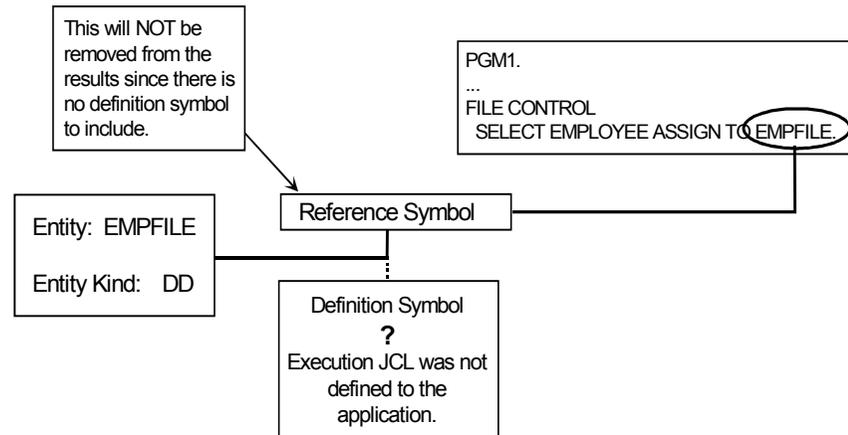
[Figure 60](#) illustrates the removal of referential symbols.

Figure 60 • Referential Symbol Removal



In cases where the definition symbol does not exist within the application, a single referential symbol is kept in the results of the impact analysis, as shown in [Figure 61](#).

Figure 61 • Referential Symbol Inclusion



Information Display Options

You can control the display and type of the information shown in detailed impact analysis results. The defaults for the display are connecting lines between related components and an initial value of 4 for the number of levels shown. These are the defaults for the information:

- Name
- Entity kind
- Containing component
- Connecting lines

Other options include:

- Containing component description
- Impact relationship
- Entity-relationship-entity information

- 3 To select a data item for analysis, type any character in the column beside the data item and press Enter.

Or

To select all entries, press PF5.

- 4 Press PF2 to view or change the impact generate options.

These options control the type and format of the information that the impact analysis displays, as well as the components of the base selection list. If you are performing the impact analysis for the first time, ASG recommends you use the defaults for these options.

Note:

For step-by-step instructions on modifying default impact generate options, see ["Specifying New Impact Generate Options" on page 108.](#)

- 5 Press PF4 to generate the detailed impact analysis. Analysis results display on the Impact Facility screen.

You can save, append, and print the detailed impact analysis results to a worklist. Use the standard ISPF scroll keys to display the qualification information for each entity identified.

[Figure 63](#) shows an example of the information displayed on the Impact Facility screen after you generate the detailed impact analysis.

Figure 63 • Example of Detailed Impact Analysis Results

```

File Edit View Search Generate Worklist Options Help
-----
Command ==> UIAINFO          Impact Facility          Appl
                               Scroll ==> CSR
Name: An Impact - Description      1 of nn
(Selection - One)                  >>>
|
| +- (Related Component A)
| |
| | +- (Related Component B)----- The cursor was placed here.
|
-----
                        Detailed Information
ENTITY KIND           : {Related Component B} {Qualification}
ENTITY KIND           : {Entity Kind for Component B}
CONTAINING COMPONENT : {Containing Name} { (Description) } {Qualification}
IMPACT RELATIONSHIP  : {Impact Relationship}
    
```

To expand or collapse a path on the Impact Facility screen

1 Select View ▶ Zoom In

Or

Select View ▶ Zoom Out.

Press Enter.

2 Choose one of these options:

- Place the cursor on the desired component and press Enter.
- Place the cursor on the desired component.
- Press PF6 to Zoom In or PF5 to Zoom Out.

To display the Detail Information pop-up on the Impact Facility screen

1 Select Facility ▶ Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.

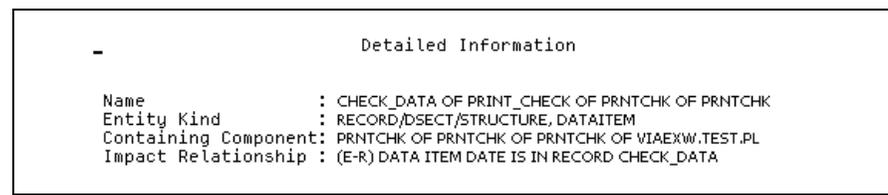
2 Select View ▶ Details and press Enter.

3 Place the cursor on the desired component and press Enter.

Or

Press PF4 to display the Detail Information pop-up (see [Figure 64](#)).

Figure 64 • Detailed Information Pop-up



4 Review the information. Use the standard ISPF scroll keys to display the qualification information for each entity.

5 Press PF3 to exit.

To view the Direct Path to an Impact result

- 1** Select View ▶ Zoom Direct Path and press Enter.
- 2** Place the cursor on the desired component and press Enter. Alliance collapses all lines except the detailed impact answers that show the path to the selected line.

6

Modifying the Impact Analysis

This chapter describes the impact generate options, gives instructions on how to generate impact options and work with impact lists, and contains these sections:

Section	Page
Changing Impact Generate Options	97
Refining the Base Selection List	110
Rerunning the Impact Analysis	113

As described in the overview, at this stage you generate the first round of impact analysis results, based upon the list of items you select for impact assessment. You can modify impact generate options, change the base selection list, and execute new impact analysis to retrace unknown components in the analysis results. Refine impact analysis parameters and eliminate unknown components to limit the impact assessment to entities germane to the change. Then, use these results to determine what actions need to be taken and whether the current flow of data between components is redirected.

Changing Impact Generate Options

Impact generate options affect how the impact analysis traverses the data flow, as well as the hierarchical relationships of the components. Using this new information, you might modify the composition of the base selection list to include application components you did not initially plan to analyze, such as application-level synonyms and local assignments.

You might also include additional hierarchical relationships to the results, depending on the nature of the application and the task at hand. Use the Impact Generate Options screen to specify parents and children of the selected targets, as well as application-level synonyms and assignment synonyms of parents and children. [Figure 65](#) shows the Impact Generate Options screen with ESW defaults marked with a slash (/).

Figure 65 • Impact Generate Options Screen

```

Command ==> Impact Generate Options
-----
Select desired Impact Generate Options.
- Include Base Selection in results
- Include Copy Detail in results

Base Impact Targets:
Targets
-----
/ Base Selection /
/ Parents of Base Selection /
- Children of Base Selection -

Application Assignments
Level Synonyms To From Alternates
-----
/ / - - -
/ / - - -
- - - - -

Expanded Impact Targets:
Targets
-----
/ Base Impact Targets /
/ Parents of Base Impact Targets /
- Children of Base Impact Targets -

Application Assignments
Level Synonyms To From Alternates
-----
/ / - - -
/ / - - -
- - - - -

PF4=Restore ASG Default Options PF5=Detailed Options
    
```

The options on the Impact Generate Options screen are available for these two groups:

- ["Step 3 - Base Impact Targets" on page 88](#)
- ["Step 4 - Expanded Impact Targets" on page 88.](#)

For a description of the impact analysis relationship assessment process, see ["Understanding Detailed Analysis" on page 87.](#)

The Impact Generate Options screen also provides access to the detailed impact analysis options that control the display of connecting lines, the number of levels displayed when the data is collapsed, and the amount of information with each entity.

For a description of the options available for detailed analysis, see ["Detailed Impact Generate Options" on page 104.](#)

For step-by-step instructions on modifying Impact Generate Options, see ["Specifying New Impact Generate Options" on page 108.](#)

Base Impact Targets Options

Base impact targets options consist of three levels: base selection, parents of base selection, and children of base selection.

Base Selection

The base selection refers to the entities chosen from the Select Impact Generation Entities pop-up (see "[Selecting Items to be Analyzed](#)" on page 82). The Base Selection option is an ESW default that cannot be deleted. These are the options available for base selection:

Option	Description
Application Level Synonyms	<p>Includes application-level synonyms of the selected entities in the analysis.</p> <p>Activates the Assignments To and Assignments From options. ESW default for Base Selection.</p>
Assignments To	<p>Includes entities that the selected entities assign value to in the analysis.</p> <p>For example, if A is the selected entity; A assigns value to B; B is included in the analysis.</p>
Assignments From	<p>Includes entities that the selected entities are assigned value from in the analysis.</p> <p>For example, if A is the selected entity; A is assigned value from B; B is included in the analysis.</p>
Assignments Alternates	<p>Includes data movement and comparisons within programs for statements other than direct moves in the analysis.</p> <p>Statements that make up this relationship are those where the data in a variable alters the value of another variable, or where one data item is compared to another.</p> <p>Local or global synonyms calculations are not affected.</p> <p>All languages and language levels supported by Alliance can use this feature.</p>

Parents of Base Selection

Parents of base selection refers to the parents of entities chosen from the Select Impact Generation Entities pop-up. The Parents of Base Selection option is an ESW default. These are the options available for the parents of base selection:

Option	Description
Application Level Synonyms	<p>Includes application-level synonyms of the parents of the selected entities in the analysis.</p> <p>Activates the Assignments To and Assignments From options.</p> <p>ESW default for Parents of Base Selection.</p>
Assignments To	<p>Includes entities that the parents of the selected entities assign value to in the analysis.</p> <p>For example, if A is the selected entity; B is the parent of A, and B assigns value to C; C is included in the analysis.</p>
Assignments From	<p>Includes entities that the parents of the selected entities are assigned value from in the analysis.</p> <p>For example, if A is the selected entity; B is the parent of A, and B is assigned value from C, C is included in the analysis.</p>
Assignments Alternates	<p>Includes data movement and comparisons within programs for statements other than direct moves in the analysis.</p> <p>Statements that make up this relationship are those where the data in a variable alters the value of another variable, or where one data item is compared to another.</p> <p>Local or global synonyms calculations are not affected.</p> <p>All languages and language levels supported by Alliance can use this feature.</p>

Children of Base Selection

Children of base selection refers to the children of entities chosen from the Select Impact Generation Entities pop-up. If you select this option, all entities that are children of the selected entities are included in the analysis. These are the options available for children of base selection:

Option	Description
Application Level Synonyms	<p>Includes application-level synonyms of the children of the selected entities in the analysis.</p> <p>Activates the Assignments To and Assignments From options.</p>
Assignments To	<p>Includes entities that the children of the selected entities assign value to in the analysis.</p> <p>For example, if A is the selected entity; B is the child of A, and B assigns value to C; C is included in the analysis.</p>
Assignments From	<p>Includes entities that the children of the selected entities are assigned value from in the analysis.</p> <p>For example, if A is the selected entity; B is the child of A, and B is assigned value from C, C is included in the analysis.</p>
Assignments Alternates	<p>Includes data movement and comparisons within programs for statements other than direct moves in the analysis.</p> <p>Statements that make up this relationship are those where the data in a variable alters the value of another variable, or where one data item is compared to another.</p> <p>Local or global synonyms calculations are not affected.</p> <p>All languages and language levels supported by Alliance can use this feature.</p>

Expanded Impact Targets Options

Expanded Impact Targets options include all entities selected for analysis as a result of the options set under Base Impact Target options. All base impact targets go into any expanded impact targets option. Expanded impact targets options consist of three levels: base impact targets, parents of impact targets, and children of impact targets.

Base Impact Targets

Base impact targets are the entities selected for analysis as a result of the options set for the base selections. The Base Impact Targets option is an ESW default that cannot be deleted. These are the options available for base impact targets:

Option	Description
Application Level Synonyms	<p>Includes application-level synonyms of the base impact targets in the analysis.</p> <p>Activates the Assignments To and Assignments From options.</p> <p>ESW defaults for Base Impact Targets.</p>
Assignments To	<p>Includes entities that the base impact targets assign value to in the analysis.</p>
Assignments From	<p>Includes entities that the base impact targets are assigned value from in the analysis.</p>
Assignments Alternates	<p>Includes data movement and comparisons within programs for statements other than direct moves in the analysis.</p> <p>Statements that make up this relationship are those where the data in a variable alters the value of another variable, or where one data item is compared to another.</p> <p>Local or global synonyms calculations are not affected.</p> <p>All languages and language levels supported by Alliance can use this feature.</p>

Parents of Base Impact Targets

The term parents of base impact targets refers to the parents of all entities selected for analysis under that option. These are the options available for parents of base impact targets:

Option	Description
Application Level Synonyms	Includes application-level synonyms of the parents of the base impact targets in the analysis. Activates the Assignments To and Assignments From options. ESW default for Parents of Base Impact Targets.
Assignments To	Includes entities that the parents of the base impact targets assign value to in the analysis.
Assignments From	Includes entities that the parents of the base impact targets are assigned value from in the analysis.
Assignments Alternates	Includes data movement and comparisons within programs for statements other than direct moves in the analysis. Statements that make up this relationship are those where the data in a variable alters the value of another variable, or where one data item is compared to another. Local or global synonyms calculations are not affected. All languages and language levels supported by Alliance can use this feature.

Children of Base Impact Targets

The term children of base impact targets refers to the children of all entities selected for analysis under that option. These are the options available for children of base impact targets:

Option	Description
Application Level Synonyms	Includes application-level synonyms of the children of the base impact targets in the analysis. Activates the Assignments To and Assignments From options.
Assignments To	Includes entities that the children of the base impact targets assign value to in the analysis.

Option	Description
Assignments From	Includes entities that the children of the base impact targets are assigned value from in the analysis.
Assignments Alternates	Includes data movement and comparisons within programs for statements other than direct moves in the analysis. Statements that make up this relationship are those where the data in a variable alters the value of another variable, or where one data item is compared to another. Local or global synonyms calculations are not affected. All languages and language levels supported by Alliance can use this feature.

Detailed Impact Generate Options

The Detailed Impact Generate Options pop-up contains options that control the display of connecting lines, the number of levels initially displayed on the Impact Facility screen for detailed analysis, and the amount of information shown with each entity. [Figure 66](#) shows the Detailed Impact Generate Options pop-up with ESW defaults marked with a slash (/).

Figure 66 • Detailed Impact Generate Options Pop-up

```

Detailed Impact Generate Options
Command ==> _____ Scroll ==> CSR
Display Options                               1 of 6
-----
/ Draw Connecting Lines
Initial Number of Levels to Display  4
  _ Display All Levels

Information Shown With Each Entity
-----
/ Name
/ Entity Kind
/ Containing Component
_ Containing Component Description
/ Impact Relationship
_ Entity-Relationship-Entity Information
***** BOTTOM OF DATA *****

```

Display Options

These are the options available to control the display of information for detailed impact analysis:

Option	Description
Draw Connecting Lines	Controls the display of the connecting lines drawn from component to related component. This is the ESW default for display options.
Initial Number of Levels to Display	Controls the number of levels shown upon initial display of the detailed impact analysis results. The number of levels specified must be between 1 and 2,147,483,647 (MAXINT). The default value is 4. This option is ignored if Display All Levels is selected.
Display All Levels	Forces all levels of impact relationship to be drawn on initial display. The ESW default is to disable this option. Note: The summary impact results do not necessarily match the detailed impact results if you do not turn on the Display All Levels detailed impact option.

Information Display Options

The options available to control the information displayed on the Impact Facility screen for each entity in detailed impact analysis results are listed in this table.

Note:

To view data for disabled information options in the Impact Facility screen, display the Detail Information pop-up for each component.

These are the default information options:

Option	Description															
Name	Includes the name of the component. ESW default is enabled.															
Entity Kind	Includes the entity kind of the component as recognized by Alliance. A component can be defined as more than one entity kind. For example, (DATAITEM, RECORD). ESW default is enabled.															
Containing Component	Includes the name of the component in the application where the entity was found. ESW default is enabled.															
Containing Component Description	Specifies the type of the containing component. These are the possible types: <table border="0" style="margin-left: 20px;"> <tr> <td>ASM</td> <td>CICS-BMS</td> <td>CICS-CDS</td> </tr> <tr> <td>CICS-FCT</td> <td>CICS-PCT</td> <td>CICS-PPT</td> </tr> <tr> <td>COBOL</td> <td>IMS-MFS</td> <td>IMS-DBD</td> </tr> <tr> <td>IMS-DFSMDA</td> <td>IMS-PSB</td> <td>JCL</td> </tr> <tr> <td>LOADMOD</td> <td>PL/I</td> <td></td> </tr> </table> <p>Note: _____ The value of UNKNOWN displays when the containing component is a dataset and its language type cannot be isolated.</p>	ASM	CICS-BMS	CICS-CDS	CICS-FCT	CICS-PCT	CICS-PPT	COBOL	IMS-MFS	IMS-DBD	IMS-DFSMDA	IMS-PSB	JCL	LOADMOD	PL/I	
ASM	CICS-BMS	CICS-CDS														
CICS-FCT	CICS-PCT	CICS-PPT														
COBOL	IMS-MFS	IMS-DBD														
IMS-DFSMDA	IMS-PSB	JCL														
LOADMOD	PL/I															
Impact Relationship	Includes the nature of the impact relationship. Possible relationships are listed on page 107 . ESW default is disabled.															
Entity-Relationship-Entity Information	Includes the specific relationship when the components are related because of an entity-relationship-entity connection. ESW default is disabled.															

These are the possible impact relationships displayed when the Impact Relationship option is enabled:

Relationship Name	Description
CALL TO {PGM}	Related items are used as actual and formal parameters in a CALL statement.
{I/O ACTIVITY} [TO/FROM] THE SAME FILE LOCATION	Related items are used in an I/O activity to the same file location. The nature of the activity displays when possible. I/O ACTIVITY: READ, CREATE, UPDATE, DELETE.
{I/O ACTIVITY} [TO/FROM] THE SAME DATA BASE COMPONENT	The related items are used in an I/O activity to the same database component. The nature of the activity displays when possible. I/O ACTIVITY: READ, CREATE, UPDATE, DELETE.
DEFINITION	Related items refer to the same information by definition, for example A REDEFINES B.
{INSTRUCTION} [TO/FROM] {RELATED COMPONENT}	Related items are linked by an instructed data flow. For example on a program level, if data was moved from one item to the other.
These are the instructions for causing data movement:	
COBOL	ACCEPT, MOVE, MOVE CORRESPONDING
I/O OPERATIONS	READ, REWRITE, WRITE, SORT, MERGE, RETURN, RELEASE
IDMS	ACCEPT, END TRANSACTION WRITE, FIND, OBTAIN, GET
SQL	FETCH, UPDATE
IMS	INSERT, GET, REPLACE
CICS	SEND MAP, RECEIVE MAP, WRITE, READ, READPREV, READNEXT, REWRITE
ENTITY RELATIONSHIP	Related items are found in an entity-relationship-entity connection.
[PARENT/CHILD]	Related items are connected because a parent or child has an impact relationship.

Figure 67 shows the format of the data shown on the Impact Facility screen. If you disable any of the information options, the information is removed from the pattern and the remaining information stays in the same sequence. Use the Detail Information pop-up for each component to view the data.

Figure 67 • Information Options Format

```

{name} {(entity kind)} {OF containing component{(description)}} {- impact
relationship{(entity-relationship-entity)}}

      Name          Containing          Impact
      |             Component           Relationship
+- PAYFILE(FD) OF PGM2(COBOL) - ENTITY RELATIONSHIP (RECORD USED BY FD)
      |             |                   |
      Entity Kind   Containing          Entity-Relationship-Entity
                   Component
                   Description
    
```

Specifying New Impact Generate Options

To change impact generate options

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.
- 2 Select Generate ► Summary and press Enter.

Note: _____

If you have already generated impact analysis results without an open worklist, a warning displays telling you to open or create a worklist, or confirm that you do not want to save the current data.

- 3 To display the Impact Generate Options pop-up, press PF2 on the Select Impact Generation Entities pop-up.

Choose one of these actions:

- a To select an option in the Targets column, type any character to the left of the option.
- b Select Application Level Synonyms options by typing any character beneath that column, next to the appropriate target level.
- c Select Assignments by typing any character in the To, From, or Alternates column.

For information about base and expanded impact targets options, see "[Base Impact Targets Options](#)" on page 99 and "[Expanded Impact Targets Options](#)" on page 102.

- 4 When you have finished selecting options, press PF3 to return to the Select Impact Generation Entities pop-up.

To reset default impact generate options

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.
- 2 Select Generate ► Summary and press Enter.
- 3 On the Select Impact Generation Entities pop-up, press PF2 to display the Impact Generate Options pop-up.
- 4 Press PF4 to restore the defaults.
- 5 Press PF3 to return to the Select Impact Generation Entities pop-up.

To change detailed impact generate options

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.
- 2 Select Generate ► Detailed and press Enter.
- 3 On the Select Impact Generation Entities pop-up, press PF2 to display the Impact Generate Options pop-up.
- 4 Press PF5 to display the Detailed Impact Generate Options pop-up.
- 5 Specify the desired options and press PF3 to return to the Impact Generate Options pop-up.
- 6 Press PF3 again to return to the Select Impact Generation Entities pop-up.

To reset default detailed impact generate options

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.
- 2 Select Generate ► Detailed and press Enter.
- 3 On the Select Impact Generation Entities pop-up, press PF2 to display the Impact Generate Options pop-up.

- 4 Press PF5 to display the Detailed Impact Generate Options pop-up.
- 5 Press PF4 to restore the defaults.
- 6 Press PF3 to return to the Impact Generate Options pop-up.
- 7 Press PF3 again to return to the Select Impact Generation Entities pop-up.

Refining the Base Selection List

Modify the composition of the base selection list to recover data not found in previous analyses. Depending on the situation, you can perform these tasks:

- Revise the impact list from where the base selection list is derived.
- Rebuild the base selection list you specified in the Select Impact Generation Entities pop-up.
- Revise impact generate options that determine the base and expanded impact targets.

Revising an Impact List

To modify an existing impact list (open)

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.
- 2 Choose Edit ► Impact list and press Enter. The Impact Entity List pop-up displays with the list of previously defined impact list entities.

To modify an existing impact list (closed)

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.
- 2 Select File ► Open impact list and press Enter.
- 3 In the Open Impact List pop-up, type any character in the selection column to the left of the impact list you want to modify and press Enter.
- 4 Press PF4 (Edit). The Impact Entity List pop-up displays with the list of previously defined impact list entities.

To add components to an open impact list

- 1 From the Impact Entity List pop-up, press PF4 (Add). The Impact Entities Selection Criteria pop-up displays.
- 2 Type the full name of an entity kind.

Or

Type a pattern for selection in the Text field.

Note: _____

If you enter a pattern and want to limit the entity kinds selected, specify one or more entity kinds by typing any character in the selection column to the left of the entity name.

- 3 Press Enter. The Add Impact List Entity pop-up displays ([Figure 48 on page 78](#)) showing the list of entity names matching the text entered on the Impact Entities Selection Criteria pop-up. The list is scrollable.
- 4 Type any character in the selection column beside the desired entity name. Select as many entities as needed.
- 5 Press PF10 to add the entries and return to the Impact Entity List pop-up.
- 6 Press PF3 to return to the Open Impact List pop-up.
- 7 Press PF3 again to return to the Impact Facility screen.

To remove components from an open impact list

- 1 On the Impact Entity List pop-up, select the entities you want to remove by typing any character in the selection column to the left of the entity name.
- 2 Press PF5 (Delete).

Caution! The deletion takes place instantly, without a confirmation message allowing you to cancel.

- 3 Press PF3 to return to the Impact Facility screen.

To save the updated impact list

- 1 Select File ► Save impact list from the Impact Facility screen and press Enter. The message `[impact_list_name] HAS BEEN SAVED TO IMPACT DEFINITION` displays confirming that the original impact list updated with the new information.

Or

Select File ► Save impact list and press Enter to preserve the original impact list and store the impact list under a new name.

The Save Impact As pop-up displays with the name and description of the original impact list.

- 2 Type the name and description of the new impact list in the Name and Description fields and press Enter.

The Impact Facility screen returns with the message `[impact_list_name] HAS BEEN SAVED TO IMPACT DEFINITION` confirming the creation of the new impact list.

Rebuilding the Base Selection List

To rebuild the base selection list

- 1 Select Facility ► Impact from the Alliance Primary screen and press Enter. The Impact Facility screen displays.
- 2 Select Generate ► Summary and press Enter. The Summary Impact - Select Impact Generation Entities pop-up displays showing all entities in the open application that match in the open impact list. The type of each entity is listed along with qualification information that provides details on the occurrence location.
- 3 To specify the new entity names to be analyzed, type any character in the selection column beside the data items you want to select.

Or

To select all entities, press PF5.

- 4 Proceed with the analysis.

For step-by-step instructions on generating summary and detailed analysis results, see ["Locating Impacted Components and Relationships \(Summary Impact Analysis\)" on page 83](#) and ["Listing the Relationships of Impacted Items \(Detailed Impact Analysis\)" on page 85](#).

Rerunning the Impact Analysis

As described in the overview, the final stage in the refining process is rerunning the impact analysis. Iterate the impact analysis until you get the results you need to proceed with a change request implementation.

To rerun the impact analysis

- 1 Review the results of previous impact analysis.
- 2 Investigate the paths traversed by components related to the change.
- 3 Refine analysis parameters by modifying the impact list, base selection list, and impact generate options.

For instructions on generating summary and detailed analysis results, see "[Locating Impacted Components and Relationships \(Summary Impact Analysis\)](#)" on page 83 and "[Listing the Relationships of Impacted Items \(Detailed Impact Analysis\)](#)" on page 85.

Appendix A

Valid Entities and Relationships

Alliance recognizes these entities by multiple names:

- FD, FILEVAR, or DCB
- PROGRAM, PROCEDURE, or FUNCTION
- RECORD, DSECT, or STRUCTURE

These are the entities recognized by Alliance (listed in alphabetical order):

AREA	LOADMODULE
CATALOG	LOGICALRECORD
CICS-TRANSACTION	MAP
COLUMN	MAPFIELD
COPYMEMBER	MAPSET
CSD	MEMBER
CSECT	PCB
CURSOR	PCT
DATABASE	PPT
DATAITEM	PROC
DBD	PROCEDURE/PROGRAM/FUNCTION
DCB/FD/FILEVAR	PROGRAM/PROCEDURE/FUNCTION
DD	PSB
DSECT/RECORD/STRUCTURE	RECORD/DSECT/STRUCTURE
DSN	SCHEMA
ENTRY	SEGFIELD
FCT	SEGMENT

FCTCSD	SENFIELD
FD/FILEVAR/DCB	SENSEG
FILEVAR/FD/DCB	SQL
FUNCTION/PROGRAM/PROCEDURE	STEP
IDMS	STOGROUP
IDMS MAP	STRUCTURE/RECORD/DSECT
IDMS RECORD	SUBSCHEMA
IMS-TRANSACTION	TABLE
IMS-FORMAT	TABLESPACE
IMSDYNALLOCMACRO	VIEW
JOB	
LIBRARY	

These are the relationships recognized by Alliance (listed in alphabetical order):

ALTERNATE	LINKED-BY
CALLED-BY	LINKS
CALLS	LOCALSYNONYM
DEFINED-IN	REFERENCED-BY
EXECUTED-BY	REFERENCES
EXECUTES	SELECTED-BY
HAS	SELECTS
HOMONYM	SYNONYM
IN	USED BY
INCLUDES	USES
IS	

These are the entity-relationships recognized by Alliance (listed in alphabetical order):

Entity	Definitions of Valid Entities in Alliance Query Usage		
AREA	HAS	IDMSRECORD	Identifies records located in logical areas of IDMS schemas.
AREA	IN	PROGRAM PROCEDURE FUNCTION	Identifies logical areas of IDMS schemas in programs.
AREA	IN	SUBSCHEMA	Identifies local areas in IDMS databases that are defined within subschemas.
CATALOG	USED BY	JOB	Identifies catalogs used by jobs in an application.
CATALOG	USED BY	STEP	Identifies catalogs used by Job Steps in an application.
CICS-TRANSACTION	EXECUTED-BY	PROGRAM PROCEDURE FUNCTION	Identifies all CICS transactions executed by programs within an application.
CICS-TRANSACTION	EXECUTES	PROGRAM PROCEDURE FUNCTION	Identifies programs executed by CICS transactions within an application.
CICS-TRANSACTION	IN	CSD	Identifies CICS transactions in CSD files.
CICS-TRANSACTION	IN	PCT	Identifies CICS transactions within PCTs.
COLUMN	IN	PROGRAM PROCEDURE FUNCTION	Identifies relational database fields as referenced in a program.
COLUMN	IN	TABLE	Identifies fields within relational database tables.
COLUMN	IN	VIEW	Identifies fields within relational database views.
COPYMEMBER	HAS	DATAITEM	Identifies copymembers that are directly included by the copymember. Same applies to JCL INCLUDES.
COPYMEMBER	IN	COPYMEMBER	Identifies copymembers that directly include copymembers.

Entity		Definitions of Valid Entities in Alliance Query Usage	
COPYMEMBER	IN	MEMBER	Identifies members that contain a copybook. Same applies to JCL INCLUDES.
COPYMEMBER	IN	PROGRAM PROCEDURE FUNCTION	Identifies copymembers referenced by programs defined to an application. Identifies JCL INCLUDES referenced by JCL defined to an application.
COPYMEMBER	INCLUDES	COPYMEMBER	Identifies copymembers referencing data items defined in a copymember. Same applies to JCL INCLUDES.
COPYMEMBER	USEDDBY	COPYMEMBER	Identifies copymembers that directly and indirectly include copymembers (nested copies). Same applies to JCL INCLUDES.
COPYMEMBER	USEDDBY	MEMBER	Lists copymembers that are included in program members. Also lists JCL INCLUDES that are included in the JCL.
COPYMEMBER	USES	COPYMEMBER	Identifies copymembers directly and indirectly included by copymembers (nested copies). Same applies to JCL INCLUDES.
CSD	HAS	CICS-TRANSACTION	Identifies CICS transactions defined in CSD files.
CSD	HAS	DD	Identifies DDs defined in CSD files.
CSECT	IN	LOADMODULE	Identifies control sections in a load module.
CSECT	IS	PROGRAM PROCEDURE FUNCTION	Identifies control sections in a program. Assigns CSECTS to a Program.
CURSOR	IN	DATABASE	Identifies SQL object pointers that can be used to run through a set of records that are returned by a query.
CURSOR	IN	PROGRAM PROCEDURE FUNCTION	Identifies SQL cursors referenced within a program.

Appendix A - Valid Entities and Relationships

Entity	Definitions of Valid Entities in Alliance Query Usage		
DATABASE	HAS	CURSOR	Identifies SQL cursors that are defined to a database.
DATABASE	HAS	TABLESPACE	Indicates the available tablespaces in a DB2 database.
DATABASE	IN	PROGRAM PROCEDURE FUNCTION	Indicates the program where DB2 databases are referenced.
DATABASE	IN	STOGROUP	Indicates the storage GROUPS where a DB2 database resides.
DATAITEM	DEFINED-IN	COPYMEMBER	Identifies data items defined by each copymember defined to the application.
DATAITEM	DEFINED-IN	PROGRAM PROCEDURE FUNCTION	Identifies data items defined by each program in an application.
DATAITEM	HOMONYM	DATAITEM	Lists homonyms of data items in an application.
DATAITEM	IN	RECORD DSECT STRUCTURE	Lists records containing data items.
DATAITEM	LOCALSYNONYM	DATAITEM	Lists local synonyms of data items within a program.
DATAITEM	SYNONYM	DATAITEM	Lists global synonyms of data items in an application.
DBD	HAS	DD	Identifies the DD names to which the database definitions are allocated.
DBD	HAS	SEGMENT	Identifies the segments (records) in an IMS database that are defined by the database definition.
DBD	IN	MEMBER	Identifies members containing an IMS DBDGEN.
DBD	IN	PCB	Identifies database definitions included in a PCB (Program Communication Block).

Entity	Definitions of Valid Entities in Alliance Query Usage		
DCB (See FD)			
DD	HAS	DSN	Identifies DSNs referenced by DD statement.
DD	IN	CSD	Identifies DD statements in CSD files.
DD	IN	DBD	Identifies DD statements in a database description.
DD	IN	FCT	Identifies DD statements in a CICS FCT.
DD	IN	FCTCSD	Identifies the location where DDs are defined by a CICS CSD entry.
DD	IN	STEP	Identifies DD statements in each step of a JCL job stream.
DD	SELECTED-BY	FD FILEVAR DCB	Identifies DD names referenced by an FD statement within a COBOL program.
DD	USED BY	PROGRAM PROCEDURE FUNCTION	Lists DD names used in a program via CICS statements.
DSECT (See RECORD)			
DSN	HAS	MEMBER	Lists the members within a partitioned dataset.
DSN	IN	DD	Specifies dataset names, as specified in the DD statement in the JCL.
ENTRY	CALLED-BY	PROGRAM PROCEDURE FUNCTION	Identifies entry names called by a program.
ENTRY	IN	PROGRAM PROCEDURE FUNCTION	Identifies a program's entry point names.
FCT	HAS	DD	Identifies DDs defined in FCTS.
FCT	IN	MEMBER	Lists members that contain an FCT.

Appendix A - Valid Entities and Relationships

Entity	Definitions of Valid Entities in Alliance Query Usage		
FCTCSD	HAS	DD	Identifies where CICS DDs were defined in CSD files.
FD FILEVAR DCB	IN	PROGRAM PROCEDURE FUNCTION	Identifies the file description statements in a COBOL source program.
FD FILEVAR DCB	SELECTS	DD	Lists the DD names referenced by an FD statement in a COBOL program.
FD FILEVAR DCB	USES	RECORD DSECT STRUCTURE	Lists record names referenced by FD statements.
FILEVAR (See FD)			
FUNCTION (See PROGRAM)			
IDMS	IN	PROGRAM PROCEDURE FUNCTION	Lists IDMS entities referenced in a program.
IDMSMAP	HAS	IDMSRECORD	Shows the relationship between IDMS screen maps and records.
IDMSMAP	IN	PROGRAM PROCEDURE FUNCTION	Identifies programs that reference IDMS maps.
IDMSRECORD	IN	AREA	Lists IDMS records contained in logical areas.
IDMSRECORD	IN	IDMSMAP	Shows the relationship between IDMS screen records and maps.
IDMSRECORD	IN	LOGICALRECORD	Lists IDMS subschema records contained in IDMS logical records.
IDMSRECORD	IN	PROGRAM PROCEDURE FUNCTION	Identifies programs that reference IDMS records.
IMSDYNALLOCMACRO	IN	MEMBER	Lists PDS members that contain IMS dynamic allocation macros.

Entity	Definitions of Valid Entities in Alliance Query Usage		
IMS-FORMAT	EXECUTED-BY	IMS-FORMAT	Lists the formats(s) executed by a format.
IMS-FORMAT	EXECUTED-BY	PROGRAM PROCEDURE FUNCTION	Identifies programs that invoke IMS formats.
IMS-FORMAT	EXECUTES	IMS-TRANSACTION	Lists the formats that execute an IMS transaction.
IMS-FORMAT	EXECUTES	IMS-FORMAT	Lists formats that execute IMS formats.
IMS-FORMAT	IN	MEMBER	Lists PDS members that contain IMS format definitions.
IMS-TRANSACTION	EXECUTED-BY	IMS-FORMAT	Lists the IMS transactions executed by IMS formats.
IMS-TRANSACTION	EXECUTED-BY	PROGRAM PROCEDURE FUNCTION	Identifies programs that invoke IMS transactions.
IMS-TRANSACTION	EXECUTES	LOADMODULE	Lists the load modules executed by IMS transactions.
JOB	HAS	STEP	Lists all JCL job names in the application along with their associated step names.
JOB	IN	MEMBER	Lists JCL job names and their containing member names.
JOB	USES	CATALOG	Lists the JCL job names in the application and the names of the catalogs they use, if any.
JOB	USES	LIBRARY	Lists the JCL job names in the application and the JOBLIBS used.
LIBRARY	HAS	MEMBER	Lists the partitioned dataset names in the application and their member names.
LIBRARY	USEDDBY	JOB	Lists the libraries used as JOBLIBS and the jobs referencing them.
LIBRARY	USEDDBY	STEP	Lists the libraries used as STEPLIBS and the steps referencing them.

Appendix A - Valid Entities and Relationships

Entity	Definitions of Valid Entities in Alliance Query Usage		
LOADMODULE	EXECUTED-BY	IMS-TRANSACTION	Lists the load module names in the application that are executed by IMS transactions.
LOADMODULE	EXECUTED-BY	STEP	Lists the load module names in the application and the JCL step names that execute them.
LOADMODULE	HAS	CSECT	Lists the load modules in an application and their CSECT names.
LOADMODULE	IN	MEMBER	Lists the load modules in an application and their member names in the partition datasets where they are located.
LOGICALRECORD	HAS	IDMSRECORD	Lists IDMS logical records in an application and the IDMS records they contain.
LOGICALRECORD	IN	PROGRAM PROCEDURE FUNCTION	Lists IDMS logical records referenced in a program.
MAP	HAS	MAPFIELD	Lists CICS maps in an application and their mapfield names.
MAP	IN	MAPSET	Lists CICS maps and the mapsets that contain them.
MAP	REFERENCES	RECORD DSECT STRUCTURE	Establishes the relationship of CICS maps to COBOL records.
MAP	USED BY	PROGRAM PROCEDURE FUNCTION	Lists CICS maps in an application, and the programs where they are used.
MAPFIELD	IN	MAP	Lists CICS mapfields in an application and the map names that contains them.
MAPSET	HAS	MAP	Lists mapsets in an application and the maps contained by the mapsets.
MAPSET	IN	MEMBER	Lists CICS mapsets in an application and their member names in a partitioned dataset.

Entity	Definitions of Valid Entities in Alliance Query Usage		
MAPSET	USEDDBY	PROGRAM PROCEDURE FUNCTION	Lists CICS mapsets in an application and the programs that reference them.
MEMBER	HAS	COPYMEMBER	Lists the member names of partitioned datasets that contain copymembers. Same applies to JCL INCLUDES.
MEMBER	HAS	DBD	Lists the member names of partitioned datasets that contain DBDS.
MEMBER	HAS	FCT	Lists the member names of partitioned datasets that contain FCTS.
MEMBER	HAS	FORMAT	Identifies member names of partitioned datasets that contain MFS definitions for the formats.
MEMBER	HAS	IMSDYNALLOCMACRO	Lists the member names of partitioned datasets that contain dynamic allocation macros.
MEMBER	HAS	JOB	Lists the member names of partitioned datasets that contain JCL job streams.
MEMBER	HAS	LOADMODULE	Lists the member names of partitioned datasets that contain load modules.
MEMBER	HAS	MAPSET	Lists the member names of partitioned datasets that contain mapsets.
MEMBER	HAS	PCT	Lists the member names of partitioned datasets that contain PCTS.
MEMBER	HAS	PPT	Lists the member names of partitioned datasets that contain PPTS.
MEMBER	HAS	PROC	Lists cataloged procedure names and their member names in a partitioned dataset.
MEMBER	HAS	PROGRAM PROCEDURE FUNCTION	Lists the program names in an application and their member name in a partitioned dataset.

Appendix A - Valid Entities and Relationships

Entity	Definitions of Valid Entities in Alliance Query Usage		
MEMBER	HAS	PSB	Lists the member names of partitioned datasets that contain PSBS.
MEMBER	IN	DSN	Lists partition dataset names and their member names.
MEMBER	IN	LIBRARY	Lists the partitioned dataset names in the application and their member names.
MEMBER	USES	COPYMEMBER	Lists the member names of partitioned datasets that contain included copymembers. Same applies to JCL INCLUDES.
PCB	HAS	SENSEG	Lists PCBs that have sensitive segments as defined by SENSEG statements.
PCB	IN	PSB	Lists IMS program communication blocks and the PSB where they are located.
PCB	REFERENCES	DBD	Lists IMS PCBs and the DBDS they reference.
PCT	HAS	CICS-TRANSACTION	Lists CICS transactions defined in PCTS.
PCT	IN	MEMBER	Lists PCTS and their member names in partitioned datasets.
PPT	IN	MEMBER	Lists PPTS and their member names in partitioned datasets.
PROC	EXECUTED-BY	STEP	Lists cataloged procedures and the step names that execute them.
PROC	HAS	STEP	Lists step names in catalog procedures.
PROC	IN	MEMBER	Lists PROC names and their member names in partitioned datasets.
PROCEDURE (See PROGRAM)			
PROGRAM PROCEDURE FUNCTION	CALLS	ENTRY	Lists programs in the application that issue calls to load module entry points.

Entity	Definitions of Valid Entities in Alliance Query Usage		
PROGRAM PROCEDURE FUNCTION	EXECUTED-BY	CICS-TRANSACTION	Lists programs in the application that are executed by CICS transactions.
PROGRAM PROCEDURE FUNCTION	EXECUTES	CICS-TRANSACTION	Lists programs in the application that execute CICS transactions.
PROGRAM PROCEDURE FUNCTION	EXECUTES	IMS-FORMAT	Lists programs in the application that execute IMS formats.
PROGRAM PROCEDURE FUNCTION	EXECUTES	IMS-TRANSACTION	Lists programs in the application that execute IMS transactions.
PROGRAM PROCEDURE FUNCTION	HAS	AREA	Lists IDMS areas referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	COLUMN	Lists DB2 columns referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	CURSOR	Lists DB2 cursors referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	DATABASE	Lists DB2 databases referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	DATAITEM	Lists programs in the application and data items defined in that program.
PROGRAM PROCEDURE FUNCTION	HAS	ENTRY	Lists programs in the application and any ENTRY point names.
PROGRAM PROCEDURE FUNCTION	HAS	FD FILEVAR DCB	Lists programs in the application that have FD statements.
PROGRAM PROCEDURE FUNCTION	HAS	IDMS	Lists programs in the application that reference IDMS entities.

Appendix A - Valid Entities and Relationships

Entity	Definitions of Valid Entities in Alliance Query Usage		
PROGRAM PROCEDURE FUNCTION	HAS	IDMSMAP	Lists IDMS maps referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	IDMSRECORD	Lists IDMS records referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	LOGICALRECORD	Lists IDMS logical records referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	RECORD DSECT STRUCTURE	Lists programs in the application and record names referenced in FD statements.
PROGRAM PROCEDURE FUNCTION	HAS	SCHEMA	Lists IDMS schemas referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	SQL	Lists programs in the application that reference SQL entities.
PROGRAM PROCEDURE FUNCTION	HAS	STOGROUP	Lists DB2 storage groups referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	SUBSCHEMA	Lists IDMS subschemas referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	TABLE	Lists DB2 tables referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	TABLESPACE	Lists DB2 tablespaces referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	HAS	VIEW	Lists DB2 views referenced by programs within an application.
PROGRAM PROCEDURE FUNCTION	IN	MEMBER	Lists the program names in the application and their member names in a partitioned dataset.

Entity	Definitions of Valid Entities in Alliance Query Usage		
PROGRAM PROCEDURE FUNCTION	INCLUDES	COPYMEMBER	Lists programs in the application that have copy or include statements.
PROGRAM PROCEDURE FUNCTION	IS	CSECT	Establishes linkage between CSECTs and programs.
PROGRAM PROCEDURE FUNCTION	LINKED-BY	PROGRAM PROCEDURE FUNCTION	Lists programs executed by CICS LINKs or XCTL statements.
PROGRAM PROCEDURE FUNCTION	LINKS	PROGRAM PROCEDURE FUNCTION	Lists programs that execute other programs via CICS LINKs or XCTLs.
PROGRAM PROCEDURE FUNCTION	USES	DD	Lists programs in the application and the DD names referenced via CICS statements.
PROGRAM PROCEDURE FUNCTION	USES	MAP	Lists programs in the application that use CICS maps.
PROGRAM PROCEDURE FUNCTION	USES	MAPSET	Lists programs in the application that reference CICS mapsets.
PROGRAM PROCEDURE FUNCTION	USES	PSB	Lists programs in the application that use IMS PSBS.
PROGRAM PROCEDURE FUNCTION	USES	SEGMENT	Lists programs in that application that use segments in an IMS database.
PSB	HAS	PCB	Lists PSBs in the application and PCBs contained by the PSBS.
PSB	IN	MEMBER	Lists PSBs in the application and the member names they are stored under in partitioned datasets.
PSB	USED BY	PROGRAM PROCEDURE FUNCTION	Lists PSBs in the application and the programs that use them.

Appendix A - Valid Entities and Relationships

Entity	Definitions of Valid Entities in Alliance Query Usage		
RECORD DSECT STRUCTURE	HAS	DATAITEM	Lists the record names in an application and their field names.
RECORD DSECT STRUCTURE	IN	PROGRAM PROCEDURE FUNCTION	Lists the record names in a program.
RECORD DSECT STRUCTURE	REFERENCED-BY	MAP	Lists the record names defined to an application and the map names that reference them.
RECORD DSECT STRUCTURE	USED BY	FD FILEVAR DCB	Lists the record names defined to an application and the FDs that use them.
SCHEMA	HAS	SUBSCHEMA	Lists the IDMS schemas defined to an application and their associated subschemas.
SCHEMA	IN	PROGRAM PROCEDURE FUNCTION	Lists programs that reference entities defined in IDMS schemas.
SEGFIELD	IN	SEGMENT	Lists the IMS segment fields defined to an application and the segments where they are defined.
SEGMENT	HAS	SEGFIELD	Lists the IMS segments defined to an application and the fields defined within them.
SEGMENT	IN	DBD	Lists IMS segments and the parental DBD associated with them.
SEGMENT	IN	SEGMENT	Lists the segments in an IMS database and their parent segments.
SEGMENT	USED BY	PROGRAM PROCEDURE FUNCTION	Lists IMS segments used by programs within an application.
SEGMENT	USES	SEGMENT	List the segments in an IMS database and their child segments.

Entity	Definitions of Valid Entities in Alliance Query Usage		
SENFIELD	IN	SENSEG	Lists sensitive fields (defined by an IMS SENFIELD statement) in a sensitive segment (defined by an IMS SENSEG statement).
SENSEG	HAS	SENFIELD	Lists IMS sensitive segments and any fields within the segment defined with a SENFIELD statement.
SENSEG	IN	PCB	Lists sensitive segments and the PCB names where they appear.
SENSEG	IN	SENSEG	Lists the SENSEGs in a PCB and their parental SENSEGs.
SENSEG	USES	SENSEG	Lists the SENSEGs in a PCB and their children SENSEGs.
SQL	IN	PROGRAM PROCEDURE FUNCTION	Lists SQL entities and any referencing programs.
STEP	EXECUTES	LOADMODULE	Lists JCL step names and the load module names that they execute.
STEP	EXECUTES	PROC	Lists JCL job steps that execute PROCS.
STEP	HAS	DD	Lists steps and DDS referenced by them.
STEP	IN	JOB	Lists step names and the job names they are associated with.
STEP	IN	PROC	Lists step names and the PROCS where they reside.
STEP	USES	CATALOG	Lists JCL step names and the catalogs that are referenced in the step.
STEP	USES	LIBRARY	Lists JCL step names and the libraries that are referenced as STEPLIBs.
STOGROUP	HAS	DATABASE	Lists the storage groups and the DB2 databases they contain.

Entity	Definitions of Valid Entities in Alliance Query Usage		
STOGROUP	IN	PROGRAM PROCEDURE FUNCTION	Lists storage group names and the programs where they are defined.
STRUCTURE (See RECORD)			
SUBSCHEMA	HAS	AREA	Lists IDMS subschema names and the logical areas in an IDMS database defined within the subschemas.
SUBSCHEMA	HAS	LOGICALRECORD	Lists IDMS subschema names and the logical record names that they are associated with.
SUBSCHEMA	IN	PROGRAM PROCEDURE FUNCTION	Lists programs that reference entities defined in IDMS subschemas.
SUBSCHEMA	IN	SCHEMA	Lists IDMS subschema names in the schemas with which they are associated.
TABLE	HAS	COLUMN	Lists DB2 relational database table definitions and the columns defined within them.
TABLE	IN	PROGRAM PROCEDURE FUNCTION	Lists programs that reference relational database tables.
TABLE	IN	TABLESPACE	Lists DB2 tables and the tablespaces that contains them.
TABLESPACE	HAS	TABLE	Lists DB2 tablespaces and the tables defined within.
TABLESPACE	HAS	VIEW	Lists DB2 tablespaces and the views defined within.
TABLESPACE	IN	DATABASE	Lists DB2 tablespaces and the databases where they are defined.
TABLESPACE	IN	PROGRAM PROCEDURE FUNCTION	Lists programs that reference entities defined within DB2 tablespaces.
VIEW	HAS	COLUMN	Lists any defined DB2 views and the columns they control.

Entity	Definitions of Valid Entities in Alliance Query Usage		
VIEW	IN	PROGRAM PROCEDURE FUNCTION	Lists any defined DB2 views and the programs that use them.
VIEW	IN	TABLESPACE	Lists DB2 views and the tablespaces where they are defined.

Appendix B

Impact Analysis Entity Relationships

The Alliance Impact Facility traverses these recognized relationships searching for additional impact answers for each of these starting entity kinds:

Starting Entity Kind	Entity Relationships		
AREA	AREA	IN	PROGRAM
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	AREA	HAS	IDMSRECORD
	IDMSRECORD	IN	PROGRAM
	IDMSRECORD	IN	IDMSMAP
	IDMSMAP	IN	PROGRAM
CATALOG	CATALOG	USED BY	JOB
	CATALOG	USED BY	STEP
	STEP	IN	JOB
	STEP	IN	PROC
	PROC	EXECUTED-BY	STEP
CICS TRANSACTION	CICSTRANSACTION	EXECUTES	PROGRAM
	CICSTRANSACTION	EXECUTED-BY	PROGRAM
	CICSTRANSACTION	IN	CSD
	CICSTRANSACTION	IN	PCT

Starting Entity Kind	Entity Relationships		
COLUMN	COLUMN	IN	PROGRAM
	COLUMN	IN	VIEW
	VIEW	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	IDMSRECORD	IN	IDMSMAP
	IDMSRECORD	IN	LOGICAL RECORD
	LOGICAL RECORD	IN	PROGRAM
	IDMSRECORD	IN	PROGRAM
	DATAITEM	IN	RECORD
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	DBD

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	DD	IN	FCT
	DD	IN	CSD
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	SENSEG	USES	SENSEG
	SENSEG	IN	SENSEG
	SENSEG	HAS	SENFIELD
	SENSEG	IN	PCB
	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
COPYMEMBER	COPYMEMBER	IN	PROGRAM
	COPYMEMBER	USED BY	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	HAS	DATAITEM
	DATAITEM	IN	RECORD
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	IN	DBD

Starting Entity Kind	Entity Relationships		
	DD	IN	FCT
	DD	IN	CSD
	DD	HAS	DSN
	DSN	HAS	MEMBER
CSECT	CSECT	IN	LOADMODULE
	LOADMODULE	IN	MEMBER
	MEMBER	IN	LIBRARY
	LOADMODULE	EXECUTED-BY	IMSTRANSACTION
	LOADMODULE	EXECUTED-BY	STEP
	STEP	IN	JOB
	STEP	IN	PROC
	PROC	EXECUTED-BY	STEP
	CSECT	IS	PROGRAM
	PROGRAM	LINKED-BY	PROGRAM
	PROGRAM	HAS	ENTRY
	ENTRY	CALLED-BY	PROGRAM
	PROGRAM	EXECUTED-BY	CICSTRANSACTION
CURSOR	CURSOR	IN	DATABASE
	DATABASE	IN	STOGROUP
	CURSOR	IN	PROGRAM
DATABASE	DATABASE	IN	PROGRAM
	DATABASE	IN	STOGROUP
	DATABASE	HAS	CURSOR
	CURSOR	IN	PROGRAM
	DATABASE	HAS	TABLESPACE

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	TABLESPACE	IN	PROGRAM
	TABLESPACE	HAS	VIEW
	VIEW	IN	PROGRAM
	VIEW	HAS	COLUMN
	TABLESPACE	HAS	TABLE
	TABLE	IN	PROGRAM
	TABLE	HAS	COLUMN
	COLUMN	IN	PROGRAM
DATAITEM	DATAITEM	DEFINED-IN	PROGRAM
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	DATAITEM	IN	RECORD
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	DD	IN	STEP
	STEP	IN	JOB

Starting Entity Kind	Entity Relationships		
	STEP	IN	PROC
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	DBD	HAS	DD
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	HAS	SEGFIELD
	SENSEG	IN	PCB
	SENSEG	HAS	SENFIELD
	SENSEG	IN	SENSEG
	SENSEG	USES	SENSEG
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	IDMSRECORD	IN	IDMSMAP
	IDSMRECORD	IN	LOGICAL RECORD
	LOGICAL RECORD	IN	PROGRAM
	IDMSRECORD	IN	PROGRAM
	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	COLUMN	IN	VIEW
	VIEW	IN	TABLESPACE

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
DBD	DBD	HAS	DD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	STEP
	STEP	IN	JOB
	STEP	IN	PROC
	PROC	EXECUTED-BY	STEP
	DBD	HAS	SEGMENT
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	USED BY	PROGRAM
	DBD	IN	PCB
	PCB	HAS	SENSEG
	SENSEG	USES	SENSEG
	SENSEG	IN	SENSEG
	SENSEG	HAS	SENFIELD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
	DBD	IN	MEMBER
MEMBER	IN	LIBRARY	
DD	DD	SELECTED-BY	FD
	FD	IN	PROGRAM
	FD	USES	RECORD
	RECORD	HAS	DATAITEM

Starting Entity Kind	Entity Relationships		
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	DD	USED BY	PROGRAM
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	STEP
	STEP	IN	JOB
	STEP	IN	PROC
	PROC	EXECUTED-BY	STEP
DSN	DSN	HAS	MEMBER
	DSN	IN	DD
	DD	USED BY	PROGRAM
	DD	IN	DBD
	DBD	HAS	SEGMENT
	SEGMENT	USED BY	PROGRAM
	DD	IN	FCT
	DD	IN	CSD
	DD	SELECTED-BY	FD
	FD	IN	PROGRAM
	FD	USES	RECORD
	RECORD	HAS	DATAITEM

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	DD	IN	STEP
	STEP	IN	JOB
	STEP	IN	PROC
	PROC	EXECUTED-BY	STEP
ENTRY	ENTRY	IN	PROGRAM
	PROGRAM	EXECUTED-BY	CICSTRANSACTION
	PROGRAM	HAS	ENTRY
	ENTRY	CALLED-BY	PROGRAM
	PROGRAM	USES	DD
	PROGRAM	HAS	FD
	FD	SELECTS	DD
	DD	IN	STEP
	DD	IN	CSD
	DD	IN	FCT
	DD	HAS	DSN
	DSN	HAS	MEMBER
	PROGRAM	HAS	AREA
	PROGRAM	HAS	LOGICAL RECORD
	PROGRAM	HAS	IDMSMAP
	PROGRAM	HAS	IDMSRECORD
	PROGRAM	HAS	SCHEMA
	PROGRAM	HAS	SUBSCHEMA

Starting Entity Kind	Entity Relationships		
	PROGRAM	HAS	RECORD
	RECORD	HAS	DATAITEM
	RECORD	REFERENCED-BY	MAP
	PROGRAM	HAS	COLUMN
	PROGRAM	HAS	CURSOR
	PROGRAM	HAS	DATABASE
	PROGRAM	HAS	STOGROUP
	PROGRAM	HAS	TABLE
	PROGRAM	HAS	TABLESPACE
	PROGRAM	HAS	VIEW
	PROGRAM	IN	MEMBER
	MEMBER	IN	LIBRARY
	PROGRAM	INCLUDES	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	PROGRAM	IS	CSECT
	CSECT	IN	LOADMODULE
	LOADMODULE	EXECUTED-BY	STEP
	STEP	IN	JOB
	STEP	IN	PROC
	PROC	EXECUTED-BY	STEP
	LOADMODULE	EXECUTED-BY	IMSTRANSAC TION
	PROGRAM	LINKED-BY	PROGRAM
	PROGRAM	USES	MAP
	MAP	IN	MAPSET
	PROGRAM	USES	MAPSET

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	PROGRAM	USES	PSB
	PROGRAM	USES	SEGMENT
	SEGMENT	IN	DBD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	HAS	SEGFIELD
	SENSEG	IN	SENSEG
	SENSEG	USES	SENSEG
	SENSEG	HAS	SENFIELD
	SENSEG	IN	PCB
FCT	FCT	IN	MEMBER
	MEMBER	IN	LIBRARY
	FCT	HAS	DD
	DD	USED BY	PROGRAM
	DD	SELECTED-BY	FD
	FD	IN	PROGRAM
	FD	USES	RECORD
	RECORD	HAS	DATAITEM
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	DD	USED BY	PROGRAM
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA

Starting Entity Kind	Entity Relationships	
IDMSRECORD	IN	IDMSMAP
IDMSRECORD	IN	LOGICAL RECORD
LOGICAL RECORD	IN	PROGRAM
IDMSRECORD	IN	PROGRAM
COLUMN	IN	PROGRAM
TABLE	IN	PROGRAM
COLUMN	IN	TABLE
TABLE	IN	TABLESPACE
TABLESPACE	IN	DATABASE
DATABASE	IN	STOGROUP
COLUMN	IN	VIEW
VIEW	IN	TABLESPACE
DATAITEM	DEFINED-IN	PROGRAM
DATAITEM	DEFINED-IN	COPYMEMBER
COPYMEMBER	USES	COPYMEMBER
COPYMEMBER	IN	PROGRAM
DATAITEM	IN	RECORD
RECORD	REFERENCED-BY	MAP
MAP	IN	MAPSET
MAP	HAS	MAPFIELD
RECORD	USED BY	FD
FD	SELECTS	DD
DD	HAS	DSN
DSN	HAS	MEMBER
DD	IN	DBD

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	DD	IN	FCT
	DD	IN	CSD
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	SENSEG	USES	SENSEG
	SENSEG	IN	SENSEG
	SENSEG	HAS	SENFIELD
	SENSEG	IN	PCB
	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
FCTCSD	FCTCSC	HAS	DD
	FCT	HAS	DD
	CSD	HAS	DD
	DD	USED BY	PROGRAM
	DD	SELECTED-BY	FD
	FD	IN	PROGRAM
	FD	USES	RECORD
	RECORD	HAS	DATAITEM
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM

Starting Entity Kind	Entity Relationships	
IDMSRECORD	IN	AREA
AREA	IN	SUBSCHEMA
SUBSCHEMA	IN	SCHEMA
IDMSRECORD	IN	IDMSMAP
IDMSRECORD	IN	LOGICALRECORD
LOGICAL RECORD	IN	PROGRAM
IDMSRECORD	IN	PROGRAM
TABLE	IN	PROGRAM
COLUMN	IN	TABLE
TABLE	IN	TABLESPACE
TABLESPACE	IN	DATABASE
DATABASE	IN	STOGROUP
COLUMN	IN	VIEW
VIEW	IN	TABLESPACE
DATAITEM	DEFINED-IN	PROGRAM
DATAITEM	DEFINED-IN	COPYMEMBER
COPYMEMBER	USES	COPYMEMBER
COPYMEMBER	IN	PROGRAM
DATAITEM	IN	RECORD
RECORD	REFERENCED-BY	MAP
MAP	IN	MAPSET
MAP	HAS	MAPFIELD
RECORD	USED BY	FD
FD	SELECTS	DD
DD	HAS	DSN

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	DSN	HAS	MEMBER
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	SENSEG	USES	SENSEG
	SENSEG	IN	SENSEG
	SENSEG	HAS	SENFIELD
	SENSEG	IN	PCB
	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
FD	FD	SELECTS	DD
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	DD	IN	STEP
	STEP	IN	JOB
	STEP	IN	PROC
	DD	HAS	DSN
	DSN	HAS	MEMBER

Starting Entity Kind	Entity Relationships		
	FD	IN	PROGRAM
	FD	USES	RECORD
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	HAS	DATAITEM
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
FORMAT	FORMAT	EXECUTED-BY	FORMAT
	FORMAT	EXECUTED-BY	PROGRAM
	PROGRAM	EXECUTED-BY	CICSTRANSACTION
	PROGRAM	HAS	ENTRY
	ENTRY	CALLED-BY	PROGRAM
	PROGRAM	IS	CSECT
	CSECT	IN	LOADMODULE
	LOADMODULE	EXECUTED-BY	IMSTRANSACTION
	IMSTRANSACTION	EXECUTED-BY	PROGRAM
	IMSTRANSACTION	EXECUTED-BY	FORMAT
	PROGRAM	LINKED-BY	PROGRAM
IDMS	IDMSMAP	HAS	IDMSRECORD
	MAP	REFERENCES	RECORD
	RECORD	HAS	DATAITEM
	IDMS	IN	PROGRAM
	IDMSMAP	IN	PROGRAM

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	IDMSRECORD	IN	PROGRAM
	LOGICAL RECORD	IN	PROGRAM
	AREA	IN	PROGRAM
	SCHEMA	IN	PROGRAM
	SUBSCHEMA	IN	PROGRAM
	IDMSRECORD	IN	LOGICAL RECORD
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	DATAITEM	DEFINED-IN	PROGRAM
	DATAITEM	IN	RECORD
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	SEGMENT	USED BY	PROGRAM

Starting Entity Kind	Entity Relationships		
	SEGMENT	IN	DBD
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	HAS	SEGFIELD
	SENSEG	IN	PCB
	SENSEG	HAS	SENFIELD
	SENSEG	IN	SENSEG
	SENSEG	USES	SENSEG
	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	COLUMN	IN	VIEW
	VIEW	IN	TABLESPACE
IDMS MAP	IDMSMAP	IN	PROGRAM
	IDMSMAP	HAS	IDMSRECORD
	IDMSRECORD	IN	LOGICAL RECORD
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	COLUMN	IN	PROGRAM
	TABLE	IN	PROGRAM
	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships	
TABLESPACE	IN	DATABASE
DATABASE	IN	STOGROUP
COLUMN	IN	VIEW
VIEW	IN	TABLESPACE
DATAITEM	DEFINED-IN	PROGRAM
DATAITEM	DEFINED-IN	COPYMEMBER
COPYMEMBER	USES	COPYMEMBER
COPYMEMBER	IN	PROGRAM
DATAITEM	IN	RECORD
RECORD	REFERENCED-BY	MAP
MAP	IN	MAPSET
MAP	HAS	MAPFIELD
RECORD	USED BY	FD
FD	SELECTS	DD
DD	HAS	DSN
DSN	HAS	MEMBER
DD	IN	DBD
DD	IN	FCT
DD	IN	CSD
SEGMENT	HAS	SEGFIELD
SEGMENT	IN	SEGMENT
SEGMENT	USES	SEGMENT
SEGMENT	USED BY	PROGRAM
SEGMENT	IN	DBD
SENSESEG	USES	SENSESEG

Starting Entity Kind	Entity Relationships		
	SENSEG	IN	SENSEG
	SENSEG	HAS	SENFIELD
	SENSEG	IN	PCB
	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
IDMS RECORD	IDMSRECORD	IN	PROGRAM
	IDMSRECORD	IN	IDMSMAP
	IDMSMAP	IN	PROGRAM
	IDMSRECORD	IN	LOGICAL RECORD
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	COLUMN	IN	PROGRAM
	TABLE	IN	PROGRAM
	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	COLUMN	IN	VIEW
	VIEW	IN	TABLESPACE
	DATAITEM	DEFINED-IN	PROGRAM
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	DATAITEM	IN	RECORD
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	SENSESEG	USES	SENSESEG
	SENSESEG	IN	SENSESEG
	SENSESEG	HAS	SENFIELD
	SENSESEG	IN	PCB
	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
IMS DYNALLOCMACRO	IMS DYNALLOCMACRO	IN	MEMBER
	MEMBER	IN	LIBRARY

Starting Entity Kind	Entity Relationships		
IMS TRANSACTION	IMSTRANSACTION	EXECUTED-BY	PROGRAM
	PROGRAM	EXECUTED-BY	CICSTRANSACTION
	PROGRAM	HAS	ENTRY
	ENTRY	CALLED-BY	PROGRAM
	PROGRAM	IS	CSECT
	CSECT	IN	LOADMODULE
	LOADMODULE	EXECUTED-BY	STEP
	STEP	IN	JOB
	STEP	IN	PROC
	PROC	EXECUTED-BY	STEP
	LOADMODULE	EXECUTED-BY	IMSTRANSACTION
	PROGRAM	LINKED-BY	PROGRAM
	IMSTRANSACTION	EXECUTED-BY	FORMAT
	FORMAT	EXECUTED-BY	FORMAT
	FORMAT	EXECUTED-BY	PROGRAM
	IMSTRANSACTION	EXECUTES	LOADMODULE
	LOADMODULE	HAS	CSECT
CSECT	IS	PROGRAM	
JCL PROC	PROC	EXECUTED-BY	STEP
	STEP	IN	JOB
	JOB	USES	LIBRARY
	JOB	USES	CATALOG
	PROC	HAS	STEP
	STEP	USES	CATALOG
	STEP	USES	LIBRARY

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	STEP	HAS	DD
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	STEP	EXECUTES	LOADMODULE
	LOADMODULE	HAS	CSECT
	CSECT	IS	PROGRAM
	PROC	IN	MEMBER
	MEMBER	IN	LIBRARY
JOB	JOB	HAS	STEP
	STEP	USES	LIBRARY
	STEP	USES	CATALOG
	STEP	HAS	DD
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	STEP	EXECUTES	LOADMODULE
	LOADMODULE	HAS	CSECT
	CSECT	IS	PROGRAM
	STEP	EXECUTES	PROC
	JOB	USES	LIBRARY

Starting Entity Kind	Entity Relationships		
	JOB	USES	CATALOG
	JOB	IN	MEMBER
	MEMBER	IN	LIBRARY
LIBRARY	LIBRARY	USED BY	STEP
	STEP	IN	JOB
	STEP	IN	PROC
	PROC	EXECUTED-BY	STEP
	LIBRARY	USED BY	JOB
	LIBRARY	HAS	MEMBER
	MEMBER	HAS	LOADMODULE
	MEMBER	HAS	PROGRAM
	MEMBER	HAS	JOB
	MEMBER	HAS	PROC
	MEMBER	HAS	PSB
	MEMBER	HAS	DBD
	MEMBER	HAS	PCT
	MEMBER	HAS	PPT
	MEMBER	HAS	FCT
	MEMBER	HAS	MAPSET
	MEMBER	HAS	IMSDYNALLOCMACRO
	MEMBER	HAS	FORMAT
	MEMBER	HAS	COPYMEMBER
LOADMODULE	LOADMODULE	HAS	CSECT
	CSECT	IS	PROGRAM
	PROGRAM	EXECUTED-BY	CICSTRANSACTION

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	PROGRAM	LINKED-BY	PROGRAM
	PROGRAM	HAS	ENTRY
	ENTRY	CALLED-BY	PROGRAM
	LOADMODULE	EXECUTED-BY	IMSTRANSACTION
	LOADMODULE	EXECUTED-BY	STEP
	STEP	IN	JOB
	STEP	IN	PROC
	PROC	EXECUTED-BY	STEP
	LOADMODULE	IN	MEMBER
	MEMBER	IN	LIBRARY
LOGICAL RECORD	LOGICALRECORD	IN	PROGRAM
	LOGICALRECORD	HAS	IDMSRECORD
	IDMSRECORD	IN	PROGRAM
	IDMSRECORD	IN	IDMSMAP
	IDMSMAP	IN	PROGRAM
	IDMSRECORD	IN	LOGICALRECORD
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	COLUMN	IN	PROGRAM
	TABLE	IN	PROGRAM
	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP

Starting Entity Kind	Entity Relationships		
	COLUMN	IN	VIEW
	VIEW	IN	TABLESPACE
	DATAITEM	DEFINED-IN	PROGRAM
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	DATAITEM	IN	RECORD
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	SENSESEG	USES	SENSESEG
	SENSESEG	IN	SENSESEG
	SENSESEG	HAS	SENFIELD

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	SENSEG	IN	PCB
	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
MAP (CICS)	MAP	HAS	MAPFIELD
	MAP	IN	MAPSET
	MAPSET	USED BY	PROGRAM
	MAP	REFERENCES	RECORD
	RECORD	HAS	DATAITEM
	DATAITEM	DEFINED-IN	PROGRAM
	MAP	USED BY	PROGRAM
	IDMSMAP	HAS	IDMSRECORD
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	IDMSRECORD	IN	LOGICALRECORD
	IDMSMAP	IN	PROGRAM
MAPFIELD (CICS)	MAPFIELD	IN	MAP
	MAP	IN	MAPSET
	MAPSET	USED BY	PROGRAM
	MAP	REFERENCES	RECORD
	RECORD	HAS	DATAITEM
	DATAITEM	DEFINED-IN	PROGRAM
	MAP	USED BY	PROGRAM
	IDMSMAP	HAS	IDMSRECORD

Starting Entity Kind	Entity Relationships		
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	IDMSRECORD	IN	LOGICAL RECORD
	IDMSMAP	IN	PROGRAM
MAPSET	MAPSET	USED BY	PROGRAM
	MAPSET	HAS	MAP
	MAP	HAS	MAPFIELD
	MAP	IN	MAPSET
	MAP	REFERENCES	RECORD
	RECORD	HAS	DATAITEM
	DATAITEM	DEFINED-IN	PROGRAM
	MAP	USED BY	PROGRAM
MEMBER	MEMBER	HAS	DBD
	MEMBER	HAS	FCT
	MEMBER	HAS	FORMAT
	MEMBER	HAS	JOB
	MEMBER	HAS	LOADMODULE
	MEMBER	HAS	MAPSET
	MEMBER	HAS	IMSDYNALLOCMACRO
	MEMBER	HAS	PCT
	MEMBER	HAS	PROGRAM
	MEMBER	HAS	PPT
	MEMBER	HAS	PROC
	MEMBER	HAS	PSB

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	MEMBER	HAS	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	COPYMEMBER	USED BY	COPYMEMBER
	COPYMEMBER	HAS	DATAITEM
	DATAITEM	IN	RECORD
	MEMBER	IN	LIBRARY
	MEMBER	IN	DSN
PCB	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
	PCB	HAS	SENSESEG
	SENSESEG	USES	SENSESEG
	SENSESEG	IN	SENSESEG
	SENSESEG	USES	SENSESEG
	SENSESEG	HAS	SENFIELD
	SEGMENT	HAS	SENFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
PCT	PCT	IN	MEMBER
	MEMBER	IN	LIBRARY
	PCT	HAS	CICSTRANSACTION
	CICSTRANSACTION	EXECUTES	PROGRAM

Starting Entity Kind	Entity Relationships		
PPT	PPT	IN	MEMBER
	MEMBER	IN	LIBRARY
PROGRAM	PROGRAM	EXECUTED-BY	CICSTRANSACTION
	PROGRAM	HAS	ENTRY
	ENTRY	CALLED-BY	PROGRAM
	PROGRAM	USES	DD
	PROGRAM	HAS	FD
	FD	SELECTS	DD
	DD	IN	STEP
	DD	IN	CSD
	DD	IN	FCT
	DD	HAS	DSN
	DSN	HAS	MEMBER
	PROGRAM	HAS	AREA
	PROGRAM	HAS	LOGICALRECORD
	PROGRAM	HAS	ISMSMAP
	PROGRAM	HAS	IDMSRECORD
	PROGRAM	HAS	SCHEMA
	PROGRAM	HAS	SUBSCHEMA
	PROGRAM	HAS	RECORD
	RECORD	HAS	DATAITEM
	RECORD	REFERENCED-BY	MAP
PROGRAM	HAS	COLUMN	
PROGRAM	HAS	CURSOR	
PROGRAM	HAS	DATABASE	

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships	
PROGRAM	HAS	STOGROUP
PROGRAM	HAS	TABLE
PROGRAM	HAS	TABLESPACE
PROGRAM	HAS	VIEW
MEMBER	IN	LIBRARY
PROGRAM	INCLUDES	COPYMEMBER
COPYMEMBER	USES	COPYMEMBER
PROGRAM	IS	CSECT
CSECT	IN	LOADMODULE
LOADMODULE	EXECUTED-BY	STEP
STEP	IN	JOB
STEP	IN	PROC
PROC	EXECUTED-BY	STEP
LOADMODULE	EXECUTED-BY	IMSTRANSACTION
PROGRAM	LINKED-BY	PROGRAM
PROGRAM	USES	MAP
MAP	IN	MAPSET
PROGRAM	USES	MAPSET
PROGRAM	USES	PSB
PROGRAM	USES	SEGMENT
SEGMENT	IN	DBD
SEGMENT	IN	SEGMENT
SEGMENT	USES	SEGMENT
SEGMENT	HAS	SEGFIELD
SENSESEG	IN	SENSESEG

Starting Entity Kind	Entity Relationships		
	SENSEG	USES	SENSEG
	SENSEG	HAS	SEGFIELD
	SENSEG	IN	PCB
PSB	PSB	USEDDBY	PROGRAM
	PSB	HAS	PCB
	PCB	REFERENCES	DBD
	PCB	HAS	SENSEG
	SENSEG	IN	SENSEG
	SENSEG	USES	SENSEG
	SENSEG	HAS	SEGFIELD
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	USEDDBY	PROGRAM
	SEGMENT	IN	DBD
	PSB	IN	MEMBER
	MEMBER	IN	LIBRARY
RECORD	RECORD	HAS	DATAITEM
	DATAITEM	DEFINED-IN	PROGRAM
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	HAS	SEGFIELD
	SENSEG	IN	PCB
	SENSEG	HAS	SENFIELD
	SENSEG	IN	SENSEG
	SENSEG	USES	SENSEG
	IDMSRECORD	IN	PROGRAM
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	IDMSRECORD	IN	IDMSMAP
	IDMSRECORD	IN	LOGICALRECORD
	LOGICALRECORD	IN	PROGRAM
	COLUMN	IN	PROGRAM

Starting Entity Kind	Entity Relationships		
	TABLE	IN	PROGRAM
	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	COLUMN	IN	VIEW
	VIEW	IN	TABLESPACE
SCHEMA	SCHEMA	IN	PROGRAM
	SCHEMA	HAS	SUBSCHEMA
	SUBSCHEMA	IN	PROGRAM
	SUBSCHEMA	HAS	LOGICALRECORD
	LOGICALRECORD	IN	PROGRAM
	LOGICALRECORD	HAS	IDMSRECORD
	SUBSCHEMA	HAS	AREA
	AREA	IN	PROGRAM
	AREA	HAS	IDMSRECORD
	IDMSRECORD	IN	PROGRAM
	IDMSRECORD	IN	IDMSMPA
	IDMSMAP	IN	PROGRAM
SEGMENT	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	DBD	HAS	DD

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	SENSEG	USES	SENSEG
	SENSEG	IN	SENSEG
	SENSEG	HAS	SENFIELD
	SENSEG	IN	PCB
	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	IDMSRECORD	IN	IDMSMAP
	IDMSRECORD	IN	LOGICALRECORD
	LOGICALRECORD	IN	PROGRAM
	IDMSRECORD	IN	PROGRAM
	COLUMN	IN	PROGRAM
	TABLE	IN	PROGRAM
	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	COLUMN	IN	VIEW
	VIEW	IN	TABLESPACE
	DATAITEM	DEFINED-IN	PROGRAM
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER

Starting Entity Kind	Entity Relationships		
	COPYMEMBER	IN	PROGRAM
	DATAITEM	IN	RECORD
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	DBD
	DD	IN	FCT
SEGFIELD	DD	IN	CSD
	SEGFIELD	IN	SEGMENT
	SENSEG	USES	SENSEG
	SENSEG	IN	SENSEG
	SENSEG	IN	PCB
	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
	SEGMENT	IN	SEGMENT
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	IDMSRECORD	IN	IDMSMAP
	IDMSRECORD	IN	LOGICALRECORD
	LOGICALRECORD	IN	PROGRAM
	IDMSRECORD	IN	PROGRAM
	COLUMN	IN	PROGRAM
	TABLE	IN	PROGRAM
	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	COLUMN	IN	VIEW
	VIEW	IN	TABLESPACE
	DATAITEM	DEFINED-IN	PROGRAM
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	DATAITEM	IN	RECORD
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	DBD

Starting Entity Kind	Entity Relationships		
	DD	IN	FCT
	DD	IN	CSD
SENFIELD	SENFIELD	IN	SENSEG
	SENFIELD	USES	SENSEG
	SENSEG	IN	SENSEG
	SENSEG	IN	PCB
	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	IDMSRECORD	IN	IDMSMAP
	IDMSRECORD	IN	LOGICALRECORD
	LOGICALRECORD	IN	PROGRAM
	IDMSRECORD	IN	PROGRAM
	COLUMN	IN	PROGRAM
	TABLE	IN	PROGRAM
	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	DATABASE	IN	STOGROUP
	COLUMN	IN	VIEW
	VIEW	IN	TABLESPACE
	DATAITEM	DEFINED-IN	PROGRAM
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	DATAITEM	IN	RECORD
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
SENSEG	SENSEG	USES	SENSEG
	SENSEG	IN	SENSEG
	SENSEG	HAS	SENFIELD
	SENSEG	IN	PCB
	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM

Starting Entity Kind	Entity Relationships		
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	IDMSRECORD	IN	IDMSMAP
	IDMSRECORD	IN	LOGICALRECORD
	LOGICALRECORD	IN	PROGRAM
	IDMSRECORD	IN	PROGRAM
	COLUMN	IN	PROGRAM
	TABLE	IN	PROGRAM
	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	COLUMN	IN	VIEW
	VIEW	IN	TABLESPACE
	DATAITEM	DEFINED-IN	PROGRAM
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	DATAITEM	IN	RECORD

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	RECORD	REFERENCED-BY	MAP
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
SQL	COLUMN	IN	TABLE
	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	COLUMN	IN	VIEW
	VIEW	IN	TABLESPACE
	SQL	IN	PROGRAM
	COLUMN	IN	PROGRAM
	TABLE	IN	PROGRAM
	VIEW	IN	PROGRAM
	TABLESPACE	IN	PROGRAM
	DATABASE	IN	PROGRAM
	DATAITEM	DEFINED-IN	PROGRAM
	DATAITEM	IN	RECORD
	RECORD	REFERENCED-BY	MAP

Starting Entity Kind	Entity Relationships		
	MAP	IN	MAPSET
	MAP	HAS	MAPFIELD
	RECORD	USED BY	FD
	FD	SELECTS	DD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	DATAITEM	DEFINED-IN	COPYMEMBER
	COPYMEMBER	USES	COPYMEMBER
	COPYMEMBER	IN	PROGRAM
	SEGMENT	USED BY	PROGRAM
	SEGMENT	IN	DBD
	SEGMENT	HAS	SEGFIELD
	SEGMENT	IN	SEGMENT
	SEGMENT	USES	SEGMENT
	SEGMENT	HAS	SEGFIELD
	SENSEG	IN	PCB
	SENSEG	HAS	SENFIELD
	SENSEG	IN	SENSEG
	SENSEG	USES	SENSEG
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships		
	IDMSRECORD	IN	IDMSMAP
	IDMSRECORD	IN	LOGICALRECORD
	LOGICALRECORD	IN	PROGRAM
	IDMSRECORD	IN	PROGRAM
STEP	STEP	IN	JOB
	STEP	IN	PROC
	STEP	EXECUTES	PROC
	STEP	USES	LIBRARY
	STEP	USES	CATALOG
	STEP	HAS	DD
	DD	IN	DBD
	DD	IN	FCT
	DD	IN	CSD
	DD	HAS	DSN
	DSN	HAS	MEMBER
	STEP	EXECUTES	LOADMODULE
	LOADMODULE	HAS	CSECT
	CSECT	IS	PROGRAM
STOGROUP	STOGROUP	IN	PROGRAM
	STOGROUP	HAS	DATABASE
	DATABASE	IN	PROGRAM
	DATABASE	HAS	CURSOR
	CURSOR	IN	PROGRAM
	DATABASE	HAS	TABLESPACE
	TABLESPACE	IN	PROGRAM

Starting Entity Kind	Entity Relationships		
	TABLESPACE	HAS	VIEW
	VIEW	IN	PROGRAM
	VIEW	HAS	COLUMN
	TABLESPACE	HAS	TABLE
	TABLE	IN	PROGRAM
	TABLE	HAS	COLUMN
	COLUMN	IN	PROGRAM
SUBSCHEMA	SUBSCHEMA	IN	PROGRAM
	SUBSCHEMA	IN	SCHEMA
	SUBSCHEMA	HAS	LOGICALRECORD
	LOGICALRECORD	IN	PROGRAM
	LOGICALRECORD	HAS	IDMSRECORD
	SUBSCHEMA	HAS	AREA
	AREA	IN	PROGRAM
	AREA	HAS	IDMSRECORD
	IDMSRECORD	IN	PROGRAM
	IDMSRECORD	IN	IDMSMAP
	IDMSMAP	IN	PROGRAM
TABLE	TABLE	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	TABLE	IN	PROGRAM
	TABLE	HAS	COLUMN
	COLUMN	IN	PROGRAM
	IDMSRECORD	IN	AREA

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships	
AREA	IN	SUBSCHEMA
SUBSCHEMA	IN	SCHEMA
IDMSRECORD	IN	IDMSMAP
IDMSRECORD	IN	LOGICALRECORD
LOGICALRECORD	IN	PROGRAM
IDMSRECORD	IN	PROGRAM
DATAITEM	IN	RECORD
RECORD	REFERENCED-BY	MAP
MAP	IN	MAPSET
MAP	HAS	MAPFIELD
RECORD	USED BY	FD
FD	SELECTS	DD
DD	HAS	DSN
DSN	HAS	MEMBER
DD	IN	DBD
DD	IN	FCTY
DD	IN	CSD
SEGMENT	HAS	SEGFIELD
SEGMENT	IN	SEGMENT
SEGMENT	USES	SEGMENT
SEGMENT	USED BY	PROGRAM
SEGMENT	IN	DBD
SENSEG	USES	SENSEG
SENSEG	IN	SENSEG
SENSEG	HAS	SENFIELD

Starting Entity Kind	Entity Relationships		
	SENSEG	IN	PCB
	PCB	REFERENCES	DBD
	PCB	IN	PSB
	PSB	USED BY	PROGRAM
TABLESPACE	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	TABLESPACE	IN	PROGRAM
	TABLESPACE	HAS	VIEW
	VIEW	IN	PROGRAM
	VIEW	HAS	COLUMN
	TABLESPACE	HAS	TABLE
	TABLE	IN	PROGRAM
	TABLE	HAS	COLUMN
	COLUMN	IN	PROGRAM
EW	VIEW	IN	TABLESPACE
	TABLESPACE	IN	DATABASE
	DATABASE	IN	STOGROUP
	VIEW	IN	PROGRAM
	VIEW	HAS	COLUMN
	COLUMN	IN	PROGRAM
	IDMSRECORD	IN	AREA
	AREA	IN	SUBSCHEMA
	SUBSCHEMA	IN	SCHEMA
	IDMSRECORD	IN	IDSMAP
IDMSRECORD	IN	LOGICALRECORD	

Appendix B - Impact Analysis Entity Relationships

Starting Entity Kind	Entity Relationships	
	LOGICALRECORD	IN PROGRAM
	IDMSRECORD	IN PROGRAM
	DATAITEM	IN RECORD
	RECORD	REFERENCED-BY MAP
	MAP	IN MAPSET
	MAP	HAS MAPFIELD
	RECORD	USED BY FD
	FD	SELECTS DD
	DD	HAS DSN
	DSN	HAS MEMBER
	DD	IN DBD
	DD	IN FCT
	DD	IN CSD
	SEGMENT	HAS SEGFIELD
	SEGMENT	IN SEGMENT
	SEGMENT	USES SEGMENT
	SEGMENT	USED BY PROGRAM
	SEGMENT	IN DBD
	SENSESEG	USES SENSESEG
	SENSESEG	IN SENSESEG
	SENSESEG	HAS SENFIELD
	SENSESEG	IN PCB
	PCB	REFERENCES DBD
	PCB	IN PSB
	PSB	USED BY PROGRAM

Appendix C

Managing the AKR

The ESW family of components provides AKR management tools through online utilities. For more information on online utilities, see the *ASG-Application Definition and Analysis User's Guide*.

Allocating the AKR

An AKR is a BDAM or VSAM file. The AKR is the repository for all of the information used by the ESW family of components.

You can create a new AKR, or expand an existing one. Each AKR can contain multiple applications and you can allocate multiple AKRs for a single change, as necessary. You may want to allocate one AKR for each major business unit in your company (for example, manufacturing, accounting, sales). The size of the AKR depends on the number and size of the application definitions it contains.

Allocating the AKR with ISPF

You can allocate an AKR with or without using ISPF. For step-by-step instructions on allocating the AKR without using ISPF, see ["Allocating and Expanding AKRs without ISPF" on page 184](#).

To use ISPF to allocate an AKR

- 1 Choose File ▶ AKR utility from Alliance Primary screen and press Enter. The ESW - AKR Utility pop-up (see [Figure 68](#)) displays.

Figure 68 • ESW - AKR Utility Pop-up

```

                                     ESW - AKR Utility
Command ==> -----
      Blank - Display member list      D - Delete member
      A     - Allocate/expand AKR      R - Rename member

Application Knowledge Repository (AKR):
Data set name . . 'VIAUSR_GENERAL.AKR'
Member . . . . . ----- (if "R" or "D" selected)
New name . . . . . ----- (if "R" selected)

Volume serial . . ----- (if not cataloged)
Password . . . . . ----- (if password protected)
```

- 2 In the Dataset name field, type the name of the AKR to be allocated. If the high-level qualifier is different than your TSO prefix or user ID, enter the name in single quotes. The last node of the name should be AKR.
- 3 Type A on the command line to specify the allocate function and press Enter.
- 4 If the AKR will not be cataloged, in the Volume serial field, type the serial number of the volume where the AKR will reside.
- 5 To password-protect the AKR, type the password in the Password field.
- 6 Press Enter to open the File - AKR Allocate/Expand pop-up.

- 7 The pop-up that displays depends on whether you use BDAM or VSAM AKRs and whether SMS is turned on for your site. [Figure 69](#) assumes a BDAM-VSAM sequential AKR.

Figure 69 • File - AKR Allocate/Expand Pop-up with a VSAM AKR

```

File - AKR Allocate/Expand
Command ==> _____
          S - Submit JCL      E - Edit JCL      C - Specify Catalog
Expand existing AKR . . . NO          (Yes or No)
AKR data set name . . . . 'USER.TEST.AKR'
Volume . . . . . _____
Unit . . . . . _____          (Generic unit name)
Space units . . . . . RECORDS      (Records, Tracks or Cylinders)
Primary space . . . . . 4000        (Primary amount in above units)
Secondary space . . . . . 0         (Secondary amount in above units)

Job statement information:
//NAME      JOB (ACCOUNT),NAME,
//          MSGCLASS=A
/** INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
/**
  
```

- 8 Specify this information:

Field name	Enter this Information
Expand existing AKR field	Specifies whether you want to expand an existing AKR. Type NO in this field.
Volume field	Specifies the volume serial number for the AKR. This field is required if Storage class is not specified.
Unit field	Specifies the generic unit name.
Space units field	Specifies the type of units to allocate. The units must be records, tracks, or cylinders.
Primary space field	Specifies the amount of space required. For information on the expected size of the AKR and analyze work files, see the <i>ASG-Application Definition and Analysis User's Guide</i> .
Secondary space field	Specifies the amount of secondary space required.
Job statement information field	Contains the job card and routing information.

- 9 To specify a private catalog to contain the name of the AKR, type C on the Command line and press Enter. The AKR Catalog Information pop-up displays.
- 10 Type the catalog dataset name and password.
- 11 To submit the AKR for allocation, type S in the command line and press Enter.
Or
To edit the JCL, type E in the command line and press Enter.
- 12 Modify the JCL, then type SUBMIT.
- 13 Type End on the command line.
Or
Press PF3 to return to the File - AKR Allocate/Expand pop-up.
- 14 Verify the allocation job by examining the job output. The allocation is successful when the condition code 0 appears in the output.

After the AKR is successfully allocated, create the application. For step-by-step instructions on creating and defining the application, see the *ASG-Application Definition and Analysis User's Guide*.

Allocating and Expanding AKRs without ISPF

The batch AKR utility can allocate and expand VSAM AKRs without using ISPF. ESW provides you with the VIASAKRA JCL to allocate an AKR, and the VIASAKRX JCL to expand an AKR. [Figure 70](#) shows the VIASAKRA JCL. [Figure 71 on page 186](#) shows the VIASAKRX JCL.

Figure 70 • VIASAKRA JCL for a VSAM AKR

```
)CM *****
00010000
)CM * ASG, INC.           ASG-CENTER           MM-YYYY           * 00020003
)CM *
)CM * SKELETON JCL FOR SUBMITTING A BATCH JOB TO ALLOCATE AND           * 00040000
)CM * INITIALIZE A ASG APPLICATION KNOWLEDGE REPOSITORY (AKR) .           * 00050003
)CM *****
00060000
)CM
)CM
)CM *****
)CM * JOB STATEMENT INFORMATION *
)CM *****
)CM
&VSV11JB1
&VSV11JB2
&VSV11JB3
&VSV11JB4
)CM
```

```

)CM *****                                00180000
)CM * IN-STREAM JCL PROCEDURE *           00190000
)CM *****                                00200000
)CM                                         00210000
)IM VIASAKAP OPT                          00220000
//      PEND                               00230000
//*                                         00240000
)CM                                         00250000
)CM *****                                00260000
)CM * EXECUTE JCL STATEMENTS *           00270000
)CM *****                                00280000
)CM                                         00290000
//VIASAKRA EXEC VIASAKAP                  00300000
//*                                         00310000
)SEL &VSV32VAC = YES                       00320000
//DEFAKR.SYSIN DD *                       00330000
  DEFINE CLUSTER -                        00340000
    (NAME(&VSVIDB) -                      00350000
)SEL &VSV32SPU = CYLINDERS                  00360000
    CYLINDERS (&VSV32SPA) -              00370000
)ENDSEL                                    00380000
)SEL &VSV32SPU = RECORDS                   00390000
    RECORDS (&VSV32SPA) -                00400000
)ENDSEL                                    00410000
)SEL &VSV32SPU = TRACKS                   00420000
    TRACKS (&VSV32SPA) -                 00430000
)ENDSEL                                    00440000
)SEL &VSV32DCL a= &Z                       00450000
    DATACLAS (&VSV32DCL) -              00460000
)ENDSEL                                    00470000
)SEL &VSV32MCL a= &Z                       00480000
    MGMTCLAS (&VSV32MCL) -               00490000
)ENDSEL                                    00500000
)SEL &VSV32VOL a= &Z                       00510000
    VOLUME (&VSV32VOL) -                 00520000
)ENDSEL                                    00530000
)SEL &VSV32SCL a= &Z                       00540000
    STORCLAS (&VSV32SCL) -               00550000
)ENDSEL                                    00560000
    CONTROLINTERVALSIZE(4096) -           00570000
    NUMBERED -                             00580000
    RECORDSIZE(4089 4089) -               00590000
    RECOVERY -                             00600000
)SEL &VSV32UNQ a= NO                       00610000
    UNIQUE -                               00620000
)ENDSEL                                    00630000
    SHAREOPTIONS(3 3) -                   00640000
)SEL &VSV32CAT a= &Z && &VSV32PSW = &Z    00650000
    CATALOG (&VSV32CAT) -                 00660000

```

To allocate an AKR without ISPF

- 1 Replace *XX* in the *NAME (XX)*, *NAME (XX.DATA)*, and the *DSNAME (XX)* parameters with the name of the AKR to be allocated.
- 2 Replace *XX (XX)* with the allocation units values and quantities for your site.

[Figure 71](#) shows an example of an AKR VIASAKRX JCL.

Figure 71 • VIASAKRX JCL for a VSAM AKR

```

)CM *****
00010000
)CM * ASG, INC.           ASG-CENTER           MM-YYYY           *   00020003
)CM *                   *                   *                   *   00030000
)CM * SKELETON JCL FOR SUBMITTING A BATCH JOB TO EXPAND A           *   00040000
)CM *   ASG APPLICATION KNOWLEDGE REPOSITORY (AKR) .                 *   00050003
)CM *****
00060000
)CM                                           00070000
)CM                                           00080000
)CM *****                               00090000
)CM * JOB STATEMENT INFORMATION *                               00100000
)CM *****                               00110000
)CM                                           00120000
&VSV11JB1                                   00130000
&VSV11JB2                                   00140000
&VSV11JB3                                   00150000
&VSV11JB4                                   00160000
)CM                                           00170000
)CM *****                               00180000
)CM * IN-STREAM JCL PROCEDURE *                               00190000
)CM *****                               00200000
)CM                                           00210000
)IM VIASAKXP OPT                               00220000
//      PEND                               00230000
// *                                         00240000
)CM                                           00250000
)CM *****                               00260000
)CM * EXECUTE JCL STATEMENTS *                               00270000
)CM *****                               00280000
)CM                                           00290000
//VIASAKRX EXEC VIASAKXP                       00300000
// *                                         00310000
)SEL &VSV32VAC = YES                           00320000
//DEFAKR.SYSIN DD *                             00330000
  DEFINE CLUSTER -                             00340000
      (NAME (&VSVIDB..EX) -                     00350000
)SEL &VSV32SPU = CYLINDERS                       00360000
      CYLINDERS (&VSV32SPA) -                   00370000
)ENDSEL                                         00380000
)SEL &VSV32SPU = RECORDS                         00390000
      RECORDS (&VSV32SPA) -                     00400000
)ENDSEL                                         00410000
)SEL &VSV32SPU = TRACKS                         00420000
      TRACKS (&VSV32SPA) -                       00430000
)ENDSEL                                         00440000
)SEL &VSV32DCL ^a= &Z                           00450000
      DATACLAS (&VSV32DCL) -                   00460000
)ENDSEL                                         00470000

```

```

)SEL &VSV32MCL a= &Z                                00480000
      MGMTCLAS(&VSV32MCL) -                          00490000
)ENDSEL                                              00500000
)SEL &VSV32VOL a= &Z                                00510000
      VOLUME(&VSV32VOL) -                            00520000
)ENDSEL                                              00530000
)SEL &VSV32SCL a= &Z                                00540000
      STORCLAS(&VSV32SCL) -                         00550000
)ENDSEL                                              00560000
      CONTROLINTERVALSIZE(4096) -                   00570000
      NUMBERED -                                     00580000
      RECORDSIZE(4089 4089) -                       00590000
      RECOVERY -                                     00600000
)SEL &VSV32UNQ a= NO                                00610000
      UNIQUE -                                       00620000
)ENDSEL                                              00630000
      SHAREOPTIONS(3 3) -                            00640000
)SEL &VSV32CAT a= &Z && &VSV32PSW = &Z             00650000
      CATALOG(&VSV32CAT) -                          00660000
)ENDSEL                                              00670000
)SEL &VSV32CAT a= &Z && &VSV32PSW a= &Z           00680000
      CATALOG(&VSV32CAT/&VSV32PSW) -                00690000
)ENDSEL                                              00700000
      DATA -                                       00710000
      (NAME(&VSVIDB..EX.DATA))                      00720000
)ENDSEL                                              00730000
/*                                                  00740000
//INITAKR.STEPLIB DD DSN=VIASHxx.PROD.XALOAD,DISP=SHR ** IN-HOUSE** 00750002
//INITAKR.VIASYSIN DD *                             00770000
INIT DSNAME(&VSVIDB..EX)                            00780000
/*                                                  00790000
)SEL &VSV32CAT a= &Z                                00800000
//INITAKR.STEPCAT DD DSN=&VSV32CAT,                 00810000
//          DISP=SHR                                00820000
)ENDSEL                                              00830000
/*                                                  00840000
)SEL &VSV32EXD a= MIX                               00850000
//REPRO.SYSIN DD *                                  00860000
REPRO INDATASET(&VSVIDB) -                          00870000
      OUTDATASET(&VSVIDB..EX) -                    00880000
      REPLACE                                       00890000
/*                                                  00900000
)SEL &VSV32CAT a= &Z                                00910000
//REPRO.STEPCAT DD DSN=&VSV32CAT,                   00920000
//          DISP=SHR                                00930000
)ENDSEL                                              00940000
/*                                                  00950000
//RESIZE.STEPLIB DD DSN=VIASHxx.PROD.XALOAD,DISP=SHR ** IN-HOUSE ** 00960002
//RESIZE.VIASYSIN DD *                              00980000
RESIZE DSNAME(&VSVIDB..EX)                          00990000
/*                                                  01000000
)SEL &VSV32CAT a= &Z                                01010000
//RESIZE.STEPCAT DD DSN=&VSV32CAT,                 01020000
//          DISP=SHR                                01030000
)ENDSEL                                              01040000
)ENDSEL                                              01050000
/*                                                  01060000
)SEL &VSV32EXD = MIX                                01070000
//COPY.VIASYSIN DD *                                01080000
COPY * NOREPLACE                                    01090000
)ENDSEL                                              01100000
//RENAME.SYSIN DD *                                 01110000

```

```

                                                                    01120000
DELETE  &VSVIDB PURGE                                             01130000
                                                                    01140000
ALTER  &VSVIDB..EX -                                             01150000
        NEWNAME(&VSVIDB)                                         01160000
)SEL  &VSV32VAC = YES                                           01170000
ALTER  &VSVIDB..EX.DATA -                                       01180000
        NEWNAME(&VSVIDB..DATA)                                   01190000
)ENDSEL                                                           01200000
//*                                                                01210000
)SEL  &VSV32CAT  ^= &Z                                          01220000
//RENAME.STEPCAT DD DSN=&VSV32CAT,                               01230000
//          DISP=SHR                                           01240000
)ENDSEL                                                           01250000
//*                                                                01260000
//DELETE.SYSIN DD *                                           01270000
DELETE  &VSVIDB..EX PURGE                                       01280000
//*                                                                01290000
)SEL  &VSV32CAT  ^= &Z                                          01300000
//DELETE.STEPCAT DD DSN=&VSV32CAT,                               01310000
//          DISP=SHR                                           01320000
)ENDSEL                                                           01330000
//*                                                                01340000
)CM                                         01350000
)CM  *** END OF VIASAKRX ***                                       01360000
```

To expand an AKR without using ISPF

- 1** Replace *XX* in the *NAME (XX.EX)*, *NAME (XX.EX.DATA)*, *DSNAME (XX.EX)*, *IN DATASET (XX)*, and the *OUT DATASET (XX.EX)* parameters with the name of the AKR to be expanded.
- 2** Replace *XX(XX)* with the allocation units values and quantities for your site.

To rename an AKR without using ISPF

- 1** Replace the *XX* in the DELETE statement with the AKR to be renamed.
- 2** Type the new AKR name in the *NEWNAME (XX)* parameter.

Appendix D

Using the Import and Export Facility

Importing an Application Definition

The Import facility imports application definitions into Alliance.

To prepare the application definition for importing, follow this step:

- ▶ Edit an existing file or create a new file using keywords from the import language described in the *ASG-Application Definition and Analysis User's Guide*.

These keywords enable Alliance to identify the components and attributes for each library and member included in the file and to convert the application definition to the structure required by the Application Analytical Engine.

To import an application definition

- 1 Using the import language described in the *ASG-Application Definition and Analysis User's Guide*, edit or create a file containing the keywords to identify the application information, defaults, and the components and attributes of the application definition.
- 2 Save the file.
- 3 Start Alliance using the procedure defined for your site.
- 4 Select File ▶ Import/Export from the Alliance Primary screen and press Enter.
- 5 Select Import Definition and press Enter.
- 6 In the Dataset name field of the File - Import Definition pop-up, type the name of the dataset containing the import file you created in [step 1](#).

If the high-level qualifier is different than your TSO prefix or user ID, enter the name in single quotes ('). For example, 'USER.TEST.APPLDEF'.

- 7 In the Member name field, type the name of the member containing the import information and press Enter.

The message APPLICATION <*application name*> IMPORTED SUCCESSFULLY displays.

Note: _____

If the import does not complete successfully, verify the dataset and member names for the import file. If the names are correct, review the structure and syntax of the import file to locate any errors or invalid data input.

Exporting the Application Information

You can export both the application definition and the information produced from an analyze job. Store the exported definition in a sequential file or a PDS member. Load exported application analysis information into DB2 tables or input it to other database managers. For information on loading into DB2, see ["Defining and Loading DB2 Databases" on page 234](#). For information on exporting application information to the Texas Instruments Transition Enablement Facility product, see ["Exporting to Transition Enablement Facility \(TEF\)" on page 301](#).

For AutoChange users, Alliance provides an automated export facility that stores both the application definition and information produced from an analyze job in PDS members for use by AutoChange. For information on using this facility see ["Using Export for AutoChange" on page 200](#).

Exporting Application Definition Data

The exported application definition consists of a series of keyword control statements with a syntax that is consistent with the requirements of the Import facility. These are the items the keyword control statements identify:

- The AKR
- The application name and description
- The application components and their associated attributes

For a description of the keyword control statement syntax, see ["Importing an Application Definition" on page 189](#).

Use the exported application definition to perform these tasks:

- Create a backup of the application definition for easy recovery.
- Copy an application definition from one AKR to another.
- Copy an application definition to use as a template when you are creating a similar definition.

To export the application definition

- 1 Identify or create a sequential file or a PDS to contain the exported definition. The file must have a record length of 80.
- 2 Select File ► Import/Export from the Alliance Primary screen and press Enter. The File - Export Definition pop-up (see [Figure 72](#)) displays.

Figure 72 • File - Export Definition Pop-up

```
Command ==>> File - Export Definition
Type AKR and Export information, then press enter.
Application Knowledge Repository (AKR):
  Data set name . . . 'VIAUID.TEST_AKR'
  Application name . .
Export:
  Data set name . . . _____
  Member . . . . .
```

- 3 On the File - Export Definition pop-up, specify this information:
 - The AKR dataset name containing the application to export
 - The application name
 - The export definition dataset name
 - If the export definition dataset is a PDS, specify the export definition member name
- 4 Press Enter to execute the export request.

Note: _____

When the export of the application definition is complete, the message *<application name> EXPORTED* displays in the short message area of the File - Export Definition pop-up.

Copying Application Definition Data

Edit the exported definition to use the import facility to copy the exported application definition to an AKR (the original AKR or another AKR).

To copy the exported application definition

- 1 On the File - Export Definition pop-up, enter the dataset name.
- 2 Enter the application name and press Enter.

Note: _____

Application names must be unique within an AKR.

- 3 To complete this process, follow the instructions located in ["Importing an Application Definition" on page 189](#).

Creating an Application Definition Template

Edit the exported definition to use the Import facility to create an application definition template.

To create a template from the exported application

- 1 On the File - Export Definition pop-up, enter the dataset name.
- 2 Enter the application name.
- 3 To update the application definition, use one of these methods:

Edit the exported application definition file. Modify the keyword control statements, and their associated attribute keyword control statements, to create the new application definition.

Or

Import the application definition into the target AKR (see ["Importing an Application Definition" on page 189](#)). Then use the online application definition function to modify the imported definition as appropriate.

Exporting Application Analysis Results

Exported application analysis information consists of comma delimited output files created for each application entity and relationship. Use the exported application information files to load DB2 tables, or transfer it to your workstation for input to database applications. For more information about AKR export output, see ["Using the Output from an AKR Export" on page 207](#).

Use the exported application analysis information as part of the input to develop these types of business information:

- Planning models
- Impact of change analyses
- Where-used queries
- Work plans
- Manpower requirement estimates

Export application information using the online facility or using batch JCL supplied with the product.

Using the Online Facility to Export Analysis Data

To export application information using the online facility

- 1 Start the product using the procedure defined for your site.
- 2 Select File ► Import/Export from the Alliance Primary screen and press Enter. The File - Import/Export pop-up displays.
- 3 Select Export Data and press Enter. The Data Export pop-up (see [Figure 73](#)) displays.

Figure 73 • Data Export Pop-up

```

                                Data Export
Command ==> -----
Specify options and jobcard information. Then press PF key for action.
WARNING! Selecting "Edit JCL" or "Submit JCL" will delete any existing
export files created under this userid.

Export Type --  1. Flat Files                DB2 Subsystem: ----
                2. DB2 Load
                3. COOL:Gen Export

Job statement information:
//NAME          JOB (ACCOUNT),NAME,
//              MSGCLASS=A
//*            INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
//*

PF4=COOL:Gen Export Options  PF5=Edit JCL  PF6=Submit JCL

```

- 4 Type the DB2 subsystem name to immediately load the files into a DB2 database in the DB2 Subsystem field.

- 5 In the Export Type field, perform one or more of these actions:

If you want	Specify Export Type
Sequential files containing the exported application data	<p>Flat File. For example, you might want to create a sequential file and bypass the DB2 database load.</p> <p>Note: _____ If you specify Flat File, the sequential files containing the exported application data are catalogued when you submit the export job.</p> <p>_____</p> <p>Use the data as input to a batch DB2 load or other software, or transfer it to your workstation for input to a database manager.</p>
Application data exported into sequential files and then loaded into a DB2 database as part of the same job	<p>DB2 Load. The appropriate DB2 entities must be defined before the DB2 export.</p> <p>Note: _____ If you specify that a DB2 load is performed as part of the export job, the default JCL deletes the files containing the exported data after a successful load. To retain these files, edit the JCL before submitting the job, and change the disposition JCL parameters.</p>
Application data exported to the TI Transition Enablement Facility	<p>COOL:Gen Export. For more information about this option, see "Exporting to Transition Enablement Facility (TEF)" on page 301.</p>

- 6 Type the job card information required for your site in the Job Statement information field.

- 7 To submit the export job, choose one of these options:

To edit the generated JCL file before submitting the export job, press PF5 to edit JCL. Type `SUBMIT` to submit the export job.

Or

To execute the export request without editing the generated JCL, press PF6 to submit JCL.

Using Batch JCL to Export Analysis Data

You can also use batch JCL to export application data. The JCL has symbolic parameters you can edit to meet your export requirements. Edit and submit these jobs when you prefer not to start the online application.

The batch JCL members you need to accomplish these tasks are located in this PDS:

`ASG.CENTER.CNTL`

If you need help locating the JCL, contact your systems administrator.

This is the JCL included with the Export facility:

Dataset Name	Description
VIABXTRJ	Exports application data from a ESW AKR into sequential files. See Figure 74 on page 196 .
VIABXDFJ	Defines a DB2 database. See Figure 75 on page 196 .
VIABXDF1	Inputs CREATE control statements for VIABXDFJ. See Figure 76 on page 197 .
VIABXDF2	Inputs DSN control statements for VIABXDFJ. See Figure 77 on page 197 .
VIABXLD	Loads application data from sequential files into a DB2 database. See Figure 78 on page 198 .
VIABXLD1	Inputs LOAD control statements for VIABXLDJ. See Figure 79 on page 199 .

Figure 74 • Sample VIABXTRJ

```

// * ASG JOB ( ), 'ASG EXPORT EXTR'                                00010001
// * INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.            00020000
// *                                                              00030000
// * *****                                                    00040000
// * * ASG, INC.          ASG-ALLIANCE          MM-YYYY          * 00050001
// * *                                                              * 00060000
// * * UNLOAD APPLICATION DATA FROM A ASG AKR INTO FLAT FILES    * 00070001
// * *                                                              * 00080000
// * * NOTE:  ELRECL & RLRECL ARE ALREADY MINIMIZED TO REFLECT  * 00090000
// * *          THE INFORMATION OUTPUT BY THE EXPORT FACILITY.    * 00100000
// * *                                                              * 00110000
// * * *****                                                    00120000
// * *                                                              00130000
// * *                                                              00140000
// * * VIABXTR PROC SYSOUT='*',          PRINTED OUTPUT MESSAGE CLASS 00150001
// * *          VIASOFT='ASG',          HIGH LEVEL NODE OF ASG DATA SETS 00160001
// * *          CENTER='VIACENXX',     MIDDLE NODE OF ASG DATA SETS    00170001
// * *          AKR='XXX.YYY.ZZZ'     ASG AKR                             00180000
// * *                                                              00190000
// * * EXTRACT EXEC PGM=VIABEXPT,REGION=4096K,                    00200000
// * *          PARM=('APPL=APPLNAME')                                00210000
// * *                                                              00220000
// * * STEPLIB DD DSN=&VIASOFT..&CENTER..LOADLIB,DISP=SHR        00230000
// * * VIAAKR DD DSN=&AKR,DISP=SHR                                  00240000
// * * VIAUT3 DD UNIT=SYSDA,SPACE=(CYL,(10,10))                   00250000
// * * VIALOG DD SYSOUT=&SYSOUT                                     00260000
// * * VIAARPT DD SYSOUT=&SYSOUT                                    00270000
// * * VIAAPRINT DD SYSOUT=&SYSOUT                                  00280000
// * * SYSPRINT DD SYSOUT=&SYSOUT                                   00290000
// * *                                                              00300000
// * *          PEND                                              00310000
// * *                                                              00320000
// * * STEP1 EXEC VIABXTR                                          00330000
// * *                                                              00340000

```

Figure 75 • Sample VIABXDFJ

```

// * ASG JOB ( ), 'ASG EXPORT DEF'                                00010001
// * INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.            00020000
// *                                                              00030000
// * *****                                                    00040000
// * * ASG, INC.          ASG-ALLIANCE          MM-YYYY          * 00050001
// * *                                                              * 00060000
// * * * DEFINE DB2 DATABASE FOR EXPORT FEATURE                    * 00070000
// * * *                                                              * 00080000
// * * *****                                                    00090000
// * * *                                                              00100000
// * * * JOBLIB DD DSN=DB2.DSNLOAD,DISP=SHR                       00110000
// * * *                                                              00120000
// * * * DEFINE EXEC PGM=IKJEFT01,DYNAMNBR=20                    00130000
// * * * SYSTSPRT DD SYSOUT=*                                       00140000
// * * * SYSTSIN DD *                                              00150000
// * * * SYSPRINT DD SYSOUT=*                                       00160000
// * * * SYSUDUMP DD SYSOUT=*                                       00170000
// * * * SYSIN DD DSN=ASG.VIACENXX.CNTL(VIABXDF1),DISP=SHR       00180001
// * * * SYSTSIN DD DSN=ASG.VIACENXX.CNTL(VIABXDF2),DISP=SHR   00190001
// * * *                                                              00200000

```

Figure 76 • Sample VIABXDF1

```
CREATE TABLE ASG.PROGRAM
  (PROGRAM_KEY      CHAR(8) NOT NULL,
   PROGRAM_NAME     CHAR(8) NOT NULL,
   PRIMARY KEY (PROGRAM_KEY))

  IN DATABASE USERDB;

CREATE UNIQUE INDEX ASG.PROGRAM_KEY_INX
  ON ASG.PROGRAM
  (PROGRAM_KEY ASC)
  USING STOGROUP USERSTGP
  PRIQTY 10
  ERASE NO
  BUFFERPOOL BP0
  CLOSE YES;

CREATE INDEX ASG.PROGRAM_NAME_INX
  ON ASG.PROGRAM
  (PROGRAM_NAME ASC)
  USING STOGROUP USERSTGP
  PRIQTY 10
  ERASE NO
  BUFFERPOOL BP0
  CLOSE YES;

COMMIT;
```

Figure 77 • Sample VIABXDF2

```
DSN SYSTEM(USERSYS)
RUN PROGRAM(DSNTIAD) LIB('USER.DB2.LOADLIB') PLAN(USERPLAN)
```

Figure 78 • Sample VIABXLDJ

//ASG JOB (), 'ASG EXPORT LOAD'	00010001
/** INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.	00020000
/**	00030000
/** *****	00040000
/** * ASG, INC. ASG-ALLIANCE MM-YYYY *	00050001
/** *	00060000
/** * LOAD APPLICATION DATA FROM FLAT FILES TO DB2 *	00070000
/** *	00080000
/** *****	00090000
/**	00100000
//VIABXLD EXEC DSNUPROC, PARM= ('SUBSYSTEM')	00110000
/**	00120000
//SORTLIB DD DSN=SYS1.SORTLIB, DISP=SHR	00130000
//SORTOUT DD UNIT=SYSDA, SPACE=(CYL, (10,10),,,ROUND)	00140000
//SORTWK01 DD UNIT=SYSDA, SPACE=(CYL, (10,10),,,ROUND)	00150000
//SORTWK02 DD UNIT=SYSDA, SPACE=(CYL, (10,10),,,ROUND)	00160000
//SORTWK03 DD UNIT=SYSDA, SPACE=(CYL, (10,10),,,ROUND)	00170000
//SORTWK04 DD UNIT=SYSDA, SPACE=(CYL, (10,10),,,ROUND)	00180000
//DSNTRACE DD SYSOUT=*	00190000
//SYSMAP DD UNIT=SYSDA, SPACE=(CYL, (1,1))	00200000
//SYSUT1 DD UNIT=SYSDA, SPACE=(CYL, (50,50),,,ROUND)	00210000
/**	00220000
//SYSIN DD DSN=ASG.VIACENXX.CNTL (VIABXLD1), DISP=SHR	00230001
/**	00240000
//DDENT000 DD DSN=USER.VIABEXPT.EXPORT.EPGM, DISP=(SHR,KEEP,KEEP)	00250000
//DDENT001 DD DSN=USER.VIABEXPT.EXPORT.EENTRY, DISP=(SHR,KEEP,KEEP)	00260000
//DDENT002 DD DSN=USER.VIABEXPT.EXPORT.ECOPYMEM, DISP=(SHR,KEEP,KEEP)	00270000
//DDENT003 DD DSN=USER.VIABEXPT.EXPORT.EFD, DISP=(SHR,KEEP,KEEP)	00280000
//DDENT004 DD DSN=USER.VIABEXPT.EXPORT.ERECORD, DISP=(SHR,KEEP,KEEP)	00290000
//DDENT005 DD DSN=USER.VIABEXPT.EXPORT.EDATAITM, DISP=(SHR,KEEP,KEEP)	00300000
//DDENT006 DD DSN=USER.VIABEXPT.EXPORT.EIDMS, DISP=(SHR,KEEP,KEEP)	00310000
//DDENT007 DD DSN=USER.VIABEXPT.EXPORT.ESQL, DISP=(SHR,KEEP,KEEP)	00320000
//DDENT008 DD DSN=USER.VIABEXPT.EXPORT.ECSECT, DISP=(SHR,KEEP,KEEP)	00330000
//DDENT009 DD DSN=USER.VIABEXPT.EXPORT.ELOADMOD, DISP=(SHR,KEEP,KEEP)	00340000
//DDENT010 DD DSN=USER.VIABEXPT.EXPORT.EJOB, DISP=(SHR,KEEP,KEEP)	00350000

Figure 79 • Sample of VIABXLD1

```
LOAD DATA REPLACE
  INDDN (DDENT000) INTO TABLE VIASOFT.PROGRAM
  (PROGRAM_KEY      POSITION(01:08)  CHAR(8),
   PROGRAM_NAME     POSITION(10:17)  CHAR(8))

LOAD DATA REPLACE
  INDDN (DDENT001) INTO TABLE VIASOFT.ENTRY
  (ENTRY_KEY        POSITION(01:08)  CHAR(8),
   ENTRY_NAME       POSITION(10:19)  CHAR(10))

LOAD DATA REPLACE
  INDDN (DDENT002) INTO TABLE VIASOFT.COPYMEMBER
  (COPYMEMBER_KEY   POSITION(01:08)  CHAR(8),
   COPYMEMBER_NAME  POSITION(10:19)  CHAR(10))

LOAD DATA REPLACE
  INDDN (DDENT003) INTO TABLE VIASOFT.FD
  (FD_KEY           POSITION(01:08)  CHAR(8),
   FD_NAME          POSITION(10:39)  CHAR(30))

LOAD DATA REPLACE
  INDDN (DDENT004) INTO TABLE VIASOFT.RECORD
  (REC_KEY          POSITION(01:08)  CHAR(8),
   REC_NAME         POSITION(10:39)  CHAR(30))

LOAD DATA REPLACE
  INDDN (DDENT005) INTO TABLE VIASOFT.DATAITEM
  (DATA_KEY         POSITION(01:08)  CHAR(8),
   DATA_NAME       POSITION(10:39)  CHAR(30),
   DATA_LEVEL      POSITION(41:42)  CHAR(2),
   DATA_PARENT     POSITION(44:51)  CHAR(8),
   DATA_CHILD      POSITION(53:60)  CHAR(8),
   DATA_NEXT_SIBLING POSITION(62:69) CHAR(8),
   DATA_PREV_SIBLING POSITION(71:78) CHAR(8),
```

Using Export for AutoChange

Use this procedure to export an application definition and the corresponding data resulting from an analyze job.

Note:

To successfully complete this export procedure, the application must be semantically linked and free of errors.

- 1 From the Alliance Primary screen (see [Figure 80](#)) open the application you want to export.

Figure 80 • Alliance Primary Screen

```
File Edit View Facility Options Help
-----
ASG-Alliance
Command ==> _____

*****
*****
****          ****          ****
****          ****          ****
*****
*****
****          ****          ****
****          ****          ****
****          ****          ****
****          ****          ****
****          ****          ****
****          ****          ****
****          ****          ****

Copyright Allen Systems Group, Inc., an unpublished work.
A proprietary product of ASG, Inc. Use restricted to authorized licensees.
Visit the ASG Support Web Site at www.asg.com
```

- 2 Select Facility ► Impact and press Enter. The Impact Facility screen displays (see [Figure 81](#)).

Figure 81 • Impact Facility Screen for Selected Application

```
File Edit View Search Generate Worklist Options Help
-----
Impact Facility
Command ==> _____ Scroll ==> CSR
Name:

***** TOP OF DATA *****
***** BOTTOM OF DATA *****
```

- 3 On the Impact Facility screen, select File ▶ New Impact List and press Enter. The New Impact pop-up displays (see [Figure 82](#)).

Figure 82 • New Impact Pop-up

```

                                New Impact
Command ==> -----
ASG3578I IMPACT "TESTENTR2" SUCCESSFULLY OPENED.
Specify the name and description. Then press PF key for action.

Name . . . . . TESTENTR2
Description . . . . . TEST FILE

                                PF4=Edit Impact  PF5=Summary Impact  PF6=Detailed Impact
    
```

- 4 Enter the Name (the Name field is 8 characters in length) and Description and press Enter.
- 5 Press PF4 to go the Impact Entity List (see [Figure 83](#)).

Figure 83 • Impact Entity List

```

                                Impact Entity List
Command ==> ----- Scroll ==> CSR
Press PF key for actions. Delete requires selection          1 of 2695
of entry(s).
Current Sort Key:  NAME
-----
Entity Kind          Name
-----
- PROGRAM/PROCEDURE/FUNCTION  .BACS-DATE
- PROGRAM/PROCEDURE/FUNCTION  .CD05C-ERROR-CODES
- PROGRAM/PROCEDURE/FUNCTION  .DATE6-YY
- PROGRAM/PROCEDURE/FUNCTION  .US-DATE8-YY
- PROGRAM/PROCEDURE/FUNCTION  .US-8-YY
- ENTRY                   'CD05B'
- ENTRY                   'CD05C'
- ENTRY                   'CD10A'
- ENTRY                   'CD10B'
- ENTRY                   'CD15B'

                                PF4=Add entity  PF5=Delete entity  PF6=Change Sort
    
```

- 6 To add an entity, press PF4. The Impact Entities Selection Criteria pop-up displays (see [Figure 84](#)).

Figure 84 • Impact Entities Selection Criteria Pop-up

```

Impact Entities Selection Criteria
Command ==> ----- Scroll ==> CSR
Text: *
Entity kind (Default = All) 1 of 51
-----
- AREA
- CATALOG
- CICS-TRANSACTION
- COLUMN
- COPYMEMBER/INCLUDE
- CSD
- CSECT
- CURSOR
- DATABASE
- DATAITEM
- DED
- DD
- DSN
PF5=Select all PF6=Unselect all
    
```

- 7 Confirm that the Text field contains an asterisk (*), then select the Dataitem entity and press Enter. The Add Impact List Entity screen displays (see [Figure 85](#)).

Figure 85 • Add Impact List Entity Screen

```

Add Impact List Entity
Command ==> ----- Scroll ==> CSR
Select desired entries then press the Add key. 1 of 157
Previously selected entries are marked with an asterisk.
Text: *
Text Entity Kind
-----
/ END-INPUT DATAITEM
/ END-OF-COLS DATAITEM
/ FIRST-TIME DATAITEM
/ FIRST-TIME-SW DATAITEM
/ INPUT-END-OF-FILE DATAITEM
/ MISC-FIELDS DATAITEM
/ OTL-CHG-CC-4 DATAITEM
/ OTL-CHG-DD-1 DATAITEM
/ OTL-CHG-DD-2 DATAITEM
/ OTL-CHG-DD-3 DATAITEM
PF5=Select all PF6=Unselect all PF10=Add
    
```

- 8 Press PF5 to Select all selections, then press PF10 to Add the selection.
- 9 Press PF3 to return to the New Impact screen.

- 10 To generate the Summary Impact, press PF5. The Summary Impact - Select Impact Generation Entities screen displays (see [Figure 86](#)).

Figure 86 • Summary Impact - Select Impact Generation Entities Screen

```

Summary Impact - Select Impact Generation Entities
Command ==> ----- Scroll ==> CSR
Select the items from which the impact list will      1 of 514
be generated.

  Text                      Entity Kind          Qualification >
-----
- END-INPUT                 DATAITEM          INPUT-END-OF-FI
- END-OF-COLS              DATAITEM          QRY-INP-FIRST-B
- END-OF-COLS              DATAITEM          QRY-INP-FIRST-B
- FIRST-TIME               DATAITEM          FIRST-TIME-SW 0

PF2=Options  PF4=Generate  PF5=Select all  PF6=Hide entity kind

```

- 11 To select all entities, press PF5.

Or

Select individual entities to include in the summary.

- 12 Press PF2 for Options. The Impact Generate Options screen displays. Confirm you have chosen all options as shown in this example (see [Figure 87](#)).

Figure 87 • Impact Generate Options Screen

```

Impact Generate Options
Command ==> -----
Select desired Impact Generate Options.
/ Include Base Selection in results
/ Include Copy Detail in results

Base Impact Targets:
Targets
-----
/ Base Selection
/ Parents of Base Selection
/ Children of Base Selection

Application Level Synonyms
-----
/
/
/

Assignments
-----
To From Alternates
/ / /
/ / /
/ / /

Expanded Impact Targets:
Targets
-----
/ Base Impact Targets
/ Parents of Base Impact Targets
/ Children of Base Impact Targets

Application Level Synonyms
-----
/
/
/

Assignments
-----
To From Alternates
/ / /
/ / /
/ / /

PF4=Restore ASG Default Options  PF5=Detailed Options

```

- 13 Press PF3 to return to the Summary Impact - Select Impact Generation Entities screen.
- 14 To generate the entity list, press PF4. When the entity list is complete, the Impact Facility screen displays.
- 15 To create a worklist to store the saved results of this summary, select Worklist ▶ New worklist and press Enter. The New Worklist pop-up displays (see [Figure 88](#)).

Figure 88 • New Worklist Pop-up

```
Command ==> _____ New Worklist
Specify the worklist name and description. Then press Enter.
Name . . . . . WKLSTTST
Description . . . . . TEST WORKLIST
Data set . . . . . _____ (Optional)
```

- 16 Enter a name and description for the new worklist. Press Enter. If you do not enter a worklist name, Alliance assigns a default name to the worklist.
- 17 Select Worklist ▶ Write worklist and press Enter. The Impact Facility screen displays the statement VIEW CONTENTS HAS BEEN SAVED TO WORKLIST "XXXXX" where XXXXX is the name given to your worklist.
- 18 Select Worklist ▶ Close worklist and press Enter.
- 19 Select Generate ▶ ASG-AutoChange Export File and press Enter. The Generate Impact Export for ASG-AutoChange screen displays (see [Figure 89](#)).

Figure 89 • Generate Impact Export for ASG-AutoChange Screen

```
Batch Options Help
-----
Command ==> _____ Generate Impact Export For ASG-AutoChange
Application Definition:
Data Set Name 'VIAUID.EXPORT.DEF'
Member Name . . VIAQANEW
Export Data:
Data Set Name 'VIAUID.EXPORT.DATA'
Member Name . . VIAQANEW
Worklist Name 'VIAUID.ALN00003.VIAWKLST'
AKR DSN . . . . 'VIAQA.QAESW5B.ATCHMAST.AKR'
```

20 Enter this information:

- Application Definition:
 - Use an existing PDS for the dataset name and enter a new member name.
- Export Data:
 - Use an existing PDS for the dataset name and enter the member name that is the name of the application.
 - The worklist name is the name of the worklist created in [step 15](#). The AKR DSN is the name of the AKR where the application definition is stored.

Note:

All dataset names must be fully qualified and datasets should have a record length of 80 and record format of fixed block.

21 From the Batch Menu choose one of these actions:

To export the impact, submit the job. The Batch job is scheduled. A message displays stating the job was submitted.

Or

To edit the JCL, edit the job. The JCL edit facility displays (see [Figure 90](#)).

Figure 90 • JCL Edit Facility

```

File Edit Confirm Menu Utilities Compilers Test Help
EDIT      VIAUID.T145714.VIAJCL                Columns 00001 00072
Command ==>                                     Scroll ==> PAGE
***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG> your edit profile using the command RECOVERY ON.
000001 //VIAJFC JOB (ACCOUNT),NAME,
000002 //          MSGCLASS=A
000003 /** INSERT /*ROUTE PRINT NODE.USER' HERE IF NEEDED.
000004 /**
000005 /**
000006 /**ASG JOB ( ),'ASG EXPORT EXTR'
000007 /** INSERT /*ROUTE PRINT NODE.USER' HERE IF NEEDED.
000008 /**
000009 /** *****
000010 /** * ASG, INC.          ASG-ALLIANCE          MM-YYYY          *
000011 /** *
000012 /** * UNLOAD APPLICATION DATA FROM A ASG AKR INTO FLAT FILES *
000013 /** *
000014 /** * NOTE: ELRECL & RLRECL ARE ALREADY MINIMIZED TO REFLECT *
000015 /** * THE INFORMATION OUTPUT BY THE EXPORT FACILITY. *
000016 /** *
. . . . .

```

Appendix E

Export Language Reference

Using the Output from an AKR Export

The AKR export job creates a sequential file for each of the entities and relationships in the application. Transfer the files created by the AKR export job to your workstation and load them into your workstation database. You can also load these files into DB2 tables or other databases. For more information about exporting data to DB2, see ["Defining and Loading DB2 Databases" on page 234](#).

The format for application data files created for entities is described in the section under Entity Files. The format for application data files created for relationships is described under ["Relationship Files" on page 216](#).

Note:

The Assembler language does not create specific ASM entity kinds. All entity kinds look the same as their COBOL counterparts (for example, a DCB appears as an FD). For more information about Assembler language support, see the chapter regarding assembler language support in the *ASG-Application Definition and Analysis User's Guide*.

Entity Files

These are the application entity data types and export files:

Entity Type	Filename
AREA	<userid>.EXPORT.<application name>.AREA
	Col 1: AREA Key
	Col 2: AREA Name
CATALOG	<userid>.EXPORT.<application name>.CATALOG
	Col 1: CATALOG Key
	Col 2: CATALOG Name

Entity Type	Filename
CICS-TRANSACTION	<userid>.EXPORT.<application name>.CICSTRAN
	Col 1: CICS-TRANSACTION Key
	Col 2: CICS-TRANSACTION Name
COLUMN	<userid>.EXPORT.<application name>.COLUMN
	Col 1: COLUMN Key
	Col 2: COLUMN Name
COPYMEMBER	<userid>.EXPORT.<application name>.COPYMEMB
	Col 1: COPYMEMBER Key
	Col 2: COPYMEMBER Name
CSD	<userid>.EXPORT.<application name>.CSD
	Col 1: CSD Key
	Col 2: CSD Name
CSECT	<userid>.EXPORT.<application name>.CSECT
	Col 1: CSECT Key
	Col 2: CSECT Name
CURSOR	<userid>.EXPORT.<application name>.CURSOR
	CURSOR Key
	CURSOR Name
DATABASE	<userid>.EXPORT.<application name>.DATABASE
	Col 1: DATABASE Key
	Col 2: DATABASE Name

Entity Type	Filename
DATAITEM	<userid>.EXPORT.<application name>.DATAITEM
	Col 1: DATAITEM Key
	Col 2: DATAITEM Name
	Col 3: ATTRIBUTE Level
	Col 4: ATTRIBUTE Parent Key
	Col 5: ATTRIBUTE Child Key
	Col 6: ATTRIBUTE Next Key
	Col 7: ATTRIBUTE Prev Key
	Col 8: ATTRIBUTE Picture Clause
	Col 9: ATTRIBUTE Usage
	Col 10: ATTRIBUTE Occurs
	Col 11: ATTRIBUTE Redefines Key
	Col 12: ATTRIBUTE Rename Key
	Col 13: ATTRIBUTE Rename Through Key
	Col 14: ATTRIBUTE Size
	Col 15: ATTRIBUTE Offset
	Col 16: ATTRIBUTE Position
	Col 17: ATTRIBUTE Value Clause
	Col 18: ATTRIBUTE Uses
	Col 19: ATTRIBUTE Modifications
	Col 20: ATTRIBUTE References
	Col 21: ATTRIBUTE Conditional
	Col 22: ATTRIBUTE Arithmetic
	Col 23: ATTRIBUTE Create

Entity Type	Filename
	Col 24: ATTRIBUTE Read
	Col 25: ATTRIBUTE Update
	Col 26: ATTRIBUTE Delete
DBD	<userid>.EXPORT.<application name>.DBD
	Col 1: DBD Key
	Col 2: DBD Name
DD	<userid>.EXPORT.<application name>.DD
	Col 1: DD Key
	Col 2: DD Name
DSN	<userid>.EXPORT.<application name>.DSN
	Col 1: DSN Key
	Col 2: DSN Name
	Col 3: DSN Entity ATTRIBUTE
ENTRY	<userid>.EXPORT.<application name>.ENTRY
	Col 1: ENTRY Key
	Col 2: ENTRY Name
FCTCSD	<userid>.EXPORT.<application name>.FCTCSD
	Col 1: FCTCSD Key
	Col 2: FCTCSD Name
FD	<userid>.EXPORT.<application name>.FDFILEVA
	Col 1: FD Key
	Col 2: FD Name
	Col 3: T/F: Indicates whether a CREATE was used
	Col 4: T/F: Indicates whether a READ was used

Entity Type	Filename
	Col 5: T/F: Indicates whether a UPDATE was used
	Col 6: T/F: Indicates whether a DELETE was used
FORMAT	<userid>.EXPORT.<application name>.IMSFORMA
	Col 1: FORMAT Key
	Col 2: FORMAT Name
FCT	<userid>.EXPORT.<application name>.FCT
	Col 1: FCT Key
	Col 2: FCT Name
IDMS	<userid>.EXPORT.<application name>.IDMS
	Col 1: IDMS Key
	Col 2: IDMS Name
IDMSMAP	<userid>.EXPORT.<application name>.IDMSMAP
	Col 1: IDMSMAP Key
	Col 2: IDMSMAP Name
IDMSRECORD	<userid>.EXPORT.<application name>.IDMSRECO
	Col 1: IDMSRECORD Key
	Col 2: IDMSRECORD Name
	Col 3: T/F: Indicates whether a CREATE was used
	Col 4: T/F: Indicates whether a READ was used
	Col 5: T/F: Indicates whether a UPDATE was used
	Col 6: T/F: Indicates whether a DELETE was used
IMS DYN ALLOC MACRO	<userid>.EXPORT.<application name>.IMSDYNAL
	Col 1: IDMS DYN ALLOC MACRO Key
	Col 2: IDMS DYN ALLOC MACRO Name

Entity Type	Filename
IMS-TRANSACTION	<userid>.EXPORT.<application name>.IMSTRANS
	Col 1: IMS-TRANSACTION Key
	Col 2: IMS-TRANSACTION Name
JOB	<userid>.EXPORT.<application name>.JOB
	Col 1: JOB Key
	Col 2: JOB Name
LIBRARY	<userid>.EXPORT.<application name>.LIBRARY
	Col 1: LIBRARY Key
	Col 2: LIBRARY Name
LOADMODULE	<userid>.EXPORT.<application name>.LOADMODU
	Col 1: LOADMOD Key
	Col 2: LOADMOD Name
LOGICALRECORD	<userid>.EXPORT.<application name>.LOGICALR
	Col 1: LOGICALRECORD Key
	Col 2: LOGICALRECORD Name
MAP	<userid>.EXPORT.<application name>.MAP
	Col 1: MAP Key
	Col 2: MAP Name
MAPFIELD	<userid>.EXPORT.<application name>.MAPFIELD
	Col 1: MAPFIELD Key
	Col 2: MAPFIELD Name
MAPSET	<userid>.EXPORT.<application name>.MAPSET
	Col 1: MAPSET Key
	Col 2: MAPSET Name

Entity Type	Filename
MEMBER	<userid>.EXPORT.<application name>.MEMBER
	Col 1: MEMBER Key
	Col 2: MEMBER Name
PCB	<userid>.EXPORT.<application name>.PCB
	Col 1: PCB Key
	Col 2: PCB Name
PCT	<userid>.EXPORT.<application name>.PCT
	Col 1: PCT Key
	Col 2: PCT Name
PPT	<userid>.EXPORT.<application name>.PPT
	Col 1: PPT Key
	Col 2: PPT Name
PROC	<userid>.EXPORT.<application name>.PROC
	Col 1: PROC Key
	Col 2: PROC Name
PROGRAM	<userid>.EXPORT.<application name>.PROGRAMP
	Col 1: PROGRAM Key
	Col 2: PROGRAM Name
PSB	<userid>.EXPORT.<application name>.PSB
	Col 1: PSB Key
	Col 2: PSB Name
RECORD	<userid>.EXPORT.<application name>.RECORDS
	Col 1: RECORD Key
	Col 2: RECORD Name

Entity Type	Filename
SCHEMA	<userid>.EXPORT.<application name>.SCHEMA
	Col 1: SCHEMA Key
	Col 2: SCHEMA Name
SEGFIELD	<userid>.EXPORT.<application name>.SEGFIELD
	Col 1: SEGFIELD Key
	Col 2: SEGFIELD Name
SEGMENT	<userid>.EXPORT.<application name>.SEGMENT
	Col 1: SEGMENT Key
	Col 2: SEGMENT name
	Col 3: T/F: Indicates whether a CREATE was used
	Col 4: T/F: Indicates whether a READ was used
	Col 5: T/F: Indicates whether a UPDATE was used
	Col 6: T/F: Indicates whether a DELETE was used
SENFIELD	<userid>.EXPORT.<application name>.SENFIELD
	Col 1: SENFIELD Key
	Col 2: SENFIELD Name
SENSEG	<userid>.EXPORT.<application name>.SENSEG
	Col 1: SENSEG Key
	Col 2: SENSEG Name
SQL	<userid>.EXPORT.<application name>.SQL
	Col 1: SQL Entity Key
	Col 2: SQL Entity Name

Entity Type	Filename
STEP	<userid>.EXPORT.<application name>.STEP
	Col 1: STEP Key
	Col 2: STEP Name
STOGROUP	<userid>.EXPORT.<application name>.STOGROUP
	Col 1: STOGROUP Key
	Col 2: STOGROUP name
SUBSCHEMA	<userid>.EXPORT.<application name>.SUBSCHEM
	Col 1: SUBSCHEMA Key
	Col 2: SUBSCHEMA Name
TABLE	<userid>.EXPORT.<application name>.TABLE
	Col 1: TABLE Key
	Col 2: TABLE Name
	Col 3: T/F: Indicates whether a CREATE was used
	Col 4: T/F: Indicates whether a READ was used
	Col 5: T/F: Indicates whether a UPDATE was used
	Col 6: T/F: Indicates whether a DELETE was used
TABLESPACE	<userid>.EXPORT.<application name>.TABLESPA
	Col 1: TABLESPACE Key
	Col 2: TABLESPACE Name
VIEW	<userid>.EXPORT.<application name>.VIEW
	Col 1: VIEW Key
	Col 2: VIEW Name

Relationship Files

These are the relationship files:

Relationship Type	Filename
AREA HAS RECORDS	<userid>.EXPORT.<application name>.ARAHRECF Col 1: AREA Key Col 2: IDMSRECORD Key
AREA IN PROGRAM	<userid>.EXPORT.<application name>.ARAINPGM Col 1: AREA Key Col 2: PROGRAM Key
AREA IN SUBSCHEMA	<userid>.EXPORT.<application name>.AINSUBSF Col 1: AREA Key Col 2: SUBSCHEMA Key
CALLS IS ENTRY	<userid>.EXPORT.<application name>.CALISENT Col 1: CALL Key Col 2: ENTRY Key
CATALOG USEDBY JOB	<userid>.EXPORT.<application name>.CATUBJOB Col 1: CATALOG Key Col 2: JOB Key
CATALOG USEDBY STEP	<userid>.EXPORT.<application name>.CATUBSTF Col 1: CATALOG Key Col 2: STEP Key
CICSDD USEDBY PROGRAM	<userid>.EXPORT.<application name>.CDDUBPGM Col 1: CICSDD Key Col 2: PROGRAM Key
CICS-TRANSACTION EXECUTED-BY PROGRAM	<userid>.EXPORT.<application name>.TRAEBPGM Col 1: CICS-TRANSACTION Key Col 2: PROGRAM Key
CICS-TRANSACTION EXECUTES PROGRAM	<userid>.EXPORT.<application name>.CTREXPGM Col 1: CICS-TRANSACTION Key Col 2: PROGRAM Key
CICS-TRANSACTION IN CSD	<userid>.EXPORT.<application name>.CTRINCSD Col 1: CICS-TRANSACTION Key Col 2: CSD Key

Relationship Type	Filename
CICS-TRANSACTION IN PCT	<userid>.EXPORT.<application name>.CTRINPCT Col 1: CICS-TRANSACTION Key Col 2: PCT Key
COLUMN IN PROGRAM	<userid>.EXPORT.<application name>.COLINPGM Col 1: COLUMN Key Col 2: PROGRAM Key
COLUMN IN TABLE	<userid>.EXPORT.<application name>.COLINTAB Col 1: COLUMN Key Col 2: TABLE Key
COLUMN IN VIEW	<userid>.EXPORT.<application name>.COLIVIEW Col 1: COLUMN Key Col 2: VIEW Key
COPYMEMBER HAS DATAITEM	<userid>.EXPORT.<application name>.INCHDDEF Col 1: COPYMEMBER Key Col 2: DATAITEM Key
COPYMEMBER IN COPYMEMBER	<userid>.EXPORT.<application name>.CPYINICP Col 1: COPYMEMBER Key Col 2: COPYMEMBER Key
COPYMEMBER IN MEMBER	<userid>.EXPORT.<application name>.COPYIMBR Col 1: COPYMEMBER Key Col 2: MEMBER Key
COPYMEMBER IN PROGRAM	<userid>.EXPORT.<application name>.CPMINPGM Col 1: COPYMEMBER Key Col 2: PROGRAM Key
COPYMEMBER INCLUDES COPYMEMBER	<userid>.EXPORT.<application name>.CPYICPYF Col 1: COPYMEMBER Key Col 2: COPYMEMBER Key
COPYMEMBER USED BY COPYMEMBER	<userid>.EXPORT.<application name>.CPYUSBCP Col 1: COPYMEMBER Key Col 2: COPYMEMBER Key
COPYMEMBER USED BY MEMBER	<userid>.EXPORT.<application name>.MBINCSRC Col 1: COPYMEMBER Col 2: MEMBER

Relationship Type	Filename
COPYMEMBER USES COPYMEMBER	<userid>.EXPORT.<application name>.CPYUSCPY Col 1: COPYMEMBER Key Col 2: COPYMEMBER Key
CSD HAS CICS-TRANSACTION	<userid>.EXPORT.<application name>.CSDHCTRF Col 1: CSD Key Col 2: CICS-TRANSACTION Key
CSD HAS DD	<userid>.EXPORT.<application name>.CSDHDDF Col 1: CSD Key Col 2: DD Key
CSECT IN LOADMODULE	<userid>.EXPORT.<application name>.CSECINLM Col 1: CSECT Key Col 2: LOADMODULE Key
CSECT IS PROGRAM	<userid>.EXPORT.<application name>.CSECTPGM Col 1: CSECT Key Col 2: PROGRAM Key
CURSOR IN DATABASE	<userid>.EXPORT.<application name>.CURINDBF Col 1: CURSOR Key Col 2: DATABASE Key
CURSOR IN PROGRAM	<userid>.EXPORT.<application name>.CSRINPGM Col 1: CURSOR Key Col 2: PROGRAM Key
DATABASE HAS CURSOR	<userid>.EXPORT.<application name>.DBHCSRF Col 1: DATABASE Key Col 2: CURSOR Key
DATABASE HAS TABLESPACE	<userid>.EXPORT.<application name>.DBHTBLSP Col 1: DATABASE Key Col 2: TABLESPACE Key
DATABASE IN PROGRAM	<userid>.EXPORT.<application name>.DBINPGMF Col 1: DATABASE Key Col 2: PROGRAM Key
DATABASE IN STOGROUP	<userid>.EXPORT.<application name>.DBINSGRP Col 1: DATABASE Key Col 2: STOGROUP Key

Relationship Type	Filename
DATAITEM DEFINED-IN INCLUDE	<userid>.EXPORT.<application name>.DTDEFINC Col 1: DATAITEM Key Col 2: INCLUDE Key
DATAITEM DEFINED-IN PROGRAM	<userid>.EXPORT.<application name>.DTDEFPGM Col 1: DATAITEM Key Col 2: PROGRAM Key
DATAITEM IN RECORD	<userid>.EXPORT.<application name>.DATINREC Col 1: DATAITEM Key Col 2: RECORD Key
DATAITEM REFERENCED-BY FIELD	<userid>.EXPORT.<application name>.DIRMAPF Col 1: DATAITEM Key Col 2: MAPSET Key
DATAITEM REFERENCED-BY MAPFIELD	<userid>.EXPORT.<application name>.DIRBMAPF Col 1: DATAITEM Key Col 2: MAPFIELD Key
DATAITEM REFERENCES MAPFIELD	<userid>.EXPORT.<application name>.DIRMAPF Col 1: DATAITEM Key Col 2: MAPFIELD Key
DBD HAS DD	<userid>.EXPORT.<application name>.DBDHSDDF Col 1: DBD Key Col 2: DD Key
DBD HAS SEGMENT	<userid>.EXPORT.<application name>.DBDHSEGF Col 1: DBD Key Col 2: SEGMENT Key
DBD IN MEMBER	<userid>.EXPORT.<application name>.DBDINMBR Col 1: DBD Key Col 2: MEMBER Key
DBD IN PCB	<userid>.EXPORT.<application name>.DBDINPCB Col 1: DBD Key Col 2: PCB Key
DD HAS DSN	<userid>.EXPORT.<application name>.DDHDSNF Col 1: DD Key Col 2: DSN Key

Relationship Type	Filename
DD IN CSD	<userid>.EXPORT.<application name>.DDINCSDF Col 1: DD Key Col 2: CSD Key
DD IN DBD	<userid>.EXPORT.<application name>.DDINDBDF Col 1: DD Key Col 2: DBD Key
DD IN FCT	<userid>.EXPORT.<application name>.DDINFCTF Col 1: DD Key Col 2: FCT Key
DD IN CICS	<userid>.EXPORT.<application name>.DDINCICS Col 1: DD Key Col 2: CICS Key
DD IN STEP	<userid>.EXPORT.<application name>.DDINSTEP Col 1: DD Key Col 2: STEP Key
DD SELECTED-BY FD	<userid>.EXPORT.<application name>.DDSELBDF Col 1: DD Key Col 2: FD Key
DSN HAS MEMBER	<userid>.EXPORT.<application name>.DSNHMBRF Col 1: DSN Key Col 2: MEMBER Key
DSN IN DD	<userid>.EXPORT.<application name>.DSNINDDF Col 1: DSN Key Col 2: DD Key
ENTRY CALLED-BY PROGRAM	<userid>.EXPORT.<application name>.ENTCBPGM Col 1: ENTRY Key Col 2: PROGRAM Key
ENTRY IN PROGRAM	<userid>.EXPORT.<application name>.ENTINPGM Col 1: ENTRY Key Col 2: PROGRAM Key
ENTRY IS CALLS	<userid>.EXPORT.<application name>.ENTISCAL Col 1: ENTRY Key Col 2: CALL Key

Relationship Type	Filename
FCT HAS DD	<userid>.EXPORT.<application name>.FCTHDDF Col 1: FCT Key Col 2: DD Key
FCT IN MEMBER	<userid>.EXPORT.<application name>.FCTINMBR Col 1: FCT Key Col 2: MEMBER Key
FCTCSD HAS DD	<userid>.EXPORT.<application name>.CICSHDDF Col 1: FCTCSD Key Col 2: DD Key
FD IN PROGRAM	<userid>.EXPORT.<application name>.FDINPGMF Col 1: FD Key Col 2: PROGRAM Key
FD SELECTS DD	<userid>.EXPORT.<application name>.FDSELDDF Col 1: FD Key Col 2: DD Key
FD USES RECORD	<userid>.EXPORT.<application name>.FDUSRECF Col 1: FD Key Col 2: RECORD Key
FORMAT EXECUTED-BY FORMAT	<userid>.EXPORT.<application name>.FMTEBFMT Col 1: FORMAT Key Col 2: FORMAT Key
FORMAT EXECUTED-BY PROGRAM	<userid>.EXPORT.<application name>.FMTEBPGM Col 1: FORMAT Key Col 2: PROGRAM Key
FORMAT EXECUTES FORMAT	<userid>.EXPORT.<application name>.FMTEXFMT Col 1: FORMAT Key Col 2: FORMAT Key
FORMAT EXECUTES IMS-TRANSACTION	<userid>.EXPORT.<application name>.FMEXITRN Col 1: FORMAT Key Col 2: IMS-TRANSACTION Key
FORMAT IN MEMBER	<userid>.EXPORT.<application name>.FMINMBRF Col 1: FORMAT Key Col 2: MEMBER Key

Relationship Type	Filename
IDMS IN PROGRAM	<userid>.EXPORT.<application name>.IDMSIPGM Col 1: IDMS Key Col 2: PROGRAM Key
IDMSMAP HAS IDMSRECORD	<userid>.EXPORT.<application name>.MAPHREC Col 1: IDMSMAP Key Col 2: IDMSRECORD Key
IDMSMAP IN PROGRAM	<userid>.EXPORT.<application name>.MAPINPGM Col 1: IDMSMAP Key Col 2: PROGRAM Key
IDMSRECORD IN AREA	<userid>.EXPORT.<application name>.RECINAF Col 1: IDMSRECORD Key Col 2: AREA Key
IDMSRECORD IN IDMSMAP	<userid>.EXPORT.<application name>.RECINMAP Col 1: IDMSRECORD Key Col 2: IDMSMAP Key
IDMSRECORD IN LOGICALRECORD	<userid>.EXPORT.<application name>.RECILREC Col 1: IDMSRECORD Key Col 2: LOGICALRECORD Key
IMSDYNALLOCMACRO IN MEMBER	<userid>.EXPORT.<application name>.MDAINMBR Col 1: IMSDYNALLOCMACRO Key Col 2: MEMBER Key
IMS-TRANSACTION EXECUTED-BY FORMAT	<userid>.EXPORT.<application name>.ITRNEBFM Col 1: IMS-TRANSACTION Key Col 2: FORMAT Key
IMS-TRANSACTION EXECUTED-BY PROGRAM	<userid>.EXPORT.<application name>.ITRNEBPG Col 1: IMS-TRANSACTION Key Col 2: PROGRAM Key
IMS-TRANSACTION EXECUTES LOADMODULE	<userid>.EXPORT.<application name>.ITRNEXLM Col 1: IMS-TRANSACTION Key Col 2: LOADMODULE Key
JOB HAS STEP	<userid>.EXPORT.<application name>.JOBHSTPF Col 1: JOB Key Col 2: STEP Key

Relationship Type	Filename
JOB IN MEMBER	<userid>.EXPORT.<application name>.JOBINMBR Col 1: JOB Key Col 2: MEMBER Key
JOB USES CATALOG	<userid>.EXPORT.<application name>.JOBUSCAT Col 1: JOB Key Col 2: CATALOG Key
JOB USES LIBRARY	<userid>.EXPORT.<application name>.JOBUSLIB Col 1: JOB Key Col 2: LIBRARY Key
LIBRARY HAS MEMBER	<userid>.EXPORT.<application name>.LIBHMBRF Col 1: LIBRARY Key Col 2: MEMBER Key
LIBRARY USED BY JOB	<userid>.EXPORT.<application name>.LIBUBJOB Col 1: LIBRARY Key Col 2: JOB Key
LIBRARY USED BY STEP	<userid>.EXPORT.<application name>.LIBUBSTF Col 1: LIBRARY Key Col 2: STEP Key
LOADMODULE EXECUTED-BY STEP	<userid>.EXPORT.<application name>.LMEXSTPF Col 1: LOADMODULE Key Col 2: STEP Key
LOADMODULE EXECUTED-BY IMS-TRANSACTION	<userid>.EXPORT.<application name>.LMEBITRN Col 1: LOADMODULE Key Col 2: IMS-TRANSACTION Key
LOADMODULE HAS CSECT	<userid>.EXPORT.<application name>.LMHCSECT Col 1: LOADMODULE Key Col 2: CSECT Key
LOADMODULE IN MEMBER	<userid>.EXPORT.<application name>.LMINMBRF Col 1: LOADMODULE Key Col 2: MEMBER Key
LOGICALRECORD HAS IDMSRECORD	<userid>.EXPORT.<application name>.LRHRECF Col 1: LOGICALRECORD Key Col 2: IDMSRECORD Key

Relationship Type	Filename
LOGICAL RECORD IN PROGRAM	<userid>.EXPORT.<application name>.LRINPGMF Col 1: LREC Key Col 2: PROGRAM Key
MAP IN MAPSET	<userid>.EXPORT.<application name>.MAPINMST Col 1: MAP Key Col 2: MAPSET Key
MAP REFERENCES RECORD	<userid>.EXPORT.<application name>.MRCRFREC Col 1: MAP Key Col 2: RECORD Key
MAP USED BY PROGRAM	<userid>.EXPORT.<application name>.MPAUBPGM Col 1: MAP Key Col 2: PROGRAM Key
MAPFIELD IN MAP	<userid>.EXPORT.<application name>.MFDINMAP Col 1: MAPFIELD Key Col 2: MAP Key
MAPFIELD REFERENCED BY DATAITEM	<userid>.EXPORT.<application name>.MAPFRFDI Col 1: MAPSET Key Col 2: DATAITEM Key
MAPSET HAS MAP	<userid>.EXPORT.<application name>.MSETHMAP Col 1: MAPSET Key Col 2: MAP Key
MAPSET IN MEMBER	<userid>.EXPORT.<application name>.MSTINMBR Col 1: MAPSET Key Col 2: MEMBER Key
MAPSET USED BY PROGRAM	<userid>.EXPORT.<application name>.MSETUPGM Col 1: MAPSET Key Col 2: PROGRAM Key
MDA HAS DD	<userid>.EXPORT.<application name>.MDAHSDDF Col 1: MDA Key Col 2: DD Key
MEMBER HAS COPYMEMBER	<userid>.EXPORT.<application name>.MBRISCIN Col 1: MEMBER Key Col 2: COPYMEMBER Key

Relationship Type	Filename
MEMBER HAS DBD	<userid>.EXPORT.<application name>.MBRHDBDF Col 1: MEMBER Key Col 2: DBD Key
MEMBER HAS FCT	<userid>.EXPORT.<application name>.MBRHFCTF Col 1: MEMBER Key Col 2: FCT Key
MEMBER HAS FORMAT	<userid>.EXPORT.<application name>.MBRHFMTF Col 1: MEMBER Key Col 2: FORMAT Key
MEMBER HAS IMSDYNALLOCMACRO	<userid>.EXPORT.<application name>.MBRHMDA Col 1: MEMBER Key Col 2: IMSDYNALLOCMACRO Key
MEMBER HAS JOB	<userid>.EXPORT.<application name>.MBRHJOBF Col 1: MEMBER Key Col 2: JOB Key
MEMBER HAS LOADMODULE	<userid>.EXPORT.<application name>.MBRHLMF Col 1: MEMBER Key Col 2: LOADMODULE Key
MEMBER HAS MAPSET	<userid>.EXPORT.<application name>.MBRHMSET Col 1: MEMBER Key Col 2: MAPSET Key
MEMBER HAS PCT	<userid>.EXPORT.<application name>.MBRHPCTF Col 1: MEMBER Key Col 2: PCT Key
MEMBER HAS PPT	<userid>.EXPORT.<application name>.MBRHPPTF Col 1: MEMBER Key Col 2: PPT Key
MEMBER HAS PROC	<userid>.EXPORT.<application name>.MBRHPROC Col 1: MEMBER Key Col 2: PROC Key
MEMBER HAS PROGRAM	<userid>.EXPORT.<application name>.MBRHPGMF Col 1: MEMBER Key Col 2: PROGRAM Key

Relationship Type	Filename
MEMBER HAS PSB	<userid>.EXPORT.<application name>.MBRHPSBF Col 1: MEMBER Key Col 2: PSB Key
MEMBER IN DSN	<userid>.EXPORT.<application name>.MBRINDSN Col 1: MEMBER Key Col 2: DSN Key
MEMBER IN LIBRARY	<userid>.EXPORT.<application name>.MBRINLIB Col 1: MEMBER Key Col 2: LIBRARY Key
MEMBER USES COPYMEMBER	<userid>.EXPORT.<application name>.SRCINCMB Col 1: MEMBER Key Col 2: COPYMEMBER Key
PCB HAS SENSEG	<userid>.EXPORT.<application name>.PCBHSSGF Col 1: PCB Key Col 2: SENSEG Key
PCB IN PSB	<userid>.EXPORT.<application name>.PCBINPSB Col 1: PCB Key Col 2: PSB Key
PCB REFERENCES DBD	<userid>.EXPORT.<application name>.PCBHDBDF Col 1: PCB Key Col 2: DBD Key
PCT HAS CICS-TRANSACTION	<userid>.EXPORT.<application name>.PCTHCTRF Col 1: PCT Key Col 2: CICS-TRANSACTION Key
PCT IN MEMBER	<userid>.EXPORT.<application name>.PCTINMBR Col 1: PCT Key Col 2: MEMBER Key
PPT IN MEMBER	<userid>.EXPORT.<application name>.PPTINMBR Col 1: PPT Key Col 2: MEMBER Key
PROC HAS STEP	<userid>.EXPORT.<application name>.PRCHSTPF Col 1: PROC Key Col 2: STEP Key

Relationship Type	Filename
PROC EXECUTED-BY STEP	<userid>.EXPORT.<application name>.PROCEBST Col 1: PROC Key Col 2: STEP Key
PROC IN MEMBER	<userid>.EXPORT.<application name>.PRCINMBR Col 1: PROC Key Col 2: MEMBER Key
PROGRAM CALLS ENTRY	<userid>.EXPORT.<application name>.PGMCENTF Col 1: PROGRAM Key Col 2: ENTRY Key
PROGRAM EXECUTED-BY CICS-TRANSACTION	<userid>.EXPORT.<application name>.PGMEBCTR Col 1: PROGRAM Key Col 2: CICS-TRANSACTION Key
PROGRAM EXECUTES CICS-TRANSACTION	<userid>.EXPORT.<application name>.PGMEXTRN Col 1: PROGRAM Key Col 2: CICS-TRANSACTION Key
PROGRAM EXECUTES FORMAT	<userid>.EXPORT.<application name>.PGMEXFMT Col 1: PROGRAM Key Col 2: FORMAT Key
PROGRAM EXECUTES IMS-TRANSACTION	<userid>.EXPORT.<application name>.PGMEXITR Col 1: PROGRAM Key Col 2: IMS-TRANSACTION Key
PROGRAM HAS AREA	<userid>.EXPORT.<application name>.PGMHAREA Col 1: PROGRAM Key Col 2: AREA Key
PROGRAM HAS COLUMN	<userid>.EXPORT.<application name>.PGHHCOLF Col 1: PROGRAM Key Col 2: COLUMN Key
PROGRAM HAS CURSOR	<userid>.EXPORT.<application name>.PGMHCSRFB Col 1: PROGRAM Key Col 2: CURSOR Key
PROGRAM HAS DATABASE	<userid>.EXPORT.<application name>.PGMHDBFB Col 1: PROGRAM Key Col 2: DATABASE Key

Relationship Type	Filename
PROGRAM HAS DATAITEM	<userid>.EXPORT.<application name>.PGMHDEF Col 1: PROGRAM Key Col 2: DATAITEM Key
PROGRAM HAS ENTRY	<userid>.EXPORT.<application name>.PGMHENTF Col 1: PROGRAM Key Col 2: ENTRY Key
PROGRAM HAS FD	<userid>.EXPORT.<application name>.PGMHFDF Col 1: PROGRAM Key Col 2: FD Key
PROGRAM HAS IDMS	<userid>.EXPORT.<application name>.PGMHIDMS Col 1: PROGRAM Key Col 2: IDMS Key
PROGRAM HAS IDSMAP	<userid>.EXPORT.<application name>.PGMHMAP Col 1: PROGRAM Key Col 2: IDSMAP Key
PROGRAM HAS IDMSRECORD	<userid>.EXPORT.<application name>.PGMHRECS Col 1: PROGRAM Key Col 2: IDMSRECORD Key
PROGRAM HAS LOGICALRECORD	<userid>.EXPORT.<application name>.PGMHLREC Col 1: PROGRAM Key Col 2: LOGICALRECORD Key
PROGRAM HAS RECORD	<userid>.EXPORT.<application name>.PGMRHREC Col 1: PROGRAM Key Col 2: RECORD Key
PROGRAM HAS SCHEMA	<userid>.EXPORT.<application name>.PGMHSCHF Col 1: PROGRAM Key Col 2: SCHEMA Key
PROGRAM HAS SQL	<userid>.EXPORT.<application name>.PGMHSQFLF Col 1: PROGRAM Key Col 2: SQL Entity Key
PROGRAM HAS STOGROUP	<userid>.EXPORT.<application name>.PGMHSTGF Col 1: PROGRAM Key Col 2: STOGROUP Key

Relationship Type	Filename
PROGRAM HAS SUBSCHEMA	<userid>.EXPORT.<application name>.PGMHSSCH Col 1: PROGRAM Key Col 2: SUBSCHEMA Key
PROGRAM HAS TABLE	<userid>.EXPORT.<application name>.PGMHTBLF Col 1: PROGRAM Key Col 2: TABLE Key
PROGRAM HAS TABLESPACE	<userid>.EXPORT.<application name>.PGMHTBLS Col 1: PROGRAM Key Col 2: TABLESPACE Key
PROGRAM HAS VIEW	<userid>.EXPORT.<application name>.PGMHVIEW Col 1: PROGRAM Key Col 2: VIEW Key
PROGRAM IN MEMBER	<userid>.EXPORT.<application name>.PGMINMBR Col 1: PROGRAM Key Col 2: MEMBER Key
PROGRAM INCLUDES COPYMEMBER	<userid>.EXPORT.<application name>.PGMILMBR Col 1: PROGRAM Key Col 2: COPYMEMBER Key
PROGRAM IS CSECT	<userid>.EXPORT.<application name>.PGMISCSC Col 1: PROGRAM Key Col 2: CSECT Key
PROGRAM LINKEDBY PROGRAM	<userid>.EXPORT.<application name>.PGMLBPGM Col 1: PROGRAM Key Col 2: PROGRAM Key
PROGRAM LINKS PROGRAM	<userid>.EXPORT.<application name>.PGMLKPGM Col 1: PROGRAM Key Col 2: LINKED PROGRAM Key
PROGRAM USES DD	<userid>.EXPORT.<application name>.PGMUSCDD Col 1: PROGRAM Key Col 2: DD Key
PROGRAM USES MAP	<userid>.EXPORT.<application name>.PGMUSMAP Col 1: PROGRAM Key Col 2: MAP Key

Relationship Type	Filename
PROGRAM USES MAPSET	<userid>.EXPORT.<application name>.PGMUSMST Col 1: PROGRAM Key Col 2: MAPSET Key
PROGRAM USES PSB	<userid>.EXPORT.<application name>.PGMUSPSB Col 1: PROGRAM Key Col 2: PSB Key
PROGRAM USES SEGMENT	<userid>.EXPORT.<application name>.PGMUSEGF Col 1: PROGRAM Key Col 2: SEGMENT Key
PROGRAM USES SPECIALREGISTER	<userid>.EXPORT.<application name>.PGMUSREG Col 1: PROGRAM Key Col 2: SPECIALREGISTER Key
PSB HAS PCB	<userid>.EXPORT.<application name>.PSBHPCBF Col 1: PSB Key Col 2: PCB Key
PSB IN MEMBER	<userid>.EXPORT.<application name>.PSBINMBR Col 1: PSB Key Col 2: MEMBER Key
PSB USED BY PROGRAM	<userid>.EXPORT.<application name>.PSBUBPGM Col 1: PSB Key Col 2: PROGRAM Key
RECORD HAS DATAITEM	<userid>.EXPORT.<application name>.RECHDATA Col 1: RECORD Key Col 2: DATAITEM Key
RECORD IN PROGRAM	<userid>.EXPORT.<application name>.RECINPGM Col 1: RECORD Key Col 2: PROGRAM Key
RECORD REFERENCED BY MAP	<userid>.EXPORT.<application name>.RECRBMRC Col 1: RECORD Key Col 2: MAP Key
RECORD USED BY FD	<userid>.EXPORT.<application name>.RECUBDFD Col 1: RECORD Key Col 2: FD Key

Relationship Type	Filename
SCHEMA HAS SUBSCHEMA	<userid>.EXPORT.<application name>.SCMHSSCM Col 1: SCHEMA Key Col 2: SUBSCHEMA Key
SCHEMA IN PROGRAM	<userid>.EXPORT.<application name>.SCHINPGM Col 1: SCHEMA Key Col 2: PROGRAM Key
SEGFIELD IN SEGMENT	<userid>.EXPORT.<application name>.FLDINSEG Col 1: SEGFIELD Key Col 2: SEGMENT Key
SEGMENT HAS SEGFIELD	<userid>.EXPORT.<application name>.SEGHFLDF Col 1: SEGMENT Key Col 2: SEGFIELD Key
SEGMENT IN DBD	<userid>.EXPORT.<application name>.SEGINDBD Col 1: SEGMENT Key Col 2: DBD Key
SEGMENT IN SEGMENT	<userid>.EXPORT.<application name>.SEGINSEG Col 1: SEGMENT Key Col 2: SEGMENT Key
SEGMENT USED BY PROGRAM	<userid>.EXPORT.<application name>.SEGUBPGM Col 1: SEGMENT Key Col 2: PROGRAM Key
SEGMENT USES SEGMENT	<userid>.EXPORT.<application name>.SEGHSEGF Col 1: SEGMENT Key Col 2: SEGMENT Key
SENFIELD IN SENSEG	<userid>.EXPORT.<application name>.SENFISEN Col 1: SENFIELD Key Col 2: SENSEG Key
SENSEG HAS SENFIELD	<userid>.EXPORT.<application name>.SSGHSFDF Col 1: SENSEG Key Col 2: SENFIELD Key
SENSEG IN PCB	<userid>.EXPORT.<application name>.SSEGIPCB Col 1: SENSEG Key Col 2: PCB Key

Relationship Type	Filename
SENSEG IN SENSEG	<userid>.EXPORT.<application name>.SENINSEN Col 1: SENSEG Key Col 2: SENSEG Key
SENSEG USES SENSEG	<userid>.EXPORT.<application name>.SSGHSSGF Col 1: SENSEG Key Col 2: SENSEG Key
SQL IN PROGRAM	<userid>.EXPORT.<application name>.SQLINPGM Col 1: SQL Entity Key Col 2: PROGRAM Key
STEP EXECUTES LOADMODULE	<userid>.EXPORT.<application name>.STEPELMF Col 1: STEP Key Col 2: LOADMODULE Key
STEP EXECUTES PROC	<userid>.EXPORT.<application name>.STPEXPRC Col 1: STEP Key Col 2: PROC Key
STEP HAS DD	<userid>.EXPORT.<application name>.STPHDDF Col 1: STEP Key Col 2: DD Key
STEP IN JOB	<userid>.EXPORT.<application name>.STPEIJOB Col 1: STEP Key Col 2: JOB Key
STEP IN PROC	<userid>.EXPORT.<application name>.STINPROC Col 1: STEP Key Col 2: PROC Key
STEP USES CATALOG	<userid>.EXPORT.<application name>.STPUSCAT Col 1: STEP Key Col 2: CATALOG Key
STEP USES LIBRARY	<userid>.EXPORT.<application name>.STPUSLIB Col 1: STEP Key Col 2: LIBRARY Key
STOGROUP HAS DATABASE	<userid>.EXPORT.<application name>.STGHDBF Col 1: STOGROUP Key Col 2: DATABASE Key

Relationship Type	Filename
STOGROUP IN PROGRAM	<userid>.EXPORT.<application name>.STGINPGM Col 1: STOGROUP Key Col 2: PROGRAM Key
SUBSCHEMA HAS AREA	<userid>.EXPORT.<application name>.SSCMHARA Col 1: SUBSCHEMA Key Col 2: AREA Key
SUBSCHEMA HAS LOGICALRECORD	<userid>.EXPORT.<application name>.SSCMHLRF Col 1: SUBSCHEMA Key Col 2: LOGICALRECORD Key
SUBSCHEMA IN PROGRAM	<userid>.EXPORT.<application name>.SSCMINPG Col 1: SUBSCHEMA Key Col 2: PROGRAM Key
SUBSCHEMA IN SCHEMA	<userid>.EXPORT.<application name>.SUBSISCM Col 1: SUBSCHEMA Key Col 2: SCHEMA Key
TABLE HAS COLUMN	<userid>.EXPORT.<application name>.TBHCOLF Col 1: TABLE Key Col 2: COLUMN Key
TABLE IN PROGRAM	<userid>.EXPORT.<application name>.TBLINPGM Col 1: TABLE Key Col 2: PROGRAM Key
TABLE IN TABLESPACE	<userid>.EXPORT.<application name>.TABINTSF Col 1: TABLE Key Col 2: TABLESPACE Key
TABLESPACE HAS TABLE	<userid>.EXPORT.<application name>.TBLSPHTB Col 1: TABLESPACE Key Col 2: TABLE Key
TABLESPACE HAS VIEW	<userid>.EXPORT.<application name>.TBLSPHVW Col 1: TABLESPACE Key Col 2: VIEW Key
TABLESPACE IN DATABASE	<userid>.EXPORT.<application name>.TSINDBF Col 1: TABLESPACE Key Col 2: DATABASE Key

Relationship Type	Filename
TABLESPACE IN PROGRAM	<userid>.EXPORT.<application name>.TBSINPGM Col 1: TABLESPACE Key Col 2: PROGRAM Key
VIEW HAS COLUMN	<userid>.EXPORT.<application name>.VWHCOLF Col 1: VIEW Key Col 2: COLUMN Key
VIEW IN PROGRAM	<userid>.EXPORT.<application name>.VWINPGMF Col 1: VIEW Key Col 2: PROGRAM Key
VIEW IN TABLESPACE	<userid>.EXPORT.<application name>.VIEWINTS Col 1: VIEW Key Col 2: TABLESPACE Key

Defining and Loading DB2 Databases

The Export facility provides for loading output sequential files, created from the unload job, to a DB2 database.

To define and load a DB2 database

- 1 Use the JCL VIABXDFJ to define the database.
- 2 To load the sequential files from the unload step (in the AKR export job), use the JCL VIABXLDJ.

DB2 Database Description**Entity Tables**

Table Name	Contents		
ASG.AREA	Col 1:	Area_Key	Char(8)
	Col 2:	Area_Name	Char(8)
	Primary Key:	Area_Key	
ASG.CATALOG	Col 1:	Catalog_Key	Char(8)
	Col 2:	Catalog_Name	Char(44)
	Primary Key:	Catalog_Key	
ASG.CICSTXN (CICS Trans)	Col 1:	CICSTXN_Key	Char(8)
	Col 2:	CICSTXN_Name	Char(8)
	Primary Key:	CICSTXN_Key	
ASG.COLUMN	Col 1:	Column_Key	Char(8)
	Col 2:	Column_Name	Char(30)
	Primary Key:	Column_Key	
ASG.COPYMEMBER	Col 1:	Copymember_Key	Char(8)
	Col 2:	Copymember_Name	Char(10)
	Primary Key:	Copymember_Key	
ASG.CSD	Col 1:	CSD_Key	Char(8)
	Col 2:	CSD_Name	Char(44)
	Primary Key:	CSD_Key	
ASG.CSECT	Col 1:	CSECT_Key	Char(8)
	Col 2:	CSECT_Name	Char(8)
	Primary Key:	CSECT_Key	

Table Name	Contents		
ASG.CURSOR	Col 1:	Cursor_Key	Char(8)
	Col 2:	Cursor_Name	Char(30)
	Primary Key:	Cursor_Key	
ASG.DATABASE	Col 1:	Database_Key	Char(8)
	Col 2:	Database_Name	Char(44)
	Primary Key:	Database_Key	
ASG.DATAITEM	Col 1:	Data_Key	Char(8) Not Null
	Col 2:	Data_Name	Char(50) Not Null
	Col 3:	Data_Level	Char(2)
	Col 4:	Data_Parent	Char(8)
	Col 5:	Data_Child	Char(8)
	Col 6:	Data_Next_Sibling	Char(8)
	Col 7:	Data_Prev_Sibling	Char(8)
	Col 8:	Data_Picture	Char(20)
	Col 9:	Data_Usage	Char(10)
	Col 10:	Data_Occurs	Char(8)
	Col 11:	Data_Redefine	Char(8)
	Col 12:	Data_Rename_From	Char(8)
	Col 13:	Data_Rename_Thru	Char(8)
	Col 14:	Data_Size	Char(8)
	Col 15:	Data_Offset	Char(8)
	Col 16:	Data_Position	Char(8)
	Col 17:	Data_Value	Char(40)
	Col 18:	Data_Use	Char(8)

Table Name	Contents		
	Col 19:	Data_Mod	Char(8)
	Col 20:	Data_Ref	Char(8)
	Col 21:	Data_Cond	Char(8)
	Col 22:	Data_Arith	Char(8)
	Col 23:	Data_Create	Char(1)
	Col 24:	Data_Read	Char(1)
	Col 25:	Data_Update	Char(1)
	Col 26:	Data_Delete	Char(1)
	Primary Key:	Data_Key	
ASG.DBD	Col 1:	DBD_Key	Char(8)
	Col 2:	DBD_Name	Char(8)
	Primary Key:	DBD_Key	
ASG.DD	Col 1:	DD_Key	Char(8)
	Col 2:	DD_Name	Char(8)
	Primary Key:	DD_Key	
ASG.DSN	Col 1:	DSN_Key	Char(8)
	Col 2:	DSN_Name	Char(44)
	Primary Key:	DSN_Key	
ASG.ENTRY	Col 1:	Entry_Key	Char(8)
	Col 2:	Entry_Name	Char(10)
	Primary Key:	Entry_key	
ASG.FCT	Col 1:	FCT_Key	Char(8)
	Col 2:	FCT_Name	Char(8)
	Primary Key:	FCT_Key	

Table Name	Contents		
ASG.FCTCSD	Col 1:	FCTCSD_Key	Char(8)
	Col 2:	FCTCSD_Name	Char(44)
	Primary Key:	FCTCSD_Key	
ASG.FD	Col 1:	FD_Key	Char(8)
	Col 2:	FD_Name	Char(30)
	Primary Key:	FD_Key	
ASG.FORMAT	Col 1:	Format_Key	Char(8)
	Col 2:	Format_Name	Char(8)
	Primary Key:	Format_Key	
ASG.IDMS	Col 1:	IDMS_Key	Char(8)
	Col 2:	IDMS_Name	Char(36)
	Primary Key:	IDMS_Key	
ASG.IDSMAP	Col 1:	IDSMAP_Key	Char(8)
	Col 2:	IDSMAP_Name	Char(8)
	Primary Key:	IDSMAP_Key	Char(8)
ASG.IDMSRECORD	Col 1:	IDMSRECORD_Key	Char(30)
	Col 2:	IDMSRECORD_Name	
	Primary Key:	IDMSRECORD_Key	
ASG.IMSDYNALLOCMACRO	Col 1:	IMSALCMAC_Key	Char(8)
	Col 2:	IMSALCMAC_Name	Char(8)
	Primary Key:	IMSALCMAC_Key	
ASG.IMSTXN (IMS Transaction)	Col 1:	IMSTXN_Key	Char(8)
	Col 2:	IMSTXN_Name	Char(8)
	Primary Key:	IMSTXN_Key	

Table Name	Contents		
ASG.JOB	Col 1:	Job_Key	Char(8)
	Col 2:	Job_Name	Char(8)
	Primary Key:	Job_Key	
ASG.LIBRARY	Col 1:	Library_Key	Char(8)
	Col 2:	Library_Name	Char(44)
	Primary Key:	Library_Key	
ASG.LOADMODULE	Col 1:	LoadModule_Key	Char(8)
	Col 2:	LoadModule_Name	Char(8)
	Primary Key:	LoadModule_Key	
ASG.LOGICAL RECORD	Col 1:	LOGICALRECORD_Key	Char(8)
	Col 2:	LOGICALRECORD_Name	Char(30)
	Primary Key:	LOGICALRECORD_Key	
ASG.MAP	Col 1:	Map_Key	Char(8)
	Col 2:	Map_Name	Char(8)
	Primary Key:	Map_Key	
ASG.MAPFIELD	Col 1:	Mapfield_Key	Char(8)
	Col 2:	Mapfield_Name	Char(30)
	Primary Key:	Mapfield_Key	
ASG.MAPSET	Col 1:	Mapset_Key	Char(8)
	Col 2:	Mapset_Name	Char(8)
	Primary Key:	Mapset_Key	
ASG.MEMBER	Col 1:	Member_Key	Char(8)
	Col 2:	Member_Name	Char(10)
	Primary Key:	Member_Key	

Table Name	Contents		
ASG.PCB	Col 1:	PCB_Key	Char(8)
	Col 2:	PCB_Name	Char(8)
	Primary Key:	PCB_Key	
ASG.PCT	Col 1:	PCT_Key	Char(8)
	Col 2:	PCT_Name	Char(8)
	Primary Key:	PCT_Key	
ASG.PPT	Col 1:	PPT_Key	Char(8)
	Col 2:	PPT_Name	Char(8)
	Primary Key:	PPT_Key	
ASG.PROC	Col 1:	Proc_Key	Char(8)
	Col 2:	Proc_Name	Char(8)
	Primary Key:	Proc_Key	
ASG.PROGRAM	Col 1:	Program_Key	Char(8)
	Col 2:	Program_Name	Char(8)
	Primary Key:	Program_Key	
ASG.PSB	Col 1:	PSB_Key	Char(8)
	Col 2:	PSB_Name	Char(8)
	Primary Key:	PSB_Key	
ASG.RECORD	Col 1:	Rec_Key	Char(8)
	Col 2:	Rec_Name	Char(30)
	Primary Key:	Rec_Key	
ASG.SCHEMA	Col 1:	Schema_Key	Char(8)
	Col 2:	Schema_Name	Char(8)
	Primary Key:	Schema_Key	

Table Name	Contents		
ASG.SEGFIELD	Col 1:	SEGFIELD_Key	Char(8)
	Col 2:	SEGFIELD_Name	Char(30)
	Primary Key:	SEGFIELD_Key	
ASG.SEGMENT	Col 1:	Segment_Key	Char(8)
	Col 2:	Segment_Name	Char(30)
	Primary Key:	Segment_Key	
ASG.SENFIELD	Col 1:	SENFIELD_Key	Char(8)
	Col 2:	SENFIELD_Name	Char(30)
	Primary Key:	SENFIELD_Key	
ASG.SENSEG	Col 1:	SENSEG_Key	Char(8)
	Col 2:	SENSEG_Name	Char(30)
	Primary Key:	SENSEG_Key	
ASG.SQL	Col 1:	SQL_Key	Char(8)
	Col 2:	SQL_Name	Char(30)
	Primary Key:	SQL_Key	
ASG.STEP	Col 1:	Step_Key	Char(8)
	Col 2:	Step_Name	Char(8)
	Primary Key:	Step_Key	
ASG.STOGROUP (Storage Group)	Col 1:	STOGROUP_Key	Char(8)
	Col 2:	STOGROUP_Name	Char(8)
	Primary Key:	STOGROUP_Key	
ASG.SUBSCHEMA	Col 1:	Subschema_Key	Char(8)
	Col 2:	Subschema_Name	Char(8)
	Primary Key:	Subschema_Key	

Table Name	Contents		
ASG.TABLE	Col 1:	Table_Key	Char(8)
	Col 2:	Table_Name	Char(30)
	Primary Key:	Table_Key	
ASG.TABLESPACE	Col 1:	TableSpace_Key	Char(8)
	Col 2:	TableSpace_Name	Char(8)
	Primary Key:	TableSpace_Key	
ASG.VIEW	Col 1:	View_Key	Char(8)
	Col 2:	View_Name	Char(30)
	Primary Key:	View_Key	

Relationship Tables

Table name	Contents		
ASG.AREA_HAS_RECORDS	Col 1:	AREA_Key	Char(8)
	Col 2:	RECORDS_Key	Char(8)
	Primary Key:	(AREA_Key, RECORDS_Key)	
	Foreign Key:	AREAKEY (AREA_Key)	
	References:	ASG.AREA	
	Foreign Key:	RECRDKEY (RECORDS_Key)	
ASG.AREAS_IN_PGM	Col 1:	AREA_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(AREA_Key, PGM_Key)	
	Foreign Key:	AREAKEY (AREA_Key)	
	References:	ASG.AREA	

Table name	Contents		
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.AREAS_IN_SUBSCHEMA	Col 1:	AREAS_Key	Char(8)
	Col 2:	SUBSCHEMA_Key	Char(8)
	Primary Key:	(AREAS_Key, SUBSCHEMA_Key)	
	Foreign Key:	AREASKEY (AREAS_Key)	
	References:	ASG.AREA	
	Foreign Key:	SUBSCKEY (SUBSCHEMA_Key)	
	References:	ASG.SUBSCHEMA	
ASG.CATALOG_USDBY_JOB	Col 1:	CATALOG_Key	Char(8)
	Col 2:	JOB_Key	Char(8)
	Primary Key:	(CATALOG_Key, JOB_Key)	
	Foreign Key:	CATKEY (CATALOG_Key)	
	References:	ASG.CATALOG	
	Foreign Key:	JOBKEY (JOB_Key)	
	References:	ASG.JOB	
ASG.CATALOG_USDBY_STP	Col 1:	CATALOG_Key	Char(8)
	Col 2:	STEP_Key	Char(8)
	Primary Key:	(CATALOG_Key, STEP_Key)	
	Foreign Key:	CATKEY (CATALOG_Key)	
	References:	ASG.CATALOG	
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	

Table name	Contents		
ASG.CICS_HAS_DD	Col 1:	CICS_Key	Char(8)
	Col 2:	DD_Key	Char(8)
	Primary Key:	(CICS_Key, DD_Key)	
	Foreign Key:	CICSKEY (CICS_Key)	
	References:	ASG.FCTCSD	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
ASG.CICSDD_USDBY_PGM	Col 1:	CICSDD_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(CICSDD_Key, PGM_Key)	
	Foreign Key:	CIDDKEY (CICSDD_Key)	
	References:	ASG.DD	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.CICSTRANS_EXEC_PGM	Col 1:	CICSTRANS_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(CICSTRANS_Key, PGM_Key)	
	Foreign Key:	CITXNKEY (CICSTRANS_Key)	
	References:	ASG.CICSTXN	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.CICSTRANS_IN_CSD	Col 1:	CICSTRANS_Key	Char(8)
	Col 2:	CSD_Key	Char(8)
	Primary Key:	(CICSTRANS_Key, CSD_Key)	

Table name	Contents		
	Foreign Key:	CTXNKEY (CICSTRANS_Key)	
	References:	ASG.CICSTXN	
	Foreign Key:	PSDKEY (CSD_Key)	
	References:	ASG.CSD	
ASG.CICSTRANS_IN_PCT	Col 1:	CICSTRANS_Key	Char(8)
	Col 2:	PCT_Key	Char(8)
	Primary Key:	(CICSTRANS_Key, PCT_Key)	
	Foreign Key:	CTXNKEY (CICSTRANS_Key)	
	References:	ASG.CICSTXN	
	Foreign Key:	PCTKEY (PCT_Key)	
	References:	ASG.PCT	
ASG.COLUMNS_IN_PGM	Col 1:	COLUMN_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(COLUMN_Key, PGM_Key)	
	Foreign Key:	COLKEY (COLUMN_Key)	
	References:	ASG.COLUMN	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.COLUMNS_IN_TABLE	Col 1:	COLUMNS_Key	Char(8)
	Col 2:	TABLE_Key	Char(8)
	Primary Key:	(COLUMNS_Key, TABLE_Key)	
	Foreign Key:	COLUMKEY (COLUMNS_Key)	
	References:	ASG.COLUMN	

Table name	Contents		
	Foreign Key:	TABLEKEY (TABLE_Key)	
	References:	ASG.TABLE	
ASG.COLUMNS_IN_VIEW	Col 1:	COLUMNS_Key	Char(8)
	Col 2:	VIEW_Key	Char(8)
	Primary Key:	(COLUMNS_Key, VIEW_Key)	
	Foreign Key:	COLUMKEY (COLUMNS_Key)	
	References:	ASG.COLUMN	
	Foreign Key:	VIEWKEY (VIEW_Key)	
	References:	ASG.VIEW	
ASG.COPY_INCLIN_MEMBER	Col 1:	COPY_Key	Char(8)
	Col 2:	MEMBER_Key	Char(8)
	Primary Key:	(COPY_Key, MEMBER_Key)	
	Foreign Key:	COPYKEY COPY_Key)	
	References:	ASG.COPYMEMBER	
	Foreign Key:	MEMBRKEY (MEMBER_Key)	
	References:	ASG.MEMBER	
ASG.COPY_INCLIN_PGM	Col 1:	COPY_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(COPY_Key, PGM_Key)	
	Foreign Key:	COPYKEY (COPY_Key)	
	References:	ASG.COPYMEMBER	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	

Table name	Contents		
ASG.COPYMB_INCLIN_COPYMB	Col 1:	COPYMBR_Key	Char(8)
	Col 2:	COPYMBR2_Key	Char(8)
	Primary Key:	(COPYMBR_Key, COPYMBR2_Key)	
	Foreign Key:	CPMBKEY (COPYMBR_Key)	
	References:	ASG.COPYMEMBER	
	Foreign Key:	CPMB2KEY (COPYMBR2_Key)	
	References:	ASG.COPYMEMBER	
ASG.COPYMB_USES_COPYMB	Col 1:	COPYMBR_Key	Char(8)
	Col 2:	COPYMBR2_Key	Char(8)
	Primary Key:	(COPYMBR_Key, COPYMBR2_Key)	
	Foreign Key:	CPMBKEY (COPYMBR_Key)	
	References:	ASG.COPYMEMBER	
	Foreign Key:	CPMB2KEY (COPYMBR2_Key)	
	References:	ASG.COPYMEMBER	
ASG.CSD_HAS_CICSTRANS	Col 1:	CSD_Key	Char(8)
	Col 2:	CICSTRANS_Key	Char(8)
	Primary Key:	(CSD_Key, CICSTRANS_Key)	
	Foreign Key:	CSDKEY (CSD_Key)	
	References:	ASG.CSD	
	Foreign Key:	CTXNKEY (CICSTRANS_Key)	
	References:	ASG.CICSTXN	
ASG.CSD_HAS_DD	Col 1:	CSD_Key	Char(8)
	Col 2:	DD_Key	Char(8)
	Primary Key:	(CSD_Key, DD_Key)	

Table name	Contents		
	Foreign Key:	CSDKEY (CSD_Key)	
	References:	ASG.CSD	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
ASG.CSECT_IN_LM	Col 1:	CSECT_Key	Char(8)
	Col 2:	LM_Key	Char(8)
	Primary Key:	(CSECT_Key, LM_Key)	
	Foreign Key:	CSECTKEY (CSECT_Key)	
	References:	ASG.CSECT	
	Foreign Key:	LMKEY (LM_Key)	
	References:	ASG.LOADMODULE	
ASG.CSECT_IS_PGM	Col 1:	CSECT_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(CSECT_Key, PGM_Key)	
	Foreign Key:	CSECTKEY (CSECT_Key)	
	References:	ASG.CSECT	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.CURSOR_IN_DATABASE	Col 1:	CURSOR_Key	Char(8)
	Col 2:	DATABASE_Key	Char(8)
	Primary Key:	(CURSOR_Key, DATABASE_Key)	
	Foreign Key:	CURSRKEY (CURSOR_Key)	
	References:	ASG.CURSOR	

Table name	Contents		
	Foreign Key:	DBASEKEY (DATABASE_Key)	
	References:	ASG.DATABASE	
ASG.CURSOR_IN_PGM	Col 1:	CURSOR_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(CURSOR_Key,PGM_Key)	
	Foreign Key:	CURSRKEY(CURSOR_Key)	
	References:	ASG.CURSOR	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.DATA_DEFIN_INCL	Col 1:	DATA_Key	Char(8)
	Col 2:	INCL_Key	Char(8)
	Primary Key:	(DATA_Key, INCL_Key)	
	Foreign Key:	DATAKEY (INCL_Key)	
	References:	ASG.DATAITEM	
	Foreign Key:	INCLKEY (INCL_Key)	
	References:	ASG.COPYMEMBER	
ASG.DATA_DEFIN_PGM	Col 1:	DATA_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(DATA_Key, PGM_Key)	
	Foreign Key:	DATAKEY (DATA_Key)	
	References:	ASG.DATAITEM	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	

Table name	Contents		
ASG.DATA_HAS_HOMONYM	Col 1:	DATA_Key	Char(8)
	Col 2:	HOMONYM_Key	Char(8)
	Primary Key:	(DATA_Key, HOMONYM_Key)	
	Foreign Key:	DATAKEY (DATA_Key)	
	References:	ASG.DATAITEM	
	Foreign Key:	HOMKEY (HOMONYM_Key)	
	References:	ASG.DATAITEM	
ASG.DATA_HAS_SYNONYM	Col 1:	DATA_Key	Char(8)
	Col 2:	SYNONYM_Key	Char(8)
	Primary Key:	(DATA_Key, SYNONYM_Key)	
	Foreign Key:	DATAKEY (DATA_Key)	
	References:	ASG.DATAITEM	
	Foreign Key:	SYNKEY (SYNONYM_Key)	
	References:	ASG.DATAITEM	
ASG.DATA_IN_RECORD	Col 1:	DATA_Key	Char(8)
	Col 2:	RECORD_Key	Char(8)
	Primary Key:	(DATA_Key, RECORD_Key)	
	Foreign Key:	DATAKEY (DATA_Key)	
	References:	ASG.DATAITEM	
	Foreign Key:	RECKEY (RECORD_Key)	
	References:	ASG.RECORD	
ASG.DATABASE_HAS_CRSR	Col 1:	DATABASE_Key	Char(8)
	Col 2:	CURSOR_Key	Char(8)
	Primary Key:	(DATABASE_Key, CURSOR_Key)	

Table name	Contents		
	Foreign Key:	DBASEKEY (DATABASE_Key)	
	References:	ASG.DATABASE	
	Foreign Key:	CURSRKEY (CURSOR_Key)	
	References:	ASG.CURSOR	
ASG.DATABASE_HAS_TBLSP	Col 1:	DATABASE_Key	Char(8)
	Col 2:	TBLSP_Key	Char(8)
	Primary Key:	(DATABASE_Key, TBLSP_Key)	
	Foreign Key:	DBASEKEY (DATABASE_Key)	
	References:	ASG.DATABASE	
	Foreign Key:	TBLSPKEY (TBLSP_Key)	
	References:	ASG.TABLESPACE	
ASG.DATABASE_IN_PGM	Col 1:	DATABASE_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(DATABASE_Key, PGM_Key)	
	Foreign Key:	DBKEY (DATABASE_Key)	
	References:	ASG.DATABASE	
	Foreign Key:	PGMPKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.DATABASE_IN_STOGRP	Col 1:	DATABASE_Key	Char(8)
	Col 2:	STOGRP_Key	Char(8)
	Primary Key:	(DATABASE_Key,STOGRP_Key)	
	Foreign Key:	DBASEKEY (DATABASE_Key)	
	References:	ASG.DATABASE	

Table name	Contents		
	Foreign Key:	STOGPKEY (STOGRP_Key)	
	References	ASG.STOGRUP	
ASG.DBD_HAS_DD	Col 1:	DBD_Key	Char(8)
	Col 2:	DD_Key	Char(8)
	Primary Key:	(DBD_Key, DD_Key)	
	Foreign Key:	DBDKEY (DBD_Key)	
	References:	ASG.DBD	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
ASG.DBD_HAS_SEGMENT	Col 1:	DBD_Key	Char(8)
	Col 2:	SEGMENT_Key	Char(8)
	Primary Key:	(DBD_Key, SEGMENT_Key)	
	Foreign Key:	DBDKEY (DBD_Key)	
	References:	ASG.DBD	
	Foreign Key:	SEGMTKEY (SEGMENT_Key)	
	References:	ASG.SEGMENT	
ASG.DBD_IN_MBR	Col 1:	DBD_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(DBD_Key, MBR_Key)	
	Foreign Key:	DBDKEY (DBD_Key)	
	References:	ASG.DBD	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	

Table name	Contents		
ASG.DBD_IN_PCB	Col 1:	DBD_Key	Char(8)
	Col 2:	PCB_Key	Char(8)
	Primary Key:	(DBD_Key, PCB_Key)	
	Foreign Key:	DBDKEY (DBD_Key)	
	References:	ASG.DBD	
	Foreign Key:	PCBKEY (PCB_Key)	
	References:	ASG.PCB	
ASG.DD_HAS_DSN	Col 1:	DD_Key	Char(8)
	Col 2:	DSN_Key	Char(8)
	Primary Key:	(DD_Key, DSN_Key)	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
	Foreign Key:	DSNKEY (DSN_Key)	
	References:	ASG.DSN	
ASG.DD_IN_CICS	Col 1:	DD_Key	Char(8)
	Col 2:	CICS_Key	Char(8)
	Primary Key:	(DD_Key, CICS_Key)	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
	Foreign Key:	CICSKEY (CICS_Key)	
	References:	ASG.CICSTXN	
ASG.DD_IN_CSD	Col 1:	DD_Key	Char(8)
	Col 2:	CSD_Key	Char(8)
	Primary Key:	(DD_Key, CSD_Key)	

Table name	Contents		
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
	Foreign Key:	CSDKEY (CSD_Key)	
	References:	ASG.CSD	
ASG.DD_IN_DBD	Col 1:	DD_Key	Char(8)
	Col 2:	DBD_Key	Char(8)
	Primary Key:	(DD_Key, DBD_Key)	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
	Foreign Key:	DBDKEY (DBD_Key)	
	References:	ASG.DBD	
ASG.DD_IN_FCT	Col 1:	DD_Key	Char(8)
	Col 2:	FCT_Key	Char(8)
	Primary Key:	(DD_Key, FCT_Key)	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
	Foreign Key:	FCTKEY (FCT_Key)	
	References:	ASG.FCT	
ASG.DD_IN_STEP	Col 1:	DD_Key	Char(8)
	Col 2:	STEP_Key	Char(8)
	Primary Key:	(DD_Key, STEP_Key)	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	

Table name	Contents		
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
ASG.DD_SELECTEDBY_FD	Col 1:	DD_Key	Char(8)
	Col 2:	FD_Key	Char(8)
	Primary Key:	(DD_Key, FD_Key)	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
	Foreign Key:	FDKEY (FD_Key)	
	References:	ASG.FD	
ASG.DI_REF_MAPFIELD	Col 1:	DATA_Key	Char(8)
	Col 2:	MAPFIELD_Key	Char(8)
	Primary Key:	(DATA_Key, MAPFIELD_Key)	
	Foreign Key:	DIKEY (DATA_Key)	
	References:	ASG.DATAITEM	
	Foreign Key:	MPFKEY (MAPFIELD_Key)	
	References:	ASG.MAPFIELD	
ASG.DSN_HAS_MEMBER	Col 1:	DSN_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(DSN_Key, MBR_Key)	
	Foreign Key:	DSNKEY (DSN_Key)	
	References:	ASG.DSN	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	

Table name	Contents		
ASG.DSN_IN_DD	Col 1:	DSN_Key	Char(8)
	Col 2:	DD_Key	Char(8)
	Primary Key:	(DSN_Key, DD_Key)	
	Foreign Key:	DSNKEY (DSN_Key)	
	References:	ASG.DSN	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
ASG.ENTRY_CALLEDBY_PGM	Col 1:	ENTRY_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(ENTRY_Key, PGM_Key)	
	Foreign Key:	ENTRYKEY (ENTRY_Key)	
	References:	ASG.ENTRY	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.ENTRY_IN_PGM	Col 1:	ENTRY_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(ENTRY_Key, PGM_Key)	
	Foreign Key:	ENTRYKEY (ENTRY_Key)	
	References:	ASG.ENTRY	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.FCT_HAS_DD	Col 1:	FCT_Key	Char(8)
	Col 2:	DD_Key	Char(8)
	Primary Key:	(FCT_Key, DD_Key)	

Table name	Contents		
	Foreign Key:	FCTKEY (FCT_Key)	
	References:	ASG.FCT	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
ASG.FCT_IN_MBR	Col 1:	FCT_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(FCT_Key, MBR_Key)	
	Foreign Key:	FCTKEY (FCT_Key)	
	References:	ASG.FCT	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
ASG.FD_IN_PGM	Col 1:	FD_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(FD_Key, PGM_Key)	
	Foreign Key:	FDKEY (FD_Key)	
	References:	ASG.FD	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.FD_SELECTS_DD	Col 1:	FD_Key	Char(8)
	Col 2:	DD_Key	Char(8)
	Primary Key:	(FD_Key, DD_Key)	
	Foreign Key:	FDKEY (FD_Key)	
	References:	ASG.FD	

Table name	Contents		
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
ASG.FD_USES_RECORD	Col 1:	FD_Key	Char(8)
	Col 2:	REC_Key	Char(8)
	Primary Key:	(FD_Key, REC_Key)	
	Foreign Key:	FDKEY (FD_Key)	
	References:	ASG.FD	
	Foreign Key:	RECKEY (REC_Key)	
	References:	ASG.RECORD	
ASG.FIELD_IN_SEGMENT	Col 1:	FIELD_Key	Char(8)
	Col 2:	SEGMENT_Key	Char(8)
	Primary Key:	(FIELD_Key, SEGMENT_Key)	
	Foreign Key:	FIELDKEY (FIELD_Key)	
	References:	ASG.SEGFIELD	
	Foreign Key:	SEGMTKEY (SEGMENT_Key)	
	References:	ASG.SEGMENT	
ASG.FMT_EXECBY_PGM	Col 1:	FMT_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(FMT_Key, PGM_Key)	
	Foreign Key:	FMTKEY (FMT_Key)	
	References:	ASG.FORMAT	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	

Table name	Contents		
ASG.FORMAT_EXBY_FORMAT	Col 1:	FORMAT_Key	Char(8)
	Col 2:	FORMAT2_Key	Char(8)
	Primary Key:	(FORMAT_Key, FORMAT2_Key)	
	Foreign Key:	FORMTKEY (FORMAT_Key)	
	References:	ASG.FORMAT	
	Foreign Key:	FORMAT2KEY (FORMAT2_Key)	
	References:	ASG.FORMAT	
ASG.FORMAT_EXEC_IMSTXN	Col 1:	FORMAT_Key	Char(8)
	Col 2:	IMSTRANS_Key	Char(8)
	Primary Key:	(FORMAT_Key, IMSTRANS_Key)	
	Foreign Key:	FRMTKEY (FORMAT_Key)	
	References:	ASG.FORMAT	
	Foreign Key:	IMTXNKEY (IMSTRANS_Key)	
	References:	ASG.IMSTXN	
ASG.FORMAT_IN_MBR	Col 1:	FORMAT_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(FORMAT_Key, MBR_Key)	
	Foreign Key:	FMTKEY (FORMAT_Key)	
	References:	ASG.FORMAT	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
ASG.IDMS_IN_PGM	Col 1:	IDMS_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(IDMS_Key, PGM_Key)	

Table name	Contents		
	Foreign Key:	IDMSKEY (IDMS_Key)	
	References:	ASG.IDMS	
	Foreign Key:	FDKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.IDMSMAP_IN_PGM	Col 1:	IDMSMAP_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(IDMSMAP_Key, PGM_Key)	
	Foreign Key:	IMAPKEY (IDMSMAP_Key)	
	References:	ASG.IDMSMAP	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.IDMSREC_IN_PGM	Col 1:	IDMSRECORD_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(IDMSRECORD_Key, PGM_Key)	
	Foreign Key:	IRECKEY (IDMSRECORD_Key)	
	References:	ASG.IDMSRECORD	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.IMSTRANS_EXEC_LOAD	Col 1:	IMSTRANS_Key	Char(8)
	Col 2:	LOAD_Key	Char(8)
	Primary Key:	(IMSTRANS_Key, LOAD_Key)	
	Foreign Key:	IMTXNKEY (IMSTRANS_Key)	
	References:	ASG.IMSTXN	

Table name	Contents		
	Foreign Key:	LOADKEY (LOAD_Key)	
	References:	ASG.LOADMODULE	
ASG.IMSTXN_EXBY_FORMAT	Col 1:	IMSTRANS_Key	Char(8)
	Col 2:	FORMAT_Key	Char(8)
	Primary Key:	(IMSTRANS_Key, FORMAT_Key)	
	Foreign Key:	IMTXNKEY (IMSTRANS_Key)	
	References:	ASG.IMSTXN	
	Foreign Key:	FORMTKEY (FORMAT_Key)	
	References:	ASG.FORMAT	
ASG.IMSTXN_EXECBY_PGM	Col 1:	IMSTRANS_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(IMSTRANS_Key, PGM_Key)	
	Foreign Key:	IMTXNKEY (IMSTRANS_Key)	
	References:	ASG.IMSTXN	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.INCL_HAS_DATA	Col 1:	INCL_Key	Char(8)
	Col 2:	DATA_Key	Char(8)
	Primary Key:	(INCL_Key, DATA_Key)	
	Foreign Key:	INCLKEY (INCL_Key)	
	References:	ASG.COPYMEMBER	
	Foreign Key:	DATAKEY (DATA_Key)	
	References:	ASG.DATAITEM	

Table name	Contents		
ASG.JOB_HAS_STEP	Col 1:	JOB_Key	Char(8)
	Col 2:	STEP_Key	Char(8)
	Primary Key:	(JOB_Key, STEP_Key)	
	Foreign Key:	JOBKEY (JOB_Key)	
	References:	ASG.JOB	
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.JOB	
ASG.JOB_IN_MBR	Col 1:	JOB_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(JOB_Key, MBR_Key)	
	Foreign Key:	JOBKEY (JOB_Key)	
	References:	ASG.JOB	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
ASG.JOB_USES_CATALOG	Col 1:	JOB_Key	Char(8)
	Col 2:	CATALOG_Key	Char(8)
	Primary Key:	(JOB_Key, CATALOG_Key)	
	Foreign Key:	JOBKEY (JOB_Key)	
	References:	ASG.JOB	
	Foreign Key:	CATKEY (CATALOG_Key)	
	References:	ASG.CATALOG	
ASG.JOB_USES_LIBRARY	Col 1:	JOB_Key	Char(8)
	Col 2:	LIBRARY_Key	Char(8)
	Primary Key:	(JOB_Key, LIBRARY_Key)	

Table name	Contents		
	Foreign Key:	JOBKEY (JOB_Key)	
	References:	ASG.JOB	
	Foreign Key:	LIBKEY (LIBRARY_Key)	
	References:	ASG.LIBRARY	
ASG.LIBRARY_HAS_MBR	Col 1:	LIBRARY_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(LIBRARY_Key, MBR_Key)	
	Foreign Key:	LIBKEY (LIBRARY_Key)	
	References:	ASG.LIBRARY	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
ASG.LIBRARY_USDBY_JOB	Col 1:	LIBRARY_Key	Char(8)
	Col 2:	JOB_Key	Char(8)
	Primary Key:	(LIBRARY_Key, JOB_Key)	
	Foreign Key:	LIBKEY (LIBRARY_Key)	
	References:	ASG.LIBRARY	
	Foreign Key:	JOBKEY (JOB_Key)	
	References:	ASG.JOB	
ASG.LIBRARY_USDBY_STP	Col 1:	LIBRARY_Key	Char(8)
	Col 2:	STEP_Key	Char(8)
	Primary Key:	(LIBRARY_Key, STEP_Key)	
	Foreign Key:	LIBKEY (LIBRARY_Key)	
	References:	ASG.LIBRARY	

Table name	Contents		
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
ASG.LM_EXECBY_STEP	Col 1:	LM_Key	Char(8)
	Col 2:	STEP_Key	Char(8)
	Primary Key:	(LM_Key, STEP_Key)	
	Foreign Key:	LMKEY (LM_Key)	
	References:	ASG.LOADMODULE	
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
ASG.LM_HAS_CSECT	Col 1:	LM_Key	Char(8)
	Col 2:	CSECT_Key	Char(8)
	Primary Key:	(LM_Key, CSECT_Key)	
	Foreign Key:	LMKEY (LM_Key)	
	References:	ASG.LOADMODULE	
	Foreign Key:	CSECTKEY (CSECT_Key)	
	References:	ASG.CSECT	
ASG.LM_IN_MBR	Col 1:	LM_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(LM_Key, MBR_Key)	
	Foreign Key:	LMKEY (LM_Key)	
	References:	ASG.LOADMODULE	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	

Table name	Contents		
ASG.LOAD_EXECBY_IMSTXN	Col 1:	LOAD_Key	Char(8)
	Col 2:	IMSTXN_Key	Char(8)
	Primary Key:	(LOAD_Key, IMSTXN_Key)	
	Foreign Key:	LOADKEY (LOAD_Key)	
	References:	ASG.LOADMODULE	
	Foreign Key:	IMTXNKEY (IMSTXN_Key)	
	References:	ASG.IMSTXN	
ASG.LREC_HAS_RECORDS	Col 1:	LREC_Key	Char(8)
	Col 2:	RECORDS_Key	Char(8)
	Primary Key:	(LREC_Key, RECORDS_Key)	
	Foreign Key:	LRECKEY (LREC_Key)	
	References:	ASG.LOGICALRECORD	
	Foreign Key:	RECRDKEY (RECORDS_Key)	
	References:	ASG.RECORD	
ASG.LREC_IN_PGM	Col 1:	LREC_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(LREC_Key, PGM_Key)	
	Foreign Key:	LRECKEY (LREC_Key)	
	References:	ASG.LOGICALRECORD	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.MAP_HAS_MAPFIELD	Col 1:	MAP_Key	Char(8)
	Col 2:	MAPFIELD_Key	Char(8)
	Primary Key:	(MAP_Key, MAPFIELD_Key)	

Table name	Contents		
	Foreign Key:	MAPKEY (MAP_Key)	
	References:	ASG.MAP	
	Foreign Key:	MAPFDKEY (MAPFIELD_Key)	
	References:	ASG.MAPFIELD	
ASG.MAP_HAS_RECORDS	Col 1:	MAP_Key	Char(8)
	Col 2:	RECORDS_Key	Char(8)
	Primary Key:	(MAP_Key, RECORDS_Key)	
	Foreign Key:	MAPKEY (MAP_Key)	
	References:	ASG.MAP	
	Foreign Key:	RECRDKEY (RECORDS_Key)	
	References:	ASG.RECORD	
ASG.MAP_IN_MAPSET	Col 1:	MAP_Key	Char(8)
	Col 2:	MAPSET_Key	Char(8)
	Primary Key:	(MAP_Key, MAPSET_Key)	
	Foreign Key:	MAPKEY (MAP_Key)	
	References:	ASG.MAP	
	Foreign Key:	MAPSTKEY (MAPSET_Key)	
	References:	ASG.MAPSET	
ASG.MAP_USDBY_PGM	Col 1:	MAP_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(MAP_Key, PGM_Key)	
	Foreign Key:	MAPKEY (MAP_Key)	
	References:	ASG.MAP	

Table name	Contents		
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.MAPFIELD_IN_MAP	Col 1:	MAPFIELD_Key	Char(8)
	Col 2:	MAP_Key	Char(8)
	Primary Key:	(MAPFIELD_Key, MAP_Key)	
	Foreign Key:	MAPFDKEY (MAPFIELD_Key)	
	References:	ASG.MAPFIELD	
	Foreign Key:	MAPKEY (MAP_Key)	
	References:	ASG.MAP	
ASG.MAPFIELD_REF_DI	Col 1:	MAPFIELD_Key	Char(8)
	Col 2:	DATA_Key	Char(8)
	Primary Key:	(MAPFIELD_Key, DATA_Key)	
	Foreign Key:	MPFKEY (MAPFIELD_Key)	
	References:	ASG.MAPFIELD	
	Foreign Key:	DIKEY (DATA_Key)	
	References:	ASG.DATAITEM	
ASG.MAPREC_REFR_RECORD	Col 1:	MAPREC_Key	Char(8)
	Col 2:	RECORD_Key	Char(8)
	Primary Key:	(MAPREC_Key, RECORDD_Key)	
	Foreign Key:	MAPRCKEY (MAPREC_Key)	
	References:	ASG.MAP	
	Foreign Key:	RECRDKEY (RECORD_Key)	
	References:	ASG.RECORD	

Table name	Contents		
ASG.MAPSET_HAS_MAP	Col 1:	MAPSET_Key	Char(8)
	Col 2:	MAP_Key	Char(8)
	Primary Key:	(MAPSET_Key, MAP_Key)	
	Foreign Key:	MAPSTKEY (MAPSET_Key)	
	References:	ASG.MAPSET	
	Foreign Key:	MAPKEY (MAP_Key)	
	References:	ASG.MAP	
ASG.MAPSET_IN_MBR	Col 1:	MAPSET_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(MAPSET_Key, MBR_Key)	
	Foreign Key:	MAPSTKEY (MAPSET_Key)	
	References:	ASG.MAPSET	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
ASG.MAPSET_USDBY_PGM	Col 1:	MAPSET_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(MAPSET_Key, PGM_Key)	
	Foreign Key:	MAPSTKEY (MAPSET_Key)	
	References:	ASG.MAPSET	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.MBR_HAS_DBD	Col 1:	MBR_Key	Char(8)
	Col 2:	DBD_Key	Char(8)
	Primary Key:	(MBR_Key, DBD_Key)	

Table name	Contents		
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	DBDKEY (DBD_Key)	
	References:	ASG.DBD	
ASG.MBR_HAS_FCT	Col 1:	MBR_Key	Char(8)
	Col 2:	FCT_Key	Char(8)
	Primary Key:	(MBR_Key, FCT_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	FCTKEY (FCT_Key)	
	References:	ASG.FCT	
ASG.MBR_HAS_FORMAT	Col 1:	MBR_Key	Char(8)
	Col 2:	FORMAT_Key	Char(8)
	Primary Key:	(MBR_Key, FORMAT_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	FMTKEY (FORMAT_Key)	
	References:	ASG.FORMAT	
ASG.MBR_HAS_JOB	Col 1:	MBR_Key	Char(8)
	Col 2:	JOB_Key	Char(8)
	Primary Key:	(MBR_Key, JOB_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	

Table name	Contents		
	Foreign Key:	JOBKEY (JOB_Key)	
	References:	ASG.JOB	
ASG.MBR_HAS_LM	Col 1:	MBR_Key	Char(8)
	Col 2:	LM_Key	Char(8)
	Primary Key:	(MBR_Key, LM_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	LMKEY (LM_Key)	
	References:	ASG.LOADMODULE	
ASG.MBR_HAS_MAPSET	Col 1:	MBR_Key	Char(8)
	Col 2:	MAPSET_Key	Char(8)
	Primary Key:	(MBR_Key, MAPSET_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	MAPSTKEY (MAPSET_Key)	
	References:	ASG.MAPSET	
ASG.MBR_HAS_MDADB	Col 1:	MBR_Key	Char(8)
	Col 2:	MDADB_Key	Char(8)
	Primary Key:	(MBR_Key, MDADB_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	MDADBKEY (MDADB_Key)	
	References:	ASG.IMSDYNALLOC-MACRO	

Table name	Contents		
ASG.MBR_HAS_PCT	Col 1:	MBR_Key	Char(8)
	Col 2:	PCT_Key	Char(8)
	Primary Key:	(MBR_Key, PCT_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	PCTKEY (PCT_Key)	
	References:	ASG.PCT	
ASG.MBR_HAS_PGM	Col 1:	MBR_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(MBR_Key, PGM_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.MBR_HAS_PPT	Col 1:	MBR_Key	Char(8)
	Col 2:	PPT_Key	Char(8)
	Primary Key:	(MBR_Key, PPT_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	PPTKEY (PPT_Key)	
	References:	ASG.PPT	
ASG.MBR_HAS_PROC	Col 1:	MBR_Key	Char(8)
	Col 2:	PROC_Key	Char(8)
	Primary Key:	(MBR_Key, PROC_Key)	

Table name	Contents		
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	PROCKEY (PROC_Key)	
	References:	ASG.PROC	
ASG.MBR_HAS_PSB	Col 1:	MBR_Key	Char(8)
	Col 2:	PSB_Key	Char(8)
	Primary Key:	(MBR_Key, PSB_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	PSBKEY (PSB_Key)	
	References:	ASG.PSB	
ASG.MBR_IN_DSN	Col 1:	MEMBER_Key	Char(8)
	Col 2:	DSN_Key	Char(8)
	Primary Key:	(MBR_Key, LIBRARY_Key)	
	Foreign Key:	MEMBERKEY (DSN_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	DSNKEY (DSN_Key)	
	References:	ASG.DSN	
ASG.MBR_IN_LIBRARY	Col 1:	MBR_Key	Char(8)
	Col 2:	LIBRARY_Key	Char(8)
	Primary Key:	(MBR_Key, LIBRARY_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	

Table name	Contents		
	Foreign Key:	LIBKEY (LIBRARY_Key)	
	References:	ASG.LIBRARY	
ASG.MBR_INCLIN_SOURCE	Col 1:	MBR_Key	Char(8)
	Col 2:	SOURCE_Key	Char(8)
	Primary Key:	(MBR_Key, SOURCE_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.COPYMEMBER	
	Foreign Key:	SRCKEY (SOURCE_Key)	
	References:	ASG.MEMBER	
ASG.MBR_IS_COPYINCL	Col 1:	MBR_Key	Char(8)
	Col 2:	COPYINCL_Key	Char(8)
	Primary Key:	(MBR_Key, COPYINCL_Key)	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
	Foreign Key:	CPYMBKEY (COPYINCL_Key)	
	References:	ASG.COPYMEMBER	
ASG.MDA_DBIN_MBR	Col 1:	MDA_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(MDA_Key, MBR_Key)	
	Foreign Key:	MDAKEY (MDA_Key)	
	References:	ASG.IMSDYNALLOC-MACRO	
	Foreign Key:	KEY (MBR_Key)	
	References:	ASG.MEMBER	

Table name	Contents		
ASG.MDA_HAS_DD	Col 1:	MDA_Key	Char(8)
	Col 2:	DD_Key	Char(8)
	Primary Key:	(MDA_Key, DD_Key)	
	Foreign Key:	MDAKEY (MDA_Key)	
	References:	ASG.IMSDYNALLOC-MACRO	
	Foreign Key:	RECRDKEY (RECORDS_Key)	
	References:	ASG.RECORD	
ASG.PCB_HAS_DBD	Col 1:	PCB_Key	Char(8)
	Col 2:	DBD_Key	Char(8)
	Primary Key:	(PCB_Key, DBD_Key)	
	Foreign Key:	PCBKEY (PCB_Key)	
	References:	ASG.PCB	
	Foreign Key:	DBDKEY DBD_Key)	
	References:	ASG.DBD	
ASG.PCB_HAS_SENSEG	Col 1:	PCB_Key	Char(8)
	Col 2:	SENSEG_Key	Char(8)
	Primary Key:	(PCB_Key, SENSEG_Key)	
	Foreign Key:	PCBKEY (PCB_Key)	
	References:	ASG.PCB	
	Foreign Key:	SENSGKEY (SENSEG_Key)	
	References:	ASG.SENSEG	
ASG.PCB_IN_PSB	Col 1:	PCB_Key	Char(8)
	Col 2:	PSB_Key	Char(8)
	Primary Key:	(PCB_Key, PSB_Key)	

Table name	Contents		
	Foreign Key:	PCBKEY (PCB_Key)	
	References:	ASG.PCB	
	Foreign Key:	PSBKEY (PSB_Key)	
	References:	ASG.PSB	
ASG.PCT_HAS_CICSTRANS	Col 1:	PCT_Key	Char(8)
	Col 2:	CICSTRANS_Key	Char(8)
	Primary Key:	(PCT_Key,CICSTRANS_Key)	
	Foreign Key:	PCTKEY (PCT_Key)	
	References:	ASG.PCT	
	Foreign Key:	CTXNKEY (CICSTRANS_Key)	
	References:	ASG.CICSTXN	
ASG.PCT_IN_MBR	Col 1:	PCT_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(PCT_Key, MBR_Key)	
	Foreign Key:	PCTKEY (JOB_Key)	
	References:	ASG.PCT	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
ASG.PGM_CALLS_ENTRY	Col 1:	PGM_Key	Char(8)
	Col 2:	Entry_Key	Char(8)
	Primary Key:	(PGM_Key, Entry_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	

Table name	Contents		
	References:	ASG.PROGRAM	
	Foreign Key:	ENTRYKEY (Entry_Key)	
	References:	ASG.ENTRY	
ASG.PGM_EXEC_FMT	Col 1:	PGM_Key	Char(8)
	Col 2:	FMT_Key	Char(8)
	Primary Key:	(PGM_Key, FMT_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	FMTKEY (FMT_Key)	
	References:	ASG.FORMAT	
ASG.PGM_EXEC_IMSTRANS	Col 1:	PGM_Key	Char(8)
	Col 2:	IMSTRANS_Key	Char(8)
	Primary Key:	(PGM_Key, IMSTRANS_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	ITXNKEY (IMSTRANS_Key)	
	References:	ASG.IMSTXN	
ASG.PGM_EXEC_TRANS	Col 1:	PGM_Key	Char(8)
	Col 2:	TRANS_Key	Char(8)
	Primary Key:	(PGM_Key, TRANS_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	TRANSKEY (TRANS_Key)	
	References:	ASG.CICSTXN	

Table name	Contents		
ASG.PGM_EXECBY_CICSTXN	Col 1:	PGM_Key	Char(8)
	Col 2:	CICSTRANS_Key	Char(8)
	Primary Key:	(PGM_Key, CICSTRANS_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	CITXNKEY (CICSTRANS_Key)	
	References:	ASG.CICSTXN	
ASG.PGM_HAS_AREAS	Col 1:	PGM_Key	Char(8)
	Col 2:	AREA_Key	Char(8)
	Primary Key:	(PGM_Key, AREA_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	AREAKEY (AREA_Key)	
	References:	ASG.AREA	
ASG.PGM_HAS_COLUMNS	Col 1:	PGM_Key	Char(8)
	Col 2:	COLUMN_Key	Char(8)
	Primary Key:	(PGM_Key, COLUMN_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	COLKEY (COLUMN_Key)	
	References:	ASG.COLUMN	
ASG.PGM_HAS_CURSOR	Col 1:	PGM_Key	Char(8)
	Col 2:	CURSOR_Key	Char(8)
	Primary Key:	(PGM_Key, CURSOR_Key)	

Table name	Contents		
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	CRSKEY (CURSOR_Key)	
	References:	ASG.CURSOR	
ASG.PGM_HAS_DB	Col 1:	PGM_Key	Char(8)
	Col 2:	DATABASE_Key	Char(8)
	Primary Key:	(PGM_Key, DATABASE_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	DBKEY (DATABASE_Key)	
	References:	ASG.DATABASE	
ASG.PGM_HAS_ENTRY	Col 1:	PGM_Key	Char(8)
	Col 2:	Entry_Key	Char(8)
	Primary Key:	(PGM_Key, Entry_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	ENTRYKEY (Entry_Key)	
	References:	ASG.ENTRY	
ASG.PGM_HAS_FD	Col 1:	PGM_Key	Char(8)
	Col 2:	FD_Key	Char(8)
	Primary Key:	(PGM_Key, FD_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	

Table name	Contents		
	Foreign Key:	FDKEY (FD_Key)	
	References:	ASG.FD	
ASG.PGM_HAS_IDMS	Col 1:	PGM_Key	Char(8)
	Col 2:	IDMS_Key	Char(8)
	Primary Key:	(PGM_Key, IDMS_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	IDMSKEY (IDMS_Key)	
	References:	ASG.IDMS	
ASG.PGM_HAS_IDSMAP	Col 1:	PGM_Key	Char(8)
	Col 2:	IDSMAP_Key	Char(8)
	Primary Key:	(PGM_Key, IDSMAP_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	IMAPKEY (IDSMAP_Key)	
	References:	ASG.IDSMAP	
ASG.PGM_HAS_IDMSREC	Col 1:	PGM_Key	Char(8)
	Col 2:	IDMSREC_Key	Char(8)
	Primary Key:	(PGM_Key, IDMSREC_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	IRECKEY (IDMSREC_Key)	
	References:	ASG.IDMSRECORD	

Table name	Contents		
ASG.PGM_HAS_LREC	Col 1:	PGM_Key	Char(8)
	Col 2:	LREC_Key	Char(8)
	Primary Key:	(PGM_Key, IDMS_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	LRECKEY (LREC_Key)	
	References:	ASG.LOGICALRECORD	
ASG.PGM_HAS_RECORDS	Col 1:	PGM_Key	Char(8)
	Col 2:	RECORDS_Key	Char(8)
	Primary Key:	(PGM_Key, RECORDS_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	RECRDKEY (RECORDS_Key)	
	References:	ASG.RECORD	
ASG.PGM_HAS_SCHEMA	Col 1:	PGM_Key	Char(8)
	Col 2:	SCHEMA_Key	Char(8)
	Primary Key:	(PGM_Key, SCHEMA_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	SCHKEY (SCHEMA_Key)	
	References:	ASG.SCHEMA	
ASG.PGM_HAS_SQL	Col 1:	PGM_Key	Char(8)
	Col 2:	SQL_Key	Char(8)
	Primary Key:	(PGM_Key, SQL_Key)	

Table name	Contents		
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	SQLKEY (SQL_Key)	
	References:	ASG.SQL	
ASG.PGM_HAS_STOGROU	Col 1:	PGM_Key	Char(8)
	Col 2:	STOGROUP_Key	Char(8)
	Primary Key:	(PGM_Key, STOGROUP_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	STGKEY (STRGROUP_Key)	
	References:	ASG.STOGROUP	
ASG.PGM_HAS_SUBSCHEMA	Col 1:	PGM_Key	Char(8)
	Col 2:	SUBSCHEMA_Key	Char(8)
	Primary Key:	(PGM_Key, SUBSCHEMA_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	SUBKEY (SUBSCHEMA_Key)	
	References:	ASG.SUBSCHEMA	
ASG.PGM_HAS_TABLES	Col 1:	PGM_Key	Char(8)
	Col 2:	TABLE_Key	Char(8)
	Primary Key:	(PGM_Key, TABLE_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	

Table name	Contents		
	Foreign Key:	TABLEKEY (TABLE_Key)	
	References:	ASG.TABLE	
ASG.PGM_HAS_TBLSP	Col 1:	PGM_Key	Char(8)
	Col 2:	TABLESPACE_Key	Char(8)
	Primary Key:	(PGM_Key, TABLESPACE_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	TBLSPKEY (TABLESPACE_Key)	
	References:	ASG.TABLESPACE	
ASG.PGM_HAS_VIEW	Col 1:	PGM_Key	Char(8)
	Col 2:	VIEW_Key	Char(8)
	Primary Key:	(PGM_Key, VIEW_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	VIEWKEY (VIEW_Key)	
	References:	ASG.VIEW	
ASG.PGM_IN_MBR	Col 1:	PGM_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(PGM_Key, MBR_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	

Table name	Contents		
ASG.PGM_INCL_COPYMBR	Col 1:	PGM_Key	Char(8)
	Col 2:	COPYMBR_Key	Char(8)
	Primary Key:	(PGM_Key, COPYMBR_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	CPYMBKEY (COPYMBR_Key)	
	References:	ASG.COPYMBR	
ASG.PGM_IS_CSECT	Col 1:	PGM_Key	Char(8)
	Col 2:	CSECT_Key	Char(8)
	Primary Key:	(PGM_Key, CSECT_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	CSECTKEY (CSECT_Key)	
	References:	ASG.CSECT	
ASG.PGM_LINK_PGM	Col 1:	PGM_Key	Char(8)
	Col 2:	PGM2_Key	Char(8)
	Primary Key:	(PGM_Key, PGM2_Key)	
	Foreign Key:	PGMKEY PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	PGM2KEY (PGM2_Key)	
	References:	ASG.PROGRAM	
ASG.PGM_LINKEDBY_PGM	Col 1:	PGM_Key	Char(8)
	Col 2:	PGM2_Key	Char(8)
	Primary Key:	(PGM_Key, PGM2_Key)	

Table name	Contents		
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	PGM2KEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.PGM_USES_CICSDD	Col 1:	PGM_Key	Char(8)
	Col 2:	CICSDD_Key	Char(8)
	Primary Key:	(PGM_Key, CICSDD_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	CIDDKEY (CICSDD_Key)	
	References:	ASG.DD	
ASG.PGM_USES_MAP	Col 1:	PGM_Key	Char(8)
	Col 2:	MAP_Key	Char(8)
	Primary Key:	(PGM_Key, MAP_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	MAPKEY (MAP_Key)	
	References:	ASG.MAP	
ASG.PGM_USES_MAPSET	Col 1:	PGM_Key	Char(8)
	Col 2:	MAPSET_Key	Char(8)
	Primary Key:	(PGM_Key, MAPSET_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	

Table name	Contents		
	References:	ASG.PROGRAM	
	Foreign Key:	MAPSTKEY (MAPSET_Key)	
	References:	ASG.MAPSET	
ASG.PGM_USES_PSB	Col 1:	PGM_Key	Char(8)
	Col 2:	PSB_Key	Char(8)
	Primary Key:	(PGM_Key, PSB_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	PSBKEY (PSB_Key)	
	References:	ASG.PSB	
ASG.PGM_USES_SEGMENT	Col 1:	PGM_Key	Char(8)
	Col 2:	SEGMENT_Key	Char(8)
	Primary Key:	(PGM_Key, SEGMENT_Key)	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	SEGKEY (SEGMENT_Key)	
	References:	ASG.SEGMENT	
ASG.PPT_IN_MBR	Col 1:	PPT_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(PPT_Key, MBR_Key)	
	Foreign Key:	PPTKEY (PPT_Key)	
	References:	ASG.PPT	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	

Table name	Contents		
ASG.PROC_EXECBY_STEP	Col 1:	PROC_Key	Char(8)
	Col 2:	STEP_Key	Char(8)
	Primary Key:	(PROC_Key, STEP_Key)	
	Foreign Key:	PROCKEY (PROC_Key)	
	References:	ASG.PROC	
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
ASG.PROC_HAS_STEP	Col 1:	PROC_Key	Char(8)
	Col 2:	STEP_Key	Char(8)
	Primary Key:	(PROC_Key, STEP_Key)	
	Foreign Key:	PROCKEY (PROC_Key)	
	References:	ASG.PROC	
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
ASG.PROC_IN_MBR	Col 1:	PROC_Key	Char(8)
	Col 2:	MBR_Key	
	Primary Key:	(PROC_Key, MBR_Key)	
	Foreign Key:	PROCKEY (REC_Key)	
	References:	ASG.PROC	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
ASG.PROGRAM_HAS_DATA	Col 1:	PROGRAM_Key	Char(8)
	Col 2:	DATA_Key	Char(8)
	Primary Key:	(PROGRAM_Key, DATA_Key)	

Table name	Contents		
	Foreign Key:	PGMKEY (PROGRAM_Key)	
	References:	ASG.PROGRAM	
	Foreign Key:	DATAKEY (DATA_Key)	
	References:	ASG.DATAITEM	
ASG.PSB_HAS_PCB	Col 1:	PSB_Key	Char(8)
	Col 2:	PCB_Key	Char(8)
	Primary Key:	(PSB_Key, PCB_Key)	
	Foreign Key:	PSBKEY (PSB_Key)	
	References:	ASG.PSB	
	Foreign Key:	PCBKEY (PCB_Key)	
	References:	ASG.PCB	
ASG.PSB_IN_MBR	Col 1:	PSB_Key	Char(8)
	Col 2:	MBR_Key	Char(8)
	Primary Key:	(PSB_Key, MBR_Key)	
	Foreign Key:	PSBKEY (PGM_Key)	
	References:	ASG.PSB	
	Foreign Key:	MBRKEY (MBR_Key)	
	References:	ASG.MEMBER	
ASG.PSB_USDBY_PGM	Col 1:	PSB_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(PSB_Key, PGM_Key)	
	Foreign Key:	PSBKEY (PSB_Key)	
	References:	ASG.PSB	

Table name	Contents		
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.REC_HAS_DATAITEM	Col 1:	REC_Key	Char(8)
	Col 2:	DATA_Key	Char(8)
	Primary Key:	(REC_Key, DATA_Key)	
	Foreign Key:	RECKEY (REC_Key)	
	References:	ASG.RECORD	
	Foreign Key:	DATAKEY (DATA_Key)	
	References:	ASG.DATAITEM	
ASG.REC_REFBY_MAPREC	Col 1:	RECORD_Key	Char(8)
	Col 2:	MAPREC_Key	Char(8)
	Primary Key:	(RECORD_Key, MAPREC_Key)	
	Foreign Key:	RECRDKEY (RECORD_Key)	
	References:	ASG.RECORD	
	Foreign Key:	MAPRCKEY (MAPREC_Key)	
	References:	ASG.MAP	
ASG.RECORD_IN_PGM	Col 1:	REC_Key	Char(8)
	Col 2:	PROGRAM_Key	Char(8)
	Primary Key:	(REC_Key, PROGRAM_Key)	
	Foreign Key:	RECKEY (REC_Key)	
	References:	ASG.RECORD	
	Foreign Key:	PGMKEY (PROGRAM_Key)	
	References:	ASG.PROGRAM	

Table name	Contents		
ASG.RECORD_USDBY_FD	Col 1:	RECORD_Key	Char(8)
	Col 2:	FD_Key	Char(8)
	Primary Key:	(RECORD_Key, FD_Key)	
	Foreign Key:	RECRDKEY (RECORD_Key)	
	References:	ASG.RECORD	
	Foreign Key:	FDKEY (FD_Key)	
	References:	ASG.FD	
ASG.RECORDS_IN_AREA	Col 1:	RECORDS_Key	Char(8)
	Col 2:	AREA_Key	Char(8)
	Primary Key:	(RECORDS_Key, AREA_Key)	
	Foreign Key:	RECRDKEY (RECORDS_Key)	
	References:	ASG.RECORD	
	Foreign Key:	AREAKEY (AREA_Key)	
	References:	ASG.AREA	
ASG.RECORDS_IN_LREC	Col 1:	RECORDS_Key	Char(8)
	Col 2:	LREC_Key	Char(8)
	Primary Key:	(RECORDS_Key, REC_Key)	
	Foreign Key:	RECRDKEY (RECORDS_Key)	
	References:	ASG.RECORD	
	Foreign Key:	LRECKEY (LREC_Key)	
	References:	ASG.LOGICALRECORD	
ASG.RECORDS_IN_MAP	Col 1:	RECORDS_Key	Char(8)
	Col 2:	MAP_Key	Char(8)
	Primary Key:	(RECORDS_Key, MAP_Key)	

Table name	Contents		
	Foreign Key:	RECRDKEY (RECORDS_Key)	
	References:	ASG.RECORD	
	Foreign Key:	MAPKEY (MAP_Key)	
	References:	ASG.MAP	
ASG.SCHEMA_HAS_SUBSCHM	Col 1:	SCHEMA_Key	Char(8)
	Col 2:	SUBSCHEMA_Key	Char(8)
	Primary Key:	(SCHEMA_Key, SUBSCHEMA_Key)	
	Foreign Key:	SCHEMKEY (SCHEMA_Key)	
	References:	ASG.SCHEMA	
	Foreign Key:	SUBCKEY (SUBSCHEMA_Key)	
	References:	ASG.SUBSCHEMA	
ASG.SCHEMA_IN_PGM	Col 1:	SCHEMA_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(SCHEMA_Key, PGM_Key)	
	Foreign Key:	SCHKEY (SCHEMA_Key)	
	References:	ASG.SCHEMA	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.SEGMENT_HAS_FIELD	Col 1:	SEGMENT_Key	Char(8)
	Col 2:	FIELD_Key	Char(8)
	Primary Key:	(SEGMENT_Key, FIELD_Key)	
	Foreign Key:	SEGMTKEY (SEGMENT_Key)	
	References:	ASG.SEGMENT	

Table name	Contents		
	Foreign Key:	FIELDKEY (FIELD_Key)	
	References:	ASG.SEGFIELD	
ASG.SEGMENT_HAS_SEGMNT	Col 1:	SEGMENT_Key	Char(8)
	Col 2:	SEGMENT2_Key	Char(8)
	Primary Key:	(SEGMENT_Key, SEGMENT2_Key)	
	Foreign Key:	SEGMTKEY (SEGMENT_Key)	
	References:	ASG.SEGMENT	
	Foreign Key:	SEGM2KEY (SEGMENT2_Key)	
	References:	ASG.SEGMENT	
ASG.SEGMENT_IN_DBD	Col 1:	SEGMENT_Key	Char(8)
	Col 2:	DBD_Key	Char(8)
	Primary Key:	(SEGMENT_Key, DBD_Key)	
	Foreign Key:	SEGMTKEY (SEGMENT_Key)	
	References:	ASG.SEGMENT	
	Foreign Key:	DBDKEY (DBD_Key)	
	References:	ASG.DBD	
ASG.SEGMENT_IN_SEGMENT	Col 1:	SEGMENT_Key	Char(8)
	Col 2:	SEGMENT2_Key	Char(8)
	Primary Key:	(SEGMENT_Key, SEGMENT2_Key)	
	Foreign Key:	SEGMTKEY (SEGMENT_Key)	
	References:	ASG.SEGMENT	
	Foreign Key:	SEGM2KEY (SEGMENT2_Key)	
	References:	ASG.SEGMENT	

Table name	Contents		
ASG.SEGMENT_USDBY_PGM	Col 1:	SEGMENT_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(SEGMENT_Key, PGM_Key)	
	Foreign Key:	SEGKEY (SEGMENT_Key)	
	References:	ASG.SEGMENT	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.SENFLD_IN_SENSEG	Col 1:	SENFIELD_Key	Char(8)
	Col 2:	SENSEG_Key	Char(8)
	Primary Key:	(SENFIELD_Key, SENSEG_Key)	
	Foreign Key:	SENFDKEY (SENFIELD_Key)	
	References:	ASG.SENFIELD	
	Foreign Key:	SENSGKEY (SENSEG_Key)	
	References:	ASG.SENSEG	
ASG.SENSEG_HAS_SENFLD	Col 1:	SENSEG_Key	Char(8)
	Col 2:	SENFIELD_Key	Char(8)
	Primary Key:	(SENSEG_Key, SENFIELD_Key)	
	Foreign Key:	SENSG (SENSEG_Key)	
	References:	ASG.SENSEG	
	Foreign Key:	SENFDKEY (SENFIELD_Key)	
	References:	ASG.SENFIELD	
ASG.SENSEG_HAS_SENSEG	Col 1:	SENSEG_Key	Char(8)
	Col 2:	SENSEG2_Key	Char(8)
	Primary Key:	(SENSEG_Key, SENSEG2_Key)	

Table name	Contents		
	Foreign Key:	SENSGKEY (SENSEG_Key)	
	References:	ASG.SENSEG	
	Foreign Key:	SENS2KEY (SENSEG2_Key)	
	References:	ASG.SENSEG	
ASG.SENSEG_IN_PCB	Col 1:	SENSEG_Key	Char(8)
	Col 2:	PCB_Key	Char(8)
	Primary Key:	(SENSEG_Key, PCB_Key)	
	Foreign Key:	SENSGKEY (SENSEG_Key)	
	References:	ASG.SENSEG	
	Foreign Key:	PCBKEY (PCB_Key)	
	References:	ASG.PCB	
ASG.SENSEG_IN_SENSEG	Col 1:	SENSEG_Key	Char(8)
	Col 2:	SENSEG2_Key	Char(8)
	Primary Key:	(SENSEG_Key, SENSEG2_Key)	
	Foreign Key:	SENSGKEY (SENSEG_Key)	
	References:	ASG.SENSEG	
	Foreign Key:	SENS2KEY (SENSEG2_Key)	
	References:	ASG.SENSEG	
ASG.SOURCE_INCL_MEMBER	Col 1:	SOURCE_Key	Char(8)
	Col 2:	MEMBER_Key	Char(8)
	Primary Key:	(SOURCE_Key, MEMBER_Key)	
	Foreign Key:	SRCKEY (SOURCE_Key)	
	References:	ASG.MEMBER	

Table name	Contents		
	Foreign Key:	MBRKEY (MEMBER_Key)	
	References:	ASG.COPYMEMBER	
ASG.SQL_IN_PGM	Col 1:	SQL_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(SQL_Key, PGM_Key)	
	Foreign Key:	SQLKEY (SQL_Key)	
	References:	ASG.SQL	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.STEP_EXECS_LM	Col 1:	STEP_Key	Char(8)
	Col 2:	LM_Key	Char(8)
	Primary Key:	(STEP_Key, LM_Key)	
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
	Foreign Key:	LMKEY (LM_Key)	
	References:	ASG.LOADMODULE	
ASG.STEP_EXECS_PROC	Col 1:	STEP_Key	Char(8)
	Col 2:	PROC_Key	Char(8)
	Primary Key:	(STEP_Key, PROC_Key)	
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
	Foreign Key:	PROCKEY (PROC_Key)	
	References:	ASG.PROC	

Table name	Contents		
ASG.STEP_HAS_DD	Col 1:	STEP_Key	Char(8)
	Col 2:	DD_Key	Char(8)
	Primary Key:	(STEP_Key, DD_Key)	
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
	Foreign Key:	DDKEY (DD_Key)	
	References:	ASG.DD	
ASG.STEP_IN_JOB	Col 1:	STEP_Key	Char(8)
	Col 2:	JOB_Key	Char(8)
	Primary Key:	(STEP_Key, JOB_Key)	
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
	Foreign Key:	JOBKEY (JOB_Key)	
	References:	ASG.JOB	
ASG.STEP_IN_PROC	Col 1:	STEP_Key	Char(8)
	Col 2:	PROC_Key	Char(8)
	Primary Key:	(STEP_Key, PROC_Key)	
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
	Foreign Key:	PROCKEY (PROC_Key)	
	References:	ASG.PROC	
ASG.STEP_USES_CATALOG	Col 1:	STEP_Key	Char(8)
	Col 2:	CATALOG_Key	Char(8)
	Primary Key:	(STEP_Key, CATALOG_Key)	

Table name	Contents		
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
	Foreign Key:	CATKEY (CATALOG_Key)	
	References:	ASG.CATALOG	
ASG.STEP_USES_LIBRARY	Col 1:	STEP_Key	Char(8)
	Col 2:	LIBRARY_Key	Char(8)
	Primary Key:	(STEP_Key, LIBRARY_Key)	
	Foreign Key:	STEPKEY (STEP_Key)	
	References:	ASG.STEP	
	Foreign Key:	LIBKEY (LIBRARY_Key)	
	References:	ASG.LIBRARY	
ASG.STOGP_HAS_DATABASE	Col 1:	STOGRP_Key	Char(8)
	Col 2:	DATABASE_Key	Char(8)
	Primary Key:	(STOGRP_Key, DATABASE_Key)	
	Foreign Key:	STOGPKEY (STOGRP_Key)	
	References:	ASG.STOGROUP	
	Foreign Key:	DBASEKEY (DATABASE_Key)	
	References:	ASG.DATABASE	
ASG.STOGROUP_IN_PGM	Col 1:	STOGROUP_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(STOGROUP_Key, PGM_Key)	
	Foreign Key:	STGKEY (STOGROUP_Key)	
	References:	ASG.STOGROUP	

Table name	Contents		
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.SUBSCH_HAS_AREAS	Col 1:	SUBSCHEMA_Key	Char(8)
	Col 2:	AREA_Key	Char(8)
	Primary Key:	SUBSCHEMA_Key, AREA_Key)	
	Foreign Key:	SUBKEY (SUBSCHEMA_Key)	
	References:	ASG.SUBSCHEMA	
	Foreign Key:	AREAEKEY (AREA_Key)	
	References:	ASG.AREA	
ASG.SUBSCHEMA_HAS_LREC	Col 1:	SUBSCHEMA_Key	Char(8)
	Col 2:	LREC_Key	Char(8)
	Primary Key:	(SUBSCHEMA_Key, LREC_Key)	
	Foreign Key:	SUBSCKEY (SUBSCHEMA_Key)	
	References:	ASG.SUBSCHEMA	
	Foreign Key:	LRECKEY (LREC_Key)	
	References:	ASG.LOGICALRECORD	
ASG.SUBSCHEMA_IN_PGM	Col 1:	SUBSCHEMA_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(SUBSCHEMA_Key, PGM_Key)	
	Foreign Key:	SUBKEY (SUBSCHEMA_Key)	
	References:	ASG.SUBSCHEMA	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	

Table name	Contents		
ASG.SUBSCHM_IN_SCHEMA	Col 1:	SUBSCHEMA_Key	Char(8)
	Col 2:	SCHEMA_Key	Char(8)
	Primary Key:	(SUBSCHEMA_Key, SCHEMA_Key)	
	Foreign Key:	SUBSCKEY (SUBSCHEMA_Key)	
	References:	ASG.SUBSCHEMA	
	Foreign Key:	SCHEMKEY (SCHEMA_Key)	
	References:	ASG.SCHEMA	
ASG.TABLE_HAS_COLUMNS	Col 1:	TABLE_Key	Char(8)
	Col 2:	COLUMNS_Key	Char(8)
	Primary Key:	(TABLE_Key, COLUMNS_Key)	
	Foreign Key:	TABLEKEY (TABLE_Key)	
	References:	ASG.TABLE	
	Foreign Key:	COLUMKEY (COLUMNS_Key)	
	References:	ASG.COLUMN	
ASG.TABLES_IN_PGM	Col 1:	TABLE_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(TABLE_Key, PGM_Key)	
	Foreign Key:	TABLEKEY (TABLE_Key)	
	References:	ASG.TABLE	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.TABLES_IN_TBLSP	Col 1:	TABLES_Key	Char(8)
	Col 2:	TBLSP_Key	Char(8)
	Primary Key:	(TABLES_Key, TBLSP_Key)	

Table name	Contents		
	Foreign Key:	TABLEKEY (TABLES_Key)	
	References:	ASG.TABLE	
	Foreign Key:	TBLSPKEY (TBLSP_Key)	
	References:	ASG.TABLESPACE	
ASG.TBLSP_HAS_TABLES	Col 1:	TBLSP_Key	Char(8)
	Col 2:	TABLES_Key	Char(8)
	Primary Key:	(TBLSP_Key, TABLES_Key)	
	Foreign Key:	TBLSPKEY (TBLSP_Key)	
	References:	ASG.TABLESPACE	
	Foreign Key:	TABLEKEY (TABLES_Key)	
	References:	ASG.TABLE	
ASG.TBLSP_HAS_VIEWS	Col 1:	TBLSP_Key	Char(8)
	Col 2:	VIEWS_Key	Char(8)
	Primary Key:	(TBLSP_Key, TBLSP_Key)	
	Foreign Key:	TBLSPKEY (TBLSP_Key)	
	References:	ASG.TABLESPACE	
	Foreign Key:	VIEWSKEY (VIEWS_Key)	
	References:	ASG.VIEW	
ASG.TBLSP_IN_DATABASE	Col 1:	TBLSP_Key	Char(8)
	Col 2:	DATABASE_Key	Char(8)
	Primary Key:	(TBLSP_Key, DATABASE_Key)	
	Foreign Key:	TBLSPKEY (TBLSP_Key)	
	References:	ASG.TABLESPACE	

Table name	Contents		
	Foreign Key:	DBASEKEY (DATABASE_Key)	
	References:	ASG.DATABASE	
ASG.TBLSP_IN_PGM	Col 1:	TBLSP_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(TBLSP_Key, PGM_Key)	
	Foreign Key:	TBLSPKEY (TBLSP_Key)	
	References:	ASG.TABLESPACE	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.TRANS_EXECBY_PGM	Col 1:	TRANS_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(TRANS_Key, PGM_Key)	
	Foreign Key:	TRANSKEY (TRANS_Key)	
	References:	ASG.CICSTXN	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.VIEW_HAS_COLUMNS	Col 1:	VIEW_Key	Char(8)
	Col 2:	COLUMNS_Key	Char(8)
	Primary Key:	(VIEW_Key, COLUMNS_Key)	
	Foreign Key:	VIEWKEY (VIEW_Key)	
	References:	ASG.VIEW	
	Foreign Key:	COLUMKEY (COLUMNS_Key)	
	References:	ASG.COLUMN	

Table name	Contents		
ASG.VIEWS_IN_PGM	Col 1:	VIEW_Key	Char(8)
	Col 2:	PGM_Key	Char(8)
	Primary Key:	(VIEW_Key, PGM_Key)	
	Foreign Key:	VIEWKEY (VIEW_Key)	
	References:	ASG.VIEW	
	Foreign Key:	PGMKEY (PGM_Key)	
	References:	ASG.PROGRAM	
ASG.VIEWS_IN_TBLSP	Col 1:	VIEWS_Key	Char(8)
	Col 2:	TBLSP_Key	Char(8)
	Primary Key:	(VIEWS_Key, TBLSP_Key)	
	Foreign Key:	VIEWSKEY (VIEWS_Key)	
	References:	ASG.VIEW	
	Foreign Key:	TBLSPKEY (TBLSP_Key)	
	References:	ASG.TABLESPACE	

Exporting to Transition Enablement Facility (TEF)

Alliance has an optional TEF export feature you can use to export a defined and analyzed application to Texas Instruments Transition Enablement Facility (TI-TEF). The application (that is, a group of programs and associated components) is created and stored in the AKR. The application components and attributes are defined.

To export to TEF

- 1 Review and edit the defined application online.

For more information on these procedures see the *ASG-Application Definition and Analysis User's Guide*.

- 2 Analyze the defined application, and review the results of the analyze.

For more information on these procedures see the *ASG-Application Definition and Analysis User's Guide*.

- 3 Export the analyzed information (including data model and control flow model) exported to TEF.
 - A batch job is submitted to build the input for TEF.
 - The information is exported into three output files (Object, Association, and Properties) that are suitable for TEF Release 2.0.
 - TEF manipulates this information into a loadable model format for use in Texas Instruments Composer Release 5.3 (for example, an Information Engineering Facility product).

For additional information on these terms, see the Texas Instruments TEF and Composer documentation:

Attribute (ATTRUSR)	Entity type (HLENT)
Business system	Expected effects
Common action block	Function
CRUD	Procedure (BUSPROC)
Dialog flow (DLGFLOW)	Procedure step (BUSPRST)
Elementary process	Subject area (SUBJ)

TEF Objects

When you create TEF objects to represent information in the AKR and the TEF object has a name property, the actual name of the object is used. You may modify the name with descriptive text, but the name itself is that of the entity found in the AKR and derived from the application. All other TEF properties are fully populated, except as noted.

Data Model

The TEF Export option provides you with data model information that is based on the data structures and database activity within programs.

- When CRUD (Create, Read, Update, Delete) activity is identified, TEF objects and relationships are created to represent local records and database entities involved in the activity.
- TEF objects and relationships are created to represent the CRUD activity itself and its association to the program.
- Any remaining data structures defined to the program are also represented.

Record Analysis

The TEF Export option includes detailed information on COBOL programs. A complete analysis of COBOL records is performed that includes REDEFINES and OCCURS clauses.

To determine file layout, you apply the buffering analysis to each record involved in CRUD activity. The buffering analysis is performed when CRUD activity determines that a TEF entity type should be created for a record.

- Buffering analysis checks each record for a single field element. A single field element can be a single description at the record level, for example, 01 XYZREC PIC X(132). If the record has a single field, direct assignments to and from the record are processed and evaluated. This process continues until a multi-component record is found or no other direct assignments are found.
- TEF objects are created for multi-component records. When multiple TEF entity types are created for a record, each entity type is linked to the same TEF subject area. If no multi-component records are found, TEF objects are created for the original record involved in the CRUD activity.

This is the data the exported field information for COBOL records includes:

- Name
- Length (memory size)
- Number of decimal places
- Domain
- Permitted values (VALUE clause)

Database Entities

The TEF Export option creates a complete representation of DB (database) entities involved in CRUD activity. These are the supported database models:

- IMS (except IMS GSAM DB)
- CICS
- IDMS
- DB2
- COBOL file operations

CRUD Activity/Expected Effects

CRUD activity is generated for all supported DB entities: IMS segments, CICS files, IDMS records, DB2 tables, and COBOL files. These are the expected effects:

- All programs encountered cause a function and elementary process TEF object to be created.
- Only programs with CRUD activity generate expected effects.
- An expected effect is created for each unique DB entity when CRUD activity is detected within the program.
- Each expected effect is linked to the appropriate function and elementary process TEF objects.

The database instructions that determine CRUD activity/expected effects are outlined in these tables for IMS, CICS, IDMS, DB2, and COBOL:

IMS

Database Instruction	CRUD Activity
INSERT	CREATE
GET	READ
REPLACE	UPDATE
DELETE	DELETE

CICS

Database Instruction	CRUD Activity
WRITE	CREATE
READ, READNEXT, READPREV	READ
REWRITE	UPDATE
DELETE	DELETE

IDMS

Database Instruction	CRUD Activity
STORE, MODIFY	CREATE
GET, OBTAIN, MODIFY, FIND	READ
STORE	UPDATE
ERASE	DELETE

DB2

Database Instruction	CRUD Activity
INSERT	CREATE
SELECT, FETCH	READ
UPDATE	UPDATE
DELETE, DROP	DELETE

COBOL

Database Instruction	CRUD Activity
WRITE	CREATE
READ	READ
REWRITE	UPDATE
DELETE	DELETE

Assembler Language Option

Database instruction	CRUD activity
READ	READ
PUT	CREATE (UPDATE)*
GET	READ (UPDATE)*

Database instruction	CRUD activity
PUTX	CREATE (UPDATE)*
WRITE	CREATE (UPDATE)*

* For these activities, the CREATE attribute is set, unless the open macro indicates that the CRUD information should be UPDATE instead. The open parameters, indicating UPDATE instead of CREATE, are INOUT and UPDATE.

Data Model	Description
IMS Data Model	Within the program, CRUD activity involving IMS entities is detected. TEF objects are created for both the IMS entities and local records. Expected effects are generated for the entity type created for the IMS segment.
CICS Data Model	Within the program, CRUD activity involving CICS files is detected. TEF objects are created for both CICS entities and local records. Expected effects are generated for entity types that are created for local records.
IDMS Data Model	Within the program, CRUD activity on IDMS records and logical records is detected. When logical records are used, the component records are resolved and used as the actual record(s) of reference. TEF objects are created for both IDMS records and local records. Expected effects are generated for the entity types created for IDMS records or logical records.
DB2 Data Model	Within the program, CRUD activity involving DB2 Tables is detected. TEF objects are created for both the DB2 entities and local records. Expected effects are generated for the entity types created for DB2.
COBOL File Operations Data Model	Within the program, CRUD activity involving COBOL file operations is detected. TEF objects are created for both the file and the records. Expected effects are generated for the entity types created for local records.
Control Flow Model	The TEF Export option provides control flow model information for both batch and online processes. Batch information is determined by JCL jobs, JCL procedures, and JCL steps. Online information is determined by transactions and screens. When analysis reaches the program level, program control flow is established (see " Program Control Flow " on page 311).

For each exported application, three business systems are created to represent the control flow model:

- `BATCH JOB application name`—This business system is a representation of the batch jobs defined in the application.
- `BATCH PROC application name`—This business system is a representation of the JCL procedures executed by the jobs defined in the application.
- `ON-LINE application name`—This Business System is a representation of the IMS, CICS, and IDMS online activities defined in the application.

Note: _____

IMS BMP applications fall under the appropriate representation for batch activities.

Batch Control Flow

Batch control flow starts with each JCL job defined to the application.

To establish Batch control flow

- 1** From the JCL job, control flow goes to each of the JCL steps. A procedure is created for the JCL job.
- 2** From each JCL step, control flow goes to the executed program or executed JCL procedure.
 - A procedure step is created for the program executed by the JCL step; the procedure step is associated to the procedure created for the JCL job.
 - Appropriate TEF objects are created for the program, as determined by the program control flow (see ["Program Control Flow" on page 311](#)).
 - If the step executes a JCL procedure, a dialog flow object is created to the procedure step that was created for the JCL procedure.
- 3** From each JCL procedure, the control flow goes to each step within the procedure. A procedure and procedure step are created for the JCL procedure.
- 4** From each procedure step, the control flow goes to the executed program or executed JCL procedure. Control flow at the program level is covered in ["Program Control Flow" on page 311](#). The steps within the JCL procedure are processed in the same manner as steps for a JCL job, except that the procedure steps created are associated to the procedure created for the JCL procedure.

This process is repeated until all control flow paths are represented.

Online Control Flow

Online control flow starts with transactions or screens. TEF objects are created to represent the physical screen display.

To establish online control flow

- 1 From a transaction, control flow goes to the executed program.
- 2 From a screen, control flow goes to other screens and to transactions.
- 3 From a program, control flow goes to transactions and screens invoked by the program (see ["Program Control Flow" on page 311](#)).

Screens Supported

These are the screens supported by the TEF Export option:

- BMS maps (CICS)
- IMS formats
- IDMS maps

For all screens (except IDMS Maps), information on the physical display attributes of the fields is generated. This field information is supplied in TEF Screen Literal objects, and includes this data:

- Row
- Column
- Literal value (if any)
- Field length
- Color
- Intensity
- Highlight

Invoking a Screen From Multiple Programs

When a screen is invoked by a program, a procedure step (TEF object) is created to represent the screen. The procedure step is linked to the procedure that contains the invoking entity. These are the procedure rules:

- A procedure step can be linked to only one procedure.
- If the screen is used by more than one procedure:
 - One procedure and one procedure step are created for the screen.
 - All procedures using the screen have dialog flows created from their respective procedure steps to the procedure step created for the screen.

- This information applies to these subsections:
 - IMS online control flow
 - CICS online control flow
 - IDMS online control flow

IMS Online Control Flow

IMS formats are referenced generically within programs, but can have multiple device definitions (with multiple pages) for each device. Because of the multiple device definitions with multiple pages, a series of procedure steps are created for each IMS format. These are the steps of the procedure created for each IMS format:

- Each MOD (Message Output Description) within the format is represented by a procedure and procedure step. The procedure step created for the MOD is the origin or destination for all control flows represented by dialog flows.
- Within the MOD definitions, each DOF (Device Output Format) is searched for device definitions.
- A procedure step represents each page within the device definitions. The DFLD (Defined Field) definitions within the device definition are used to create the screen literal information.
- Pages are associated to the procedure created for the MOD. Each device/page is given a name in this format:

`<device> "PAGE" <page number>`

IMS Control Flow

IMS control flow is controlled by the formats and transactions defined to the application. Control flow is generated on these relationships:

IMS-FORMAT EXECUTES IMS-TRANSACTION

IMS-FORMAT EXECUTES IMS-FORMAT

```
IMS-TRANSACTION (relationship) PROGRAM  
  
PROGRAM EXECUTES IMS-FORMAT  
  
PROGRAM EXECUTES IMS TRANSACTION
```

Note: _____

This is the relationship in the AKR:

```
IMS-TRANSACTION EXECUTES LOADMODULE  
LOADMODULE HAS CSECT  
CSECT IS PROGRAM
```

- IMS transactions and formats (as listed above) are represented by procedures.
- The program executed by the IMS transaction becomes the procedure step for the transaction (except in cases explained in ["Program Control Flow" on page 311](#)).
- The relationships listed above and the program control flow analysis are used to establish links between procedure steps.

CICS Online Control Flow

CICS online control flow is based on BMS map invocation, transaction execution on the program level, and on the transactions defined to the application.

- Each transaction is represented by a procedure.
- The program executed by the transaction:
 - Becomes the procedure step, unless it invokes a BMS map.
 - If the program executed invokes a BMS map, the BMS map is represented by a procedure step and the program is represented by a Common Action Block.
- A CALLs association is established to link the procedure step for the BMS map to the Common Action Block for the program.

Program control flow is followed to establish links between procedure steps.

IDMS Online Control Flow

IDMS online control flow is based solely on IDMS map invocation.

- Program control flow, if transferred to other programs using the IDMS map, creates a dialog flow object.
- The dialog flow object links the procedure steps created for the IDMS map.

Program Control Flow

When any form of control flow (that is, batch or online) reaches a program, detailed external control flow analysis of the program is represented by TEF objects and relationships. Control flow from the program to screens, transactions, and other programs is established.

TEF Objects Created for Programs

Depending on the activity of the program represented, the TEF object created may be a procedure, procedure step, or Common Action Block.

Procedure Step and Procedure

- Procedure steps are always created for programs that contain these activities:

PROGRAM EXECUTES CICS-TRANSACTION

PROGRAM EXECUTES IMS-TRANSACTION

PROGRAM EXECUTES IMS-FORMAT

- Procedure steps are always created for programs that are the *target* of these activities.

CICS-TRANSACTION EXECUTES PROGRAM

IMS-TRANSACTION (EXECUTES) PROGRAM

JCL STEP (EXECUTES) PROGRAM

Note: _____

This is the (EXECUTES) relationship in the AKR:

```
<ENTITY> EXECUTES LOADMODULE  
LOADMODULE HAS CSECT  
CSECT IS PROGRAM
```

- The procedure step created is linked to the procedure that initiated the control flow path to the program, unless more than one control flow path reaches the program.
- If more than one control flow path reaches the program, a procedure is also created for the program.
 - The procedure step created for the program is associated to the procedure created for the program.
 - Dialog flow is established from the procedure steps, in all invoking control flow paths, to the procedure step created for the program.

Common Action Block

All other known program activities (including batch execution) create a Common Action Block to represent the program. Invoking TEF objects have a CALLs association to the Common Action Block created for the program.

Procedure Step Links

You can execute several programs from a procedure step object before another procedure step object is encountered.

- Each intermediate program is represented by a Common Action Block with CALLs association to subsequent programs.
- When a procedure step object is encountered, a dialog flow is created from the encountered procedure step to the procedure step of origin.

Program Control Flow Path

These control flow paths are evaluated with Program Control Flow analysis:

PROGRAM EXECUTES IMS-TRANSACTION	IDMS DC RETURN
PROGRAM EXECUTES CICS-TRANSACTION	IDMS TRANSFER CONTROL
EXEC CICS XCTL	BMS MAP INVOCATION
EXEC CICS LINK	IDMS MAP INVOCATION
COBOL CALL	PROGRAM EXECUTES IMS-FORMAT

Online TEF Export Procedure

To export application data to TEF

- 1 Start Alliance using the procedure defined for your site.
- 2 Select File ► Open application and press Enter. The AMF panel displays as the default.
- 3 Select File ► Export application and press Enter.

- 4 Select File - Export Definition pop-up ► Export Data and press Enter. The Data Export pop-up (see [Figure 91](#)) displays.

Figure 91 • Data Export Pop-up

```

                                Data Export
Command ==> -----
Specify options and jobcard information. Then press PF key for action.

WARNING! Selecting "Edit JCL" or "Submit JCL" will delete any existing
export files created under this userid.

Export Type --  1. Flat Files                DB2 Subsystem: ----
                2. DB2 Load
                3. COOL:Gen Export

Job statement information:
//NAME          JOB (ACCOUNT),NAME,
//              MSGCLASS=A
//*  INSERT /*ROUTE PRINT NODE.USER' HERE IF NEEDED.
//*

                                PF4=COOL:Gen Export Options  PF5=Edit JCL  PF6=Submit JCL
  
```

- 5 In the Export Type field of the Data Export pop-up, type 3 to select TI-Export.
- 6 Under Job statement information, enter the information required for your site.
- 7 To edit the generated JCL file before submitting the export job, press the Edit JCL key.
- 8 Add export parameters to the parm string specified for the extract step, as required (Exec Pgm = VIABTIEF). For a definition of the available export parameters, see ["TEF Export Parameters" on page 314](#). Add these parameters after the existing application parameter. Include export parameters in any order, as shown:


```

      PARM=('APPL=applicationname,NOWORKSET,NOPERMVAL,NOBSD,LEN254')
      
```
- 9 After editing the JCL, submit the job using the TSO/ISPF SUBMIT command.
- 10 To execute the export request without editing the generated JCL, press PF6 to submit the JCL.

Note: _____

If an unresolvable entity is detected during the export, a message is sent to the VIALOG file. You should review the VIALOG file after each export to identify any entities not represented with corresponding TEF objects.

TEF Export Parameters

Add these TEF export parameters to the parm string specified for extract step (Exec Pgm = VIABTIEF):

Parameter	Description
NOWORKSET	Suppresses the generation of data model information, SUBJECT, HLENT, ATTRIBUTE, etc., for records that are not involved in CRUD activity.
NOPERMVAL	Suppresses the generation of the permitted values property for attributes.
NOBSD	Suppresses the generation of the control flow model information, such as Business System Design (BSD).
LEN254	Specifies a maximum length of 254 bytes, for any textual field in the model.

A

Add Impact List Entity screen 78, 202
affiliation, synonyms 35
AKR
 allocating 181
 allocating without ISPF 184, 186
 allocation procedure 181
 considerations 181
 expanding without ISPF 184, 188
 management 181
 renaming without ISPF 188
 unload output entity files 207
 VIASAKRA JCL 184
aliasing entities, in queries 66
Alliance
 accessing from ESW screen xi
 description viii
 linking xi
analysis
 detailed impact analysis 85
 generating a base impact analysis 82
 generating a summary impact list 84
analysis errors 8
analyzing the application 7
application
 boundaries 5
 unload file formats 207
application definition
 create a template procedure 192
 refining 8
Application Knowledge Repository 4
application-level synonyms 33, 39
application-level synonyms, Impact
 Generate Options 97
ATTRIBUTES BLOCK
 DATAITEM entity kind attribute 57
 DSN Entity Attributes 61
 FORMAT BLANK WHEN
 REPEATED attribute 56
 POSITION attribute 57
 qualifying entities 66
 SELECT attribute 54

 SORT attribute 56
 syntax rules 53
 TITLE attribute 55
 WIDTH attribute 55

AutoChange
 accessing from ESW screen xi
 description viii

B

base impact target options
 base selection options, list 99
 children of base selection options 101
 parents of base selection options 100
base selection entities
 parents/children 87
base selection list
 impact generate option 97
 rebuilding 112
 refining 14
Batch control flow, TEF 307
become 36
Bridge
 accessing from ESW screen xi
 description viii

C

Center, description viii
CICS data model 306
COBOL data model 306
collapsing a path, detailed impact
 analysis 95
components
 discovered, missing, dead, overview 8
 homonyms, synonyms 9
 listing relationships 13
 open impact list, adding 111
 removing from open impact list 111
 synonyms, parents/children 36
continuous search path in a query 51
control flow model, TEF 306
conventions page xv
cross-reference reports 9, 26

CRUD activity/expected effects 304
custom queries
 creating 67
 defining 47
 executing 71

D

Data Export pop-up 193
data model, TEF 302
database entities, TEF 303
DATAITEM entity kind attribute
 ATTRIBUTES BLOCK 57
DB2 data model, TEF 306
DB2 entity tables 235, 237–242
DB2 relationship tables 242–247, 249–265,
 267–274, 276–301
dead components 8
default options, impact generate 12, 14
defaults
 ATTRIBUTES BLOCK 52
 detailed impact generate options 104
 in impact generate options 98
 resetting default detailed impact
 generate options 109
 resetting default impact generate
 options 109
definition blocks 11, 48
Detail Information pop-up
 displaying 95
 information shown format 108
detailed impact analysis
 expanding/collapsing a path 95
 generating 93
 understanding 87
detailed impact generate options
 changing 109
 display options 105
 information shown options 105
 resetting defaults 109
Detailed Impact Generate Options
 pop-up 104
Detailed Information pop-up 93
discovered components 8
display options, detailed impact generate
 options 105
display, Detail Information pop-up detailed
 impact analysis 95
DSN entity attributes 61

E

Encore
 accessing from ESW screen xi
 description ix

entities
 aliasing 66
 qualifying 65
 specific, searching for 72
entity relationships 89
ENTITY_RELATIONSHIP_BLOCK
 qualifying entities 66
 syntax rules 51
entityfiles 207–208, 210–215
Estimate
 accessing from ESW screen xi
 description ix
ESW
 description vii
 invoking products x
 product integration xi
ESW - AKR Utility pop-up 182
existing impact list, modifying 110
existing query source file, editing 70
Existing System Workbench (ESW) 4, 181
expanded impact target options
 base impact target options 102
 children of base impact target
 options 103
 parents of base impact target
 options 103
expanded impact targets, description 88
expanding a path, detailed impact
 analysis 95
expanding an AKR without ISPF 188
export facility
 comma delimited files 193
 file formats 207
 output format 207
export facility JCL 195
export parameters, TEF 314
exported application definition
 keyword control statements 190

F

File - AKR Allocate/Expand pop-up with a
 VSAM AKR 183
File - Export Definition pop-up 191
File - Open Application pop-up 76
flow of data, evaluating 81
Format 108
FORMAT BLANK WHEN REPEATED
 attribute 56
full analysis 7

G

Generate Impact Export for
 ASG-AutoChange screen 204

H

hierarchical relationships
 in impact generate options 98
 homonyms 42

I

IDMS data model, TEF 306
 Impact analysis results - Impact Facility 85
 Impact Entities Selection Criteria
 pop-up 202
 Impact Entities Selection Criteria screen 77
 impact entity list 10
 Impact Entity List pop-up 78
 Impact Entity List screen 201
 impact facility process 82
 impact generate options
 base impact target options 99
 changing 14
 changing existing 108
 Detailed Impact Generate Options
 pop-up 104
 expanded impact target options 102
 resetting defaults 109
 specifying new 108
 Impact Generate Options screen 84, 98, 203
 impact list
 adding components 111
 building 12, 75
 creating 76
 instructions, building 75
 modifying existing 110
 removing components 111
 saving updated 112
 impacted components and relationships 13
 implied dependencies, synonyms 38
 import facility 189
 importing, application definition 189
 IMS data model, TEF 306
 incremental analysis 7
 indirect dependencies 37
 information display options 92
 Insight
 accessing from ESW screen xi
 description ix
 using analysis functions xi

J

JCL Edit facility 205
 JCL INCLUDEs 125

K

keyword control statements
 exported application definition 190

L

LANGUAGE attribute 62
 list file, allocating 19
 listing relationships of impacted items 13
 literal strings, searching for 74
 local assignments, in impact generate
 options 97
 local synonym query 72
 log file, allocating 19
 Log Name Customization pop-up 21

M

MEMBER, JCL INCLUDEs 125
 missing components 8
 multiple paths, in a query 51

N

New Impact pop-up 201
 new query, defining 70
 New Worklist pop-up 204

O

online control flow, TEF 308
 open impact list
 adding components 111
 removing components 111
 options
 impact generate 12, 14
 information display 92
 Options - Log/List Definition pop-up 20
 Options - Log/List Name Customization
 pop-up 21
 Options - PF Key Definition pop-up with
 defaults 23
 Options - Product Allocations pop-up 19
 Options - Product Parameters pop-up 18
 Options - Script File Allocations pop-up 22

P

parameters, synonyms 35
 PF Key Definition pop-up 23
 PF keys, viewing current settings 23
 POSITION attribute 57
 product integration xi
 program control flow, TEF 311
 program cross-reference reports 9, 30
 list 69
 program-level synonyms 33

Q

- qualifying entities, in queries 65
- queries
 - aliasing entities 66
 - continuous/new search path 51
 - custom 11, 67, 71
 - defining new 70
 - local synonym 72
 - multiple paths 51
 - qualifying entities 65
 - standard, defined 9
- query and search results, saving/printing 43
- query language
 - ATTRIBUTES BLOCK 53
 - ENTITY_RELATIONSHIP BLOCK 50
 - new source files 67
 - SELECTION BLOCK 64
 - syntax 48
 - understanding syntax 48
- query results 43, 49
- query source file
 - creating 70
 - modifying 69
- query structure 48
- query, syntax rules 49

R

- rebuilding base selection list 112
- Recap
 - accessing from ESW screen xi
 - description ix
- record analysis, TEF 303
- redefinitions 36
- referential symbols 91
- refining the base selection list 14
- related application components 4
- relationship files 216
- relationships
 - entity 89
 - synonym 32
- renames 36
- renaming an AKR without ISPF 188
- reports
 - cross-reference 26
 - program cross-reference 9
 - system cross-reference 9, 26
 - VIAAPRT 6
 - VIALOG 8
- resetting defaults
 - detailed impact generate options 109
 - impact generate options 109
- root entity 51

S

- Script File Allocations pop-up 22
- search path, continuous, new 51
- search results, saving/printing 43
- searches 9
 - literal strings 74
 - running 32
 - specific 11
 - specific entities 72
- SELECT attribute 54
- SELECT clause, syntax 54
- selecting items
 - base selection list 12
 - for analysis 82
- SELECTION BLOCK 63
- SELECTION BLOCK, syntax rules 64
- selection criteria, impact list 10
- semantic linking 7
- semantic linking, referential symbols 91
- SmartDoc
 - accessing from ESW screen xi
 - description ix
- SmartEdit
 - accessing from ESW screen xi
 - description x
- SmartQuest
 - accessing from ESW screen xi
 - description x
- SmartTest
 - accessing from ESW screen xi
 - description x
- SORT attribute 56
- source files
 - creating 70
 - program cross-reference, list 69
 - query language, editing 70
 - query language, modifying 69
 - query language, new 67
 - system cross-reference, list 68
- specific entities, searching for 11, 72
- Summary Impact - Select Impact Generation Entities screen 203
- summary impact analysis 12-13
- summary impact analysis, generating 84
- synonyms 36
 - application-level 39
 - assignment 34
 - assignments 87
 - calculating 33
 - implied dependencies 38
 - indirect dependencies 36
 - parameters, actual and formal 35

- parents/children 36
- relationships 32
- utility program 39
- synonyms, affiliation 35
- syntax, query language 48
- system cross-reference
 - reports 9, 26
 - source files, list 68

T

- TEF 301
 - Batch control flow 307
 - COBOL file operations data
 - model 306
 - control flow model 306
 - CRUD activity/expected effects 304
 - data model 302
 - database entities 303
 - DB2 data model 306
 - export parameters 314
 - IDMS data model 306
 - IMS data model 306
 - online control flow 308
 - program control flow 311–312
 - record analysis 303
 - TEF objects 302
- TEF export parameters 314
- TITLE attribute 55
- transition enablement facility 301

U

- unload file formats, export facility 207
- user options
 - Log/List definition 19
 - PF keys 23
 - Script File Allocations 22
- utility programs, synonym definition 39

V

- valid entities and relationships 115
- valid relationships 116
- VIAAPRT 6
- VIABXDF2 198
- VIALOG 8

W

- WIDTH attribute 55
- worklist 81

Z

- zooming in/out 95

ASG Worldwide Headquarters Naples Florida USA | asg.com