

# Everything You Always Wanted to Know about IDCAMS But Were Afraid to Ask

in partnership with



Keith Winnard

Stephen M. Branch



**z Systems**





International Technical Support Organization

**Everything You Always Wanted to Know about IDCAMS  
But Were Afraid to Ask**

October 2016

**Note:** Before using this information and the product it supports, read the information in “Notices” on page v.

**First Edition (October 2016)**

This document was created or updated on October 7, 2016.

**© Copyright International Business Machines Corporation 2016. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	v
Trademarks .....	vi
<b>IBM Redbooks promotions</b> .....	vii
<b>Preface</b> .....	ix
Authors .....	ix
Acknowledgements .....	x
Now you can become a published author, too! .....	x
Comments welcome .....	xi
Stay connected to IBM Redbooks .....	xi
<b>Chapter 1. Live at SHARE: Innovation through collaboration</b> .....	1
1.1 A question can lead you anywhere .....	2
1.2 Producing this publication .....	2
1.2.1 SHARE member benefits .....	2
1.2.2 SHARE benefits .....	3
1.2.3 IBM Redbooks benefits .....	3
1.2.4 Collective benefits .....	3
1.3 How this collaboration works .....	3
1.3.1 Preparation .....	4
1.3.2 Marketing .....	4
1.3.3 The event .....	5
1.3.4 Collateral and review .....	7
<b>Chapter 2. Introduction to IDCAMS</b> .....	9
2.1 IDCAMS overview .....	10
2.1.1 IDCAMS functions .....	10
2.1.2 Starting IDCAMS .....	10
2.2 IDCAMS commands .....	12
2.2.1 Functional commands .....	12
2.2.2 Modal commands .....	14
2.2.3 Comments .....	15
<b>Chapter 3. Setting up and maintaining data sets scenario</b> .....	17
3.1 Project scenario overview .....	18
3.1.1 Your role in a new project .....	18
3.1.2 Application development environment .....	18
3.2 Setting up the APP3 project .....	19
3.2.1 Creating the APP3 user catalog .....	19
3.2.2 Creating the APP3 alias .....	20
3.2.3 Creating the APP3 data sets .....	22
3.3 Maintaining the APP3 project data sets .....	30
3.3.1 Copying data sets .....	30
3.4 Obtaining information about the APP3 data sets .....	36
3.4.1 Using the LISTCAT command .....	36
3.4.2 Using the PRINT commands .....	37
3.4.3 Checking the KSDS structural integrity .....	38
3.5 Removing the APP3 project .....	39

3.5.1 Delete command . . . . .	39
3.6 Useful things to know . . . . .	40
3.6.1 Missing comments in your SYSPRINT . . . . .	40
3.6.2 DYNAMBR parameter . . . . .	42
3.6.3 UNIQUE KEY . . . . .	43
3.7 Provisioning jobs . . . . .	44
3.7.1 JES2 JOBGROUPs . . . . .	44
3.7.2 Using a procedure with instream SYSIN and variables . . . . .	51
3.8 Provisioning . . . . .	54
3.9 Thank you . . . . .	54
<b>Related publications . . . . .</b>	<b>55</b>
IBM Redbooks . . . . .	55
Other publications . . . . .	55
Online resources . . . . .	55
Help from IBM . . . . .	55

# Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

AIX®	RACF®	z Systems™
Destination Z™	Redbooks®	z/OS®
IBM®	Redpaper™	
IBM z Systems™	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

Other company, product, or service names may be trademarks or service marks of others.



## Find and read thousands of IBM Redbooks publications

- ▶ Search, bookmark, save and organize favorites
- ▶ Get personalized notifications of new content
- ▶ Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App



## Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



[ibm.com/Redbooks](http://ibm.com/Redbooks)  
About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

# Preface

This IBM® Redpaper™ publication is different from others you have read because it was not written in the conventional way with a residency and a handful of authors. It was written by people who want to make a difference.

The IBM Redbooks® organization and SHARE teamed up to give all of the attendees at the SHARE 2016 conference that was held in Atlanta, US, July 31 - August 5 the opportunity to contribute their thoughts and ideas about the latest IDCAMS capabilities. Many discussions arose about the subject and related techniques.

This Redpaper publication is the result of the following activities:

- ▶ The “Everything You Wanted To Know About IDCAMS But Were Afraid To Ask” session that was held on Wednesday at 8:30 AM.
- ▶ Techtalk sessions at the SHARE booth.
- ▶ Discussions at the IBM Redbooks publications booth.
- ▶ Discussions that were held at the SHARE booth Influence area.
- ▶ Discussions in coffee lounges.

This Redpaper publication features the following chapters:

- ▶ Chapter 1, “Live at SHARE: Innovation through collaboration” on page 1, describes how the event went from idea to completion.
- ▶ Chapter 2, “Introduction to IDCAMS” on page 9, presents a basic overview of IDCAMS.
- ▶ Chapter 3, “Setting up and maintaining data sets scenario” on page 17, guides you through a provisioning scenario that uses IDCAMS.

On behalf of everyone who took part in this project, we hope you enjoy the collective thoughts of many people who were so willing to help to increase your expertise.

## Authors

This paper was produced as a collaboration of specialists from around the world at the SHARE Conference that was held in Atlanta, US, in 2016. The following authors captured the information:

**Keith Winnard** is the IBM z/OS® Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and is keen to engage with customers to understand what they want from IBM Redbooks publications. Before joining the ITSO in 2014, Keith worked for clients and Business Partners in the UK and Europe in various technical and account management roles. He is experienced with blending and integrating new technologies into the traditional landscape of mainframes.

**Stephen M. Branch** is an IBM Senior Software Engineer whose 40-year career includes all aspects of DFSMS. Stephen specializes in ICF Catalog, IDCAMS, and VSAM. He holds several patents for these components and is the author of the Catalog Search Interface (CSI) and many ICF Catalog and VSAM functions. Stephen has a Master of Science Degree from Louisiana State University.

## Acknowledgements

Thanks to the following people for entering the spirit of the collaboration and making this initiative a success:

- ▶ **Donna J Hudi** (Executive Director, SHARE)
- ▶ **Ed Jaffe** (SHARE)
- ▶ **Martha M McConaghy** (SHARE)
- ▶ **Skip Robinson** (SHARE)
- ▶ **James Vincent** (SHARE)
- ▶ **Harry Williams** (SHARE)
- ▶ **Russell Witt** (SHARE)
- ▶ **Jim Erdahl** (Systems Architect, US bank)
- ▶ **Bruce Koss** (z/OS Systems Engineer, Wells Fargo)
- ▶ **Marty Hasegawa** (Senior Software Engineer, Rocket Software)
- ▶ **Ray Mullins** (Senior Software Developer, Phoenix Software International)
- ▶ **Gary Puchkoff** (Senior Technical Staff Member, IBM)
- ▶ **Barbara White-McDonald** (DFSMS Product Strategist, IBM)
- ▶ **Bob Haimowitz** (DSG, IBM)
- ▶ **Deana Coble** (DSG, IBM)
- ▶ **Martin Keen** (DSG, IBM)
- ▶ **Jim Pistilli** (DSG, IBM)
- ▶ **Chris Rayns** (DSG, IBM)
- ▶ **Tom Wasik** (JES2 Development, IBM)
- ▶ **Bill White** (DSG, IBM)
- ▶ **Debbie Willmschen** (DSG, IBM)
- ▶ **Tara Woodman** (DSG, IBM)

Many thanks to all those people who stopped by the Redbooks publications booth and attended the “Everything You Wanted To Know About IDCAMS But Were Afraid To Ask” session. In addition, to all of those people who joined in the discussions at the Techtalk sessions at the SHARE booth.

## Now you can become a published author, too!

Here’s an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:  
[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- ▶ Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>





# Live at SHARE: Innovation through collaboration

SHARE is an independent, volunteer-run information technology association that provides education, professional networking, and industry influence. The opportunity to partner with the IBM Redbooks publications organization, an acknowledged leader in experiential education, allowed the SHARE membership to collaborate on a technical, user experience publication.

SHARE prides itself on sharing user-led proven practices across the IT ecosystem, which this opportunity demonstrated. The skills and expertise that the IBM Redbooks publications team brought to SHARE allowed us all to quickly produce a technical first draft publication by the end of the event. Historically, a project such as this one often takes weeks (if not months) of dedicated commitment of all involved to complete.

Our members were demonstrated the depth of their technical leadership quickly and easily while they collaborated with their peers within the industry. SHARE is proud to be part of the team that provided this opportunity to our membership.

This IBM Redpaper publication is the result of a live event that was held at SHARE Atlanta, US, in August 2016. If you participated in the making of this publication in any way, we thank you for helping to further the knowledge of others by sharing your thoughts, questions, and ideas. Great things can happen when minds come together with a common goal.

If you are interested in running a similar initiative, this chapter is intended to provide you with useful information about how we produced this publication.

We encourage you to improve on what we produced by sharing your ideas with SHARE and seeing how innovation through collaboration can bring benefits to this long-established community of devotees. For more information about getting involved, see this website:

<http://www.share.org/>

Finally, from all of us who took part in this initiative, we wish you the best of luck.

## 1.1 A question can lead you anywhere

Two people sit opposite each other in a room in Poughkeepsie, NY. Both people returned the previous week from a successful SHARE event in San Antonio, TX. Topics of conversation within the room revolve around IBM z/OS related new enhancements and how IBM Redbooks publications might handle the topics.

The following conversation unfolded:

Keith Winnard (z/OS Project Leader at the International Technical Support Organization): “It’s a question of aligning the target readership with the subject content to provide relevance.”

Gary Puchkoff (z/OS New Technology, IBM Systems): “Agreed, but how do you capture that alignment on a specific topic in a reasonable time period to increase the value?”

Keith “Hmmm, if we could have captured people’s thoughts at SHARE, that would have helped. Then, we could follow up with a publication fairly soon after.”

Gary: “Why not write the book live at SHARE?”

Silence accompanied by chin stroking and hmmmms.

Fast forward 24 hours to a telephone conversation:

Keith: “Gary posed an interesting question yesterday about writing a book live at SHARE.”

Barbara (DFSMS Product Strategist, IBM z™ Systems Software, IBM Systems): “That’s very interesting; I like that.”

Keith: “Yes, it might well appeal to many people.”

Barbara: “Shall we try it?”

And that is how this publication came to be. Well, not quite, many helping hands were available to put this Redpaper publication in front of you.

## 1.2 Producing this publication

After the initial conversation, the SHARE board fully supported the initiative and provided valuable help and guidance. Thus, our work on this publication began.

### 1.2.1 SHARE member benefits

Discussions between various parties occurred and the key value is to try and envisage how SHARE members might benefit.

Consider the following key points:

- ▶ Attendees can ask questions directly of the presenter and (where possible) have the question and answer captured and included in a formal publication.
- ▶ The initiative involves collaboration on a topic that was agreed to by SHARE members and presenters and thus promotes a sense of common purpose.
- ▶ People wanting to be involved in improving the coverage of a topic have the chance to do so.



## 1.2.2 SHARE benefits

Although SHARE is made up of its members, it is also an entity. By exploring innovations, SHARE might realize the following benefits:

- ▶ Provide SHARE with the opportunities to channels through which SHARE can distribute their materials.
- ▶ Provide the opportunity to increase the reach of SHARE into the target readership for SHARE and non-SHARE members.
- ▶ Increase the scope for vendors the opportunity to partake in a new engagement model.

## 1.2.3 IBM Redbooks benefits

In this instance, IBM Redbooks publications are involved in the initiative. However, this involvement might apply to other parties that want to be innovative through collaboration. IBM Redbooks publications might realize the following benefits:

- ▶ Engage with our target readership.
- ▶ Refine the relevance of our materials to our readership.
- ▶ Produce an “up-to-the-minute” publication that is relevant to our target readership’s thoughts.

## 1.2.4 Collective benefits

This pilot initiative brings all three parties together by creating, publishing, and distributing a publication and related supporting collateral to strengthen the value of the SHARE community and beyond.

## 1.3 How this collaboration works

A collaborative process of this nature relies on all parties working together to achieve common goals. This process included the following phases:

- ▶ Preparation
- ▶ SHARE event
- ▶ Publication
- ▶ Collateral

Figure 1-1 shows a high-level outline of the four phases.

Preparation	SHARE Event	Publication	Collateral
<ul style="list-style-type: none"> <li>• <b>Topic Selection</b> <ul style="list-style-type: none"> <li>• Choose a suitable pilot topic</li> <li>• Make it relevant</li> <li>• Use a provisioning based scenario</li> </ul> </li> <li>• <b>Marketing</b> <ul style="list-style-type: none"> <li>• Promo video</li> <li>• Social media plan</li> <li>• Special book cover</li> <li>• Set attendees expectations</li> </ul> </li> <li>• Presenter and authors alignment</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Presentation</b> <ul style="list-style-type: none"> <li>• Attendees encouraged to contribute</li> <li>• ASK by #tag option</li> <li>• Live tweeting</li> <li>• Photographs</li> </ul> </li> <li>• <b>Booths</b> <ul style="list-style-type: none"> <li>• Presence in SHARE and IBM Redbooks booth</li> <li>• Use MVSS blog</li> <li>• Techtalk presentations at SHARE booth</li> </ul> </li> <li>• SHARE and Redbooks regular assessment</li> </ul>	<ul style="list-style-type: none"> <li>• SHARE announces that draft is available for download on Friday morning</li> <li>• Initial social media figures included: <ul style="list-style-type: none"> <li>• Donna and Martin to identify coverage</li> <li>• Blogs issued pointing to pub</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Social media figures collected after event</li> <li>• Downloads of draft monitored</li> <li>• SHARE board approval of IBM Redpaper publication</li> <li>• Second promo video</li> <li>• Blogs to encourage other parties to try it</li> <li>• SHARE and IBM Redbooks publication review of pilot</li> <li>• Download figures to be reported to SHARE</li> </ul>

Figure 1-1 Current high-level schematic

### 1.3.1 Preparation

This initiative is new and an established topic was chosen. Several candidate topics were evaluated. Eventually, the topic that was chosen was IDCAMS. The initial IDCAMS presentation was written, refined, and titled, “Everything You Always Wanted to Know about IDCAMS But Were Afraid to Ask”. The presentation was scheduled for the SHARE event on Wednesday morning at 8:30 AM.

The presentation was divided into the following parts:

- ▶ An introduction to IDCAMS.
- ▶ A simple provisioning, which showed a practical use of IDCAMS in setting up a basic application development environment.

### 1.3.2 Marketing

A promotional video was created by IBM Redbooks publications to be used at the event. The video was passed to SHARE Executive Director Donna J Hudi to use to promote the initiative. The video is available on YouTube:

<https://www.youtube.com/watch?v=nfSjEGdYi5c&feature=youtu.be>

The video was also made available in MP4 format to increase the opportunity to promote the paper.

The IBM Redbooks publications and SHARE social media teams also produced a social marketing campaign to promote the event.

The social coverage consisted of two parts leading up the day of the session and social coverage as the session is happening from the IBM Redbooks publications social channels.

Leading up to the event, IBM Redbooks publications highlighted and emphasized participating in the live IBM Redpaper publication across their mainstream social channels of Twitter, Facebook, LinkedIn (Business and Group), and Google+. These social channels added up to over 82,000 fans and followers across all platforms. The #ShareRedpaper hashtag was created to promote the event.

The content that was shared on these channels was social channels and text in long and short form that announced the session and invited readers to participate in the session. We also coordinated with other IBM groups to mention on their social channels. Groups, such as IBM z Systems™, LinuxOne, Destination Z™, MSP Techmedia, and other IBM social pages were also mentioning the live Redpaper session to increase awareness and visibility.

A Tech Talk session was also arranged at the SHARE booth in the Expo center. While the Tech Talk and main session were happening, IBM Redbooks Project Leader Martin Keen was covering the event on the IBM Redbooks Twitter page, which included tweets that described the class and photos.

SHARE and IBM Redbooks publications also designed a special cover for the paper to include the SHARE logo to emphasize the collaboration.

### 1.3.3 The event

A special slide was created to be included in Monday's keynote speech and to be used at Tech Talk to promote awareness, as shown in Figure 1-2. The slide also was used in other promotional activities at the event.



Figure 1-2 Tech Talk slide

The Tech Talk session on Monday that is shown in Figure 1-3 outlined how IDCAMS can be used to help provision environments. A basic application development project provisioning scenario showed how provisioning can be done.

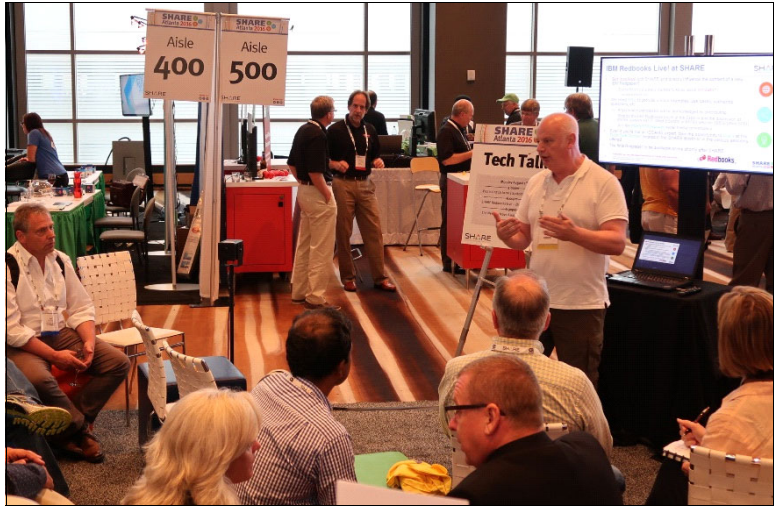


Figure 1-3 Tech Talks session

Many contributions and discussions were captured during the event and added to the IBM Redpaper publication. A few of the topics that were covered required further discussion. These topics were followed up after the event.

The main session on the Wednesday (shown in Figure 1-4) was well-attended, with great questions, suggestions, and discussions interspersed throughout.



Figure 1-4 Wednesday session

The positive response from attendees led to several communications being continued beyond the event to refine some of the points that were raised and to add other points.

The IBM Redpaper publication was published in draft on Friday, August 5 around 5 PM.

### 1.3.4 Collateral and review

This initiative was a pilot. The experience and collateral that was taken from it will be reviewed with SHARE representatives to provide valuable input for future initiatives.

The pilot was successful, and we encourage others to work with SHARE to run similar initiatives and give members the opportunity to share their experiences and expertise in the chosen topic for the benefit of others in the spirit of sharing.

The figures that are listed in Table 1-1 show the promotion and interest for the IBM Redpaper publication from the IBM Redbooks' coverage activities.

*Table 1-1 Redpaper publication promotion and interest*

<b>Description</b>	<b>Value</b>
Downloads of draft Redpaper from 5 PM Aug. 5 - Aug. 27	1085
Impressions from IBM Redbooks tweets on Twitter at SHARE	46,095
Engagements from IBM Redbooks tweets on Twitter at SHARE	1,009
Coverage of live Redpaper publication	82,000

To complete the collaborative efforts, the figures that are listed in Table 1-2 show the promotion and interest for the IBM Redpaper publication from the SHARE coverage activities.

*Table 1-2 SHARE public promotion and interest*

<b>Description</b>	<b>Value</b>
Social reach	1,995
Social impressions	4,689
Clicks	52
Social media interactions	5
Twitter retweets	12
Facebook and LinkedIn likes	60
Social engagements	63
Social shares	4
Attendees that were reached through four onsite daily emails throughout the week of SHARE Atlanta	1,000+

Ultimately, we reached several thousand individuals by using email and numerous social engagements while sharing and interacting with the IBM Redbooks publications posts from the SHARE channels.

The collaboration was a great success and both parties complemented each other to reach out to the attendees and the rest of the SHARE and IBM Redbooks publications communities.





# Introduction to IDCAMS

For the purposes of this chapter, we assume that you are new to IDCAMS. Therefore, after IDCAMS is described, we set up a scenario for you to follow. This scenario is described in Chapter 3, “Setting up and maintaining data sets scenario” on page 17.

This chapter includes the following topics:

- ▶ 2.1, “IDCAMS overview” on page 10
- ▶ 2.2, “IDCAMS commands” on page 12

## 2.1 IDCAMS overview

By using Access Method Services (AMS), you can create and maintain catalogs and several types of data sets. But, how does IDCAMS fit into AMS? IDCAMS is the utility that does all the work for you.

### 2.1.1 IDCAMS functions

IDCAMS helps you to perform the following tasks:

- ▶ Define, delete, alter, copy, and list catalogs, data sets, catalog entries, or tape library entries.
- ▶ Collect data set and Systems Managed Storage (SMS) information, detect catalog problems, detect VSAM data set problems, retrieve Direct Access Storage Device (DASD) information, set cache parameters for DASD devices, and communicate with SMSVSAM.
- ▶ Turn VSAM Record Level Sharing (RLS) on and off.

### 2.1.2 Starting IDCAMS

The following options that can be used to run IDCAMS are shown in Figure 2-1:

- ▶ Job Control Language (JCL)
- ▶ Time Sharing Option (TSO)
- ▶ Problem Program Interface

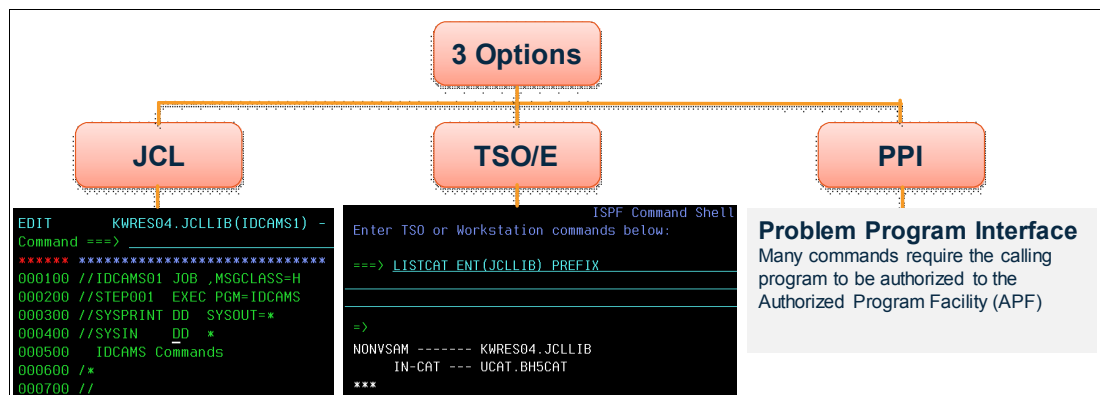


Figure 2-1 Three options for starting IDCAMS

These options are described next.



## JCL option

You can run IDCAMS as a batch job. A sample basic JCL is shown in Figure 2-2.

```
EDIT          KWRES04.JCLLIB(IDCAMS1) - 01.00          Member IDCAMS1 saved
Command ==> _____ Scroll ==> PAGE
***** Top of Data *****
000100 //IDCAMS01 JOB ,MSGCLASS=H
000200 //STEP001 EXEC PGM=IDCAMS
000300 //SYSPRINT DD  SYSOUT=*
000400 //SYSIN     DD  *
000500     IDCAMS Commands
000600 /*
000700 //
***** Bottom of Data *****
```

Figure 2-2 IDCAMS sample JCL

The SYSPRINT DD statement contains the output from the commands that were used.

The SYSIN DD statement is the file that contains your commands. You can write them in-stream (denoted by the “\*”) or you can include the commands in a separate data set and use the SYSIN DD statement to identify the data set, as shown in Example 2-1.

*Example 2-1 SYSIN identifies a data set*

---

```
//SYSIN DD DSN=APP3.IDCAMS.PARMS,DISP=OLD
```

---

## TSO option

The Interactive System Productivity Feature (ISPF) Command Shell is the best option if you want to use the IDCAMS command interactively, as shown in Figure 2-3.

```
ISPF Command Shell
Enter TSO or Workstation commands below:
==> LISTCAT ENT(JCLLIB) PREFIX

Place cursor on choice and press enter to Retrieve command

=>
=>
=>
=>
=>
=>
=>
=>
=>
=>
NONVSAM ----- KWRES04.JCLLIB
IN-CAT --- UCAT.BH5CAT
***
```

Figure 2-3 Using IDCAMS interactively

As shown in Figure 2-3, the IDCAMS **LISTCAT** command is entered at the top of the window and the output appears at the bottom of the window.

**Note:** The PREFIX option on the TSO command was introduced in z/OS Version 2 Release 2. The PREFIX/NOPREFIX was introduced so that the user can add the TSO user ID.

## PPI option

Although your program can use the PPI to start IDCAMS, many commands require the calling program to be authorized to the Authorized Program Facility (APF). This option is beyond the scope of this publication.

## 2.2 IDCAMS commands

In this section, we describe the following types of IDCAMS commands:

- ▶ Functional
- ▶ Modal
- ▶ Comments

### 2.2.1 Functional commands

The functional commands perform an action and are listed in Table 2-1.

Table 2-1 IDCAMS functional commands

Command	Description
ALLOCATE	Allocates VSAM and NONVSAM data sets.
ALTER	Alters attributes of catalogs, data sets, tape library entries, and tape volume entries that were defined.
BLDINDEX	Builds alternative indexes for data sets.
CREATE	Creates tape library entries and tape volume entries.
DCOLLECT	Collects data set, volume usage, and migration utility information.
DEFINE	Defines the following objects: <ul style="list-style-type: none"><li>▶ ALIAS</li><li>▶ ALTERNATEINDEX</li><li>▶ CLUSTER</li><li>▶ GENERATIONDATAGROUP (GDG)</li><li>▶ NONVSAM</li><li>▶ PAGESPACE</li><li>▶ PATH</li><li>▶ USERCATALOG or MASTERCATALOG</li></ul>
DELETE	Deletes catalogs, VSAM data sets, and non-VSAM data sets.
DIAGNOSE	Scans a basic catalog structure (BCS) or a VSAM volume data set (VVDS) to validate the data structures and detect structure errors.
EXAMINE	Analyzes and reports the structural consistency of an index or data component of a key-sequence data set (KSDS) cluster.
EXPORT	Disconnects user catalogs exports VSAM data sets and catalogs.
IMPORT	Connects user catalogs and imports VSAM data sets and catalogs.
LISTCAT	Lists catalog entries.
PRINT	Prints VSAM data sets, non-VSAM data sets, and catalogs.

Command	Description
REPRO	<p>Performs the following functions:</p> <ul style="list-style-type: none"> <li>▶ Copies VSAM and non-VSAM data sets, user catalogs, master catalogs, and volume catalogs</li> <li>▶ Splits catalog entries between two catalogs</li> <li>▶ Merges catalog entries into another user or master catalog</li> <li>▶ Merges tape library catalog entries from one volume catalog into another volume catalog</li> </ul>
SHCDS	<p>Lists SMSVSAM recovery that is associated with subsystems spheres and controls that recovery. This command works in batch and in the TSO/E foreground. It includes subcommands with which the following tasks can be performed:</p> <ul style="list-style-type: none"> <li>▶ List information that is kept by the SMSVSAM server and the catalog as related to VSAM RLS or DFSMSStvs</li> <li>▶ Take action on work that was shunted</li> <li>▶ Control a manual forward recovery</li> <li>▶ Run critical non-RLS batch window work if necessary</li> <li>▶ Perform a subsystem cold start</li> </ul>
VERIFY	<p>Performs the following functions:</p> <ul style="list-style-type: none"> <li>▶ Causes a catalog to correctly reflect the end of a data set after an error occurred while closing a VSAM data set. The error might cause the catalog to be incorrect.</li> <li>▶ Causes VSAM Record Management VERIFY to back out or complete any interrupted CA reclaim in addition to regular IDCAMS VERIFY functions.</li> <li>▶ When run with a preceding EXAMINE, causes VERIFY to attempt to fix any errors it can that is based on the information that is passed in by EXAMINE. Only the HURBA is corrected if there is concurrent access on the data set when VERIFY is being run.</li> <li>▶ When run with a preceding EXAMINE, causes VERIFY to attempt to fix any errors it can that based on the information that is passed in by EXAMINE. Only the HURBA is corrected if there is concurrent access on the data set when VERIFY is being run.</li> </ul>

Commands are separated from their parameters by one or more separators (blanks, commas, or comments). For some parameters, parentheses are used as separators.

After each function command completes, a condition code is generated to notify you of the result of your command. The condition codes are listed in Table 2-2.

*Table 2-2 Functional command condition code summary*

Code	Description
0	The function executes as directed and expected.
4	A problem occurred in executing the complete function, but it continued.
8	A requested function was completed, but major specifications were unavoidably bypassed.
12	The program cannot perform requested function.
16	A severe error occurred that erased the remainder of the command stream.

## 2.2.2 Modal commands

The condition codes that are generated by the function commands can be tested by the IF-THEN-ELSE sequence. You might want to conditionally run other commands, depending on the result of the previous functional commands. In the case of multiple functional commands, you can use the modal commands to apply logic to functional command flow.

**Note:** Modal commands are not available when the TSO option is used.

The IF-THEN-ELSE sequence is listed in Table 2-3.

Table 2-3 IF-THEN-ELSE sequence

Command	Parameters
IF	{LASTCC MAXCC} operator number THEN[ command   DO command set END] [ELSE[ command   DO command set END]]

LASTCC refers to the condition code that results from the preceding function command that is compared.

MAXCC specifies the maximum condition code value that is established by any previous function command or by using a SET command that is compared.

The six available operators are listed in Table 2-4.

Table 2-4 Operator expression

Comparison	Expressed as
Equal to	= or EQ
Not equal to	≠ or NE
Greater than	> or GT
Less than	< or LT
Greater than or equal to	>= or GE
Less than or equal to	<= or LE

Examples to test condition codes are shown in Example 2-2.

Example 2-2 Operator examples

---

```
Test if the last condition code was 0
IF LASTCC = 0
Test if the last condition code is greater than 4
IF LASTCC GT 4
Test if the MAXCC value is less than 12
IF MAXCC LT 12
```

---

## SET command

The **SET** command changes or resets a defined condition code. This change is made because a common technique (which is described in Chapter 3, “Setting up and maintaining data sets scenario” on page 17) is to include a **DELETE** command before a **DEFINE** command for the same data set.

If you want to **DEFINE** a VSAM data set, you might want to check whether it exists before you define it. If you include a **DELETE** command for the data set first, the use of the **DELETE** command removes the data set if it exists (assuming it is not being used by someone else) and then, your **DEFINE** command likely is okay.

However, if the use of the **DELETE** command did not find the data set, it issues a non-zero return code. You can then reset the return code and continue, as shown in Example 2-3.

### *Example 2-3 SET example*

---

```
DELETE your data set
SET MAXCC=0
DEFINE your data set
```

---

You can end all processing by setting MAXCC or LASTCC to 16.

## 2.2.3 Comments

Comments are important because they indicate what you are trying to achieve. Check your local policies to determine whether there are guidelines for including comments.

Comments are strings of characters that are surrounded by `/*` and `*/`. Comments can contain any characters except `*/`. Commands can begin at, or to the right of, the left margin. For batch processing jobs, the default margins are 2 and 72.

Comments must be kept current, accurate, and aligned with the functional and modal commands so that they are useful to other users. A sample comment and command flow is shown in Example 2-4.

### *Example 2-4 Sample use of comments*

---

```
//RECOVERY EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//BACKUP DD DISP=OLD,DSN=NEWCUST.APPS.BACKUP
//NCAPPS DD DISP=SHR,DSN=NEWCUST.APPS
//SYSIN DD *
/* APPLICATION RECOVERY SEQUENCE */
/* DELETE THE TEMP LOG1 AND TEMP LOG 2 DATA SETS */
/* RE LOAD THE NEW CUSTOMER APPLICANTS FILE */
/* RESTART THE APPLICATION. */
/* +++ IF THE PROGRAM FAILS AFTER THE RELOAD THEN +++ */
/* +++ ISSUE CALL OUT TO THE APPLICATION SUPPORT TEAM +++*/
DELETE TEMP.LOG1
DELETE TEMP.LOG2
SET MAXCC=0
REPRO INFILE(BACKUP) OUTFILE(NCAPPS)
/*
```

---

The recovery job deletes temporary log files and restores the customer new applicants file from a backup. The comments direct that the application program is run again.

If the return fails, the problem might be an application program issue; therefore, the comments can indicate that a call out to the application support group must be made for more support.



## Setting up and maintaining data sets scenario

Now that you are familiar with IDCAMS, you can work on a hypothetical scenario to exercise your new found expertise.

This chapter provides a sample provisioning scenario and guides you through how to complete all of the tasks in the scenario. Your scenario is based on setting up and maintaining data sets for a small application during its development cycle.

The following components are described:

- ▶ Your requirements
- ▶ The environment
- ▶ Set up commands
- ▶ Maintenance and reporting tasks
- ▶ Job packing considerations

Although the initial base scenario was designed by the authors, the scenario was expanded by the SHARE attendees of SHARE 2016 event in Atlanta.

This chapter includes the following topics:

- ▶ 3.1, “Project scenario overview” on page 18
- ▶ 3.2, “Setting up the APP3 project” on page 19
- ▶ 3.3, “Maintaining the APP3 project data sets” on page 30
- ▶ 3.4, “Obtaining information about the APP3 data sets” on page 36
- ▶ 3.5, “Removing the APP3 project” on page 39
- ▶ 3.6, “Useful things to know” on page 40
- ▶ 3.7, “Provisioning jobs” on page 44
- ▶ 3.8, “Provisioning” on page 54
- ▶ 3.9, “Thank you” on page 54

## 3.1 Project scenario overview

In this section, you learn how to use some of these commands in the scenario we created.

### 3.1.1 Your role in a new project

You are assigned tasks to provision and provide maintenance jobs for an application development project that is named APP3.

You have the following responsibilities:

- ▶ Set up the data sets for a new project. Consider the following points:
  - The project data sets must be in their own user catalog.
  - The project requires VSAM data sets, alternative indexes, and non-VSAM data sets.
- ▶ When the development cycle competes, you must connect the project data sets into the test.
- ▶ You must develop the IDCAMS statements to create, back up, delete, change, and print the data sets and maintain and list the project's user catalog.

### 3.1.2 Application development environment

The application development environment features the following application projects:

- ▶ APP1: A project that is underway and not your responsibility.
- ▶ APP2: A project that is underway and not your responsibility.
- ▶ APP3: The new project to which you were assigned.

The environment is shown in Figure 3-1.

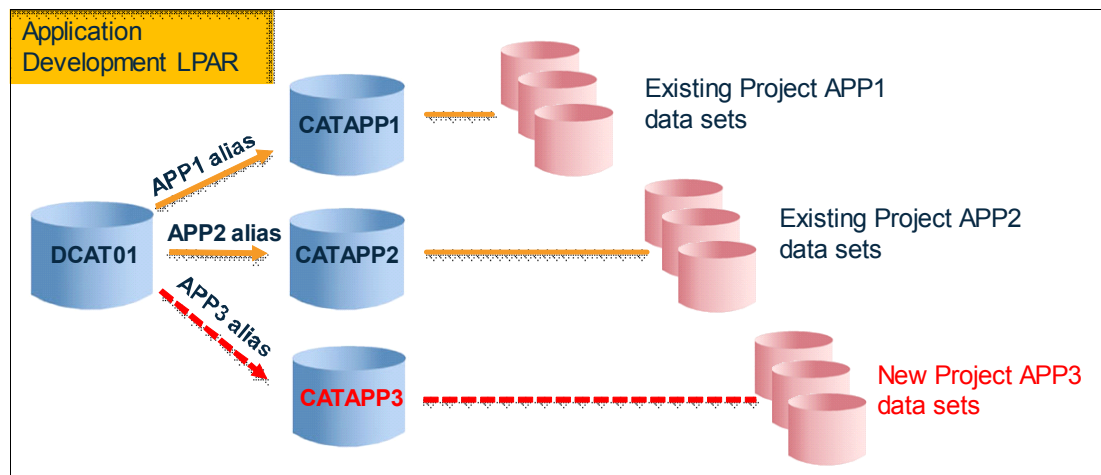


Figure 3-1 Scenario environment

The APP3 project is self-contained. It includes its own recognizable user catalog and unique high-level qualifier for the data sets. These attributes make it easily portable.



## 3.2 Setting up the APP3 project

The scenario is divided into six steps. An overview of the tasks you must complete to support the project is shown in Figure 3-2.

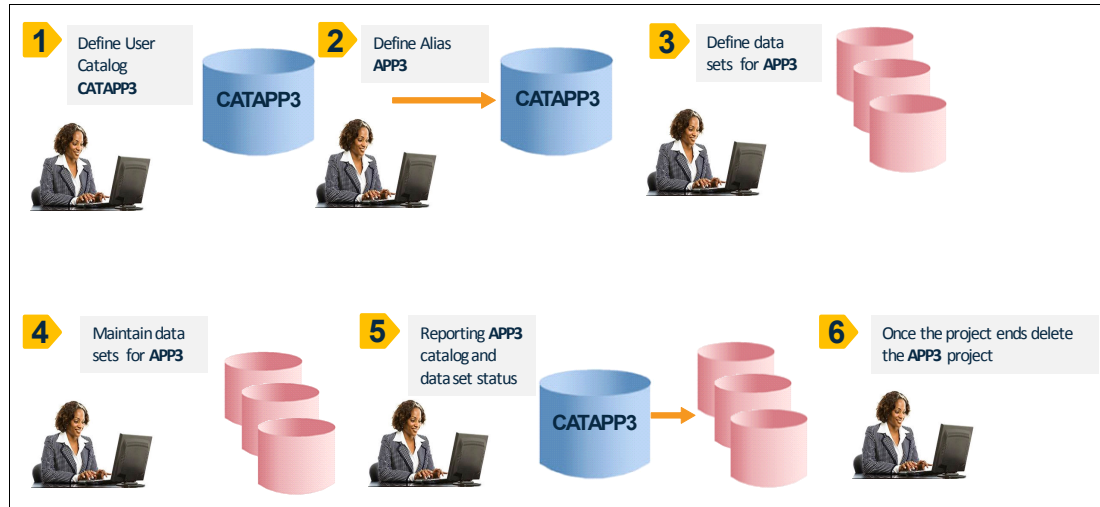


Figure 3-2 Scenario steps overview

As shown in Figure 3-2, the scenario includes the following steps:

1. Define the user catalog. APP3 has its own user catalog that is named CATAPP3. All of the data sets that are used by the application during the development phase are cataloged in CATAPP3.
2. Define the alias APP3. The alias directs all of the data sets with the APP3 high-level qualifier (HLQ) to the user catalog CATAPP3.
3. Create the APP3 project's data sets to hold the programs and the JCL to perform the IDCAMS functions, and test files with a few test records in it.
4. Maintain the data sets. Backups are created, files are reloaded, and the integrity of the data sets is checked.
5. Report the APP3 catalog and data set status. Data sets change as they contain more data; therefore, they must be monitored for inefficiencies and to identify any possible issues in advance so preventive action can be taken.
6. Delete the APP3 project. After the development phase completes, all of the data sets must be backed up and the APP3 environment must be deleted.

### 3.2.1 Creating the APP3 user catalog

There is one master catalog in this scenario. All user catalogs on the z/OS system must be cataloged in the master catalog. The user catalog is created by using the **DEFINE USERCATALOG** function command.

The master catalog in the scenario is named DCAT01 and the APP3 user catalog is named CATAPP3, as shown in Figure 3-3.

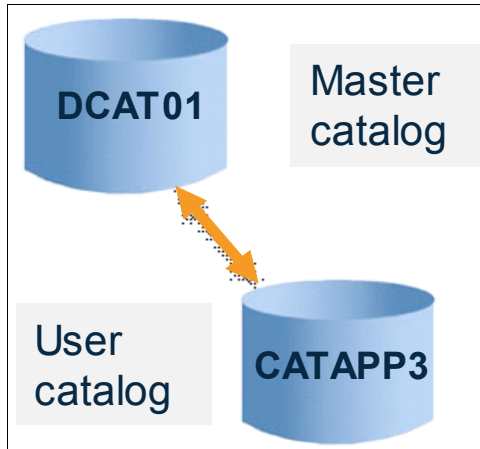


Figure 3-3 Master and user catalogs

By using the **DEFINE USERCATALOG** command, the user catalog CATAPP3 is created. A user catalog connector record in the master catalog also is created.

### 3.2.2 Creating the APP3 alias

The APP3 alias is created by using the **DEFINE ALIAS** function command. An alias entry is added into the master catalog, which directs all of the data sets with the APP3 HLQ to be cataloged in the CATAPP3 user catalog, as shown in Figure 3-4.

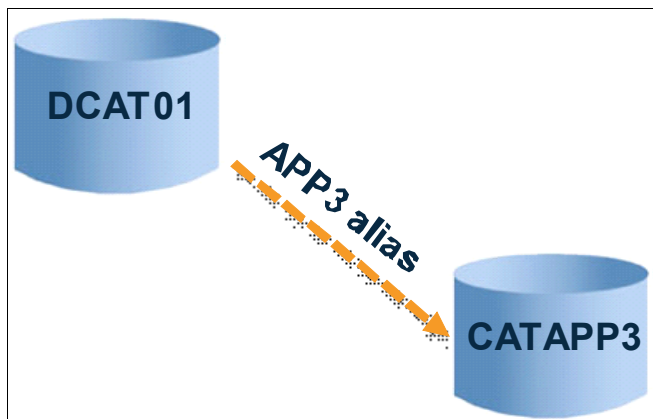


Figure 3-4 APP3 alias

## IDCAMS commands to define CATAPP3 and APP3 alias

To help you learn and combine the use of comments, functional, and modal commands, both tasks are combined into a single stream, as shown in Example 3-1.

*Example 3-1 Defining the user catalog and alias*

---

```
/* DEFINE THE CATAPP3 UCAT */
DEFINE -
    USERCATALOG          -
    (NAME (CATAPP3)      -
    STORCLAS(S1P03S01)  -
    CYLINDERS(1 1))

    /* CONDITIONALLY DEFINE THE APP3 ALIAS */
    IF LASTCC = 0 -
    THEN DO
    DEFINE ALIAS(NAME(APP3) RELATE(CATAPP3))
    END
```

---

**Note:** Some of the comments were changed throughout this chapter to help emphasize the related points.

You can see the results of your command in the SYSPRINT SYSOUT. The output from the user catalog and alias definition is shown in Example 3-2.

*Example 3-2 Defining user catalog and alias output*

---

```
IDCAMS  SYSTEM SERVICES

    DEFINE -
    USERCATALOG          -
    (NAME (CATAPP3)      -
    STORCLAS(S1P03S01)  -
    CYLINDERS(1 1))
IDC0510I CATALOG ALLOCATION STATUS FOR VOLUME BH50E1 IS 0
IDC0512I NAME GENERATED-(I) CATAPP3.CATINDEX
IDC0181I STORAGECLASS USED IS S1P03S01
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

    IF LASTCC = 0 -
    THEN DO

    DEFINE ALIAS(NAME(APP3) RELATE(CATAPP3))
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

    END

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

---

You can see the condition codes that were generated for each functional command. In the remainder of this chapter, only the selected output is shown to avoid repetition. Steps 1 and 2 are now complete.

### 3.2.3 Creating the APP3 data sets

The application data sets are divided into the following categories:

- ▶ Non-VSAM data sets
- ▶ VSAM data sets

The data sets that must be allocated are listed in Table 3-1.

Table 3-1 APP3 data sets

Non-VSAM data sets	VSAM data sets
APP3.PROGRAMS.SOURCE	APP3.PARTS (KSDS)
APP3.PROGRAMS.LOADLIB	APP3.PARTNAME.AIX
APP3.PROJECT.JCL	APP3.PARTNAME.PATH
	APP3.PARTSUPP.AIX
	APP3.PARTSUPP.PATH
	APP3.PARTS.LOG (ESDS)

**Note:** In this scenario, all data sets are created by using JCL to run batch jobs.

The **ALLOCATE** command is used for the non-VSAM data sets and the **DEFINE CLUSTER** command is used for the VSAM data sets.

#### Allocating APP3 non-VSAM data sets

There can be questions as to why the **ALLOCATE** command is used instead of the **DEFINE NONVSAM** command. The **DEFINE NONVSAM** option does not create a data set; instead, it is used to catalog a non-VSAM data set.

**Note:** The **NORECATALOG** (default) option of the **DEFINE NONVSAM** command adds a **NONVSAM** entry to the catalog. It is not mandated that the data set is on the volume.

The **RECATALOG** option adds an SMS-managed **NONVSAM** entry to a catalog by using the non-VSAM volume record (NVR) in the (VSAM Volume data set) VVDS. In this case, the NVR *must* be on the volume.

The APP3.PROGRAMS.SOURCE is to be allocated as a partitioned data set (PDS). The data sets are used by the application developers to hold the source code for their programs.

The APP3.PROGRAMS.LOADLIB is also a PDS and holds the load modules for the APP3 application.

The APP3.PROJECT.JCL is a PDS that contains the JCL and whatever else is required to maintain and report on the APP3 project's data sets and catalog.

## IDCAMS commands to allocate APP3 non-VSAM data sets

The IDCAM SYSIN in-stream commands that are used to allocate the three non-VSAM data sets are shown in Example 3-3.

### Example 3-3 Allocating the non-VSAM data sets

---

```
DELETE APP3.PROGRAMS.SOURCE
DELETE APP3.PROGRAMS.LOADLIB
DELETE APP3.PROJECT.JCL
  SET MAXCC=0
ALLOC DSNAME('APP3.PROGRAMS.SOURCE') NEW RECFM(F B) LRECL(80) -
  BLKSIZE(27920) DSORG(PO) DIR(50) SPACE(10,2) CYLINDERS -
  STORCLAS(S1P03S01)
IF LASTCC = 0 -
  THEN DO
  ALLOC DSNAME('APP3.PROGRAMS.LOADLIB') NEW RECFM(U) LRECL(0) -
  BLKSIZE(32760) DSORG(PO) DIR(50) SPACE(5,1) CYLINDERS -
  STORCLAS(S1P03S01)
  END
IF LASTCC = 0 -
  THEN DO
  ALLOC DSNAME('APP3.PROJECT.JCL') NEW RECFM(F B) LRECL(80) -
  BLKSIZE(27920) DSORG(PO) DIR(50) SPACE(5,1) CYLINDERS -
  STORCLAS(S1P03S01)
  END
```

---

The use of the **DELETE** command ensures that the data set allocations do not fail because the data set is available. A sample from the SYSPRINT SYSOUT output is shown in Example 3-4.

### Example 3-4 Allocating non-VSAM output extract

---

```
IDCAMS  SYSTEM SERVICES

  DELETE APP3.PROGRAMS.LOADLIB
IDC3012I ENTRY APP3.PROGRAMS.LOADLIB NOT FOUND
IDC3009I ** VSAM CATALOG RETURN CODE IS 8 - REASON CODE IS IGGOCLEG-42
IDC0551I ** ENTRY APP3.PROGRAMS.LOADLIB NOT DELETED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8

  SET MAXCC=0

  DELETE APP3.PROJECT.JCL
IDC3012I ENTRY APP3.PROJECT.JCL NOT FOUND
IDC3009I ** VSAM CATALOG RETURN CODE IS 8 - REASON CODE IS IGGOCLEG-42
IDC0551I ** ENTRY APP3.PROJECT.JCL NOT DELETED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8

  SET MAXCC=0

  ALLOC DSNAME('APP3.PROGRAMS.LOADLIB') NEW RECFM(U) LRECL(0) -
  BLKSIZE(32760) DSORG(PO) DIR(50) SPACE(5,1) CYLINDERS -
  STORCLAS(S1P03S01)
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
  END
```

```

IF LASTCC = 0 -
  THEN DO

      ALLOC DSNAME('APP3.PROJECT.JCL') NEW RECFM(F B) LRECL(80) -
      BLKSIZE(27920) DSORG(PO) DIR(50) SPACE(5,1) CYLINDERS -
      STORCLAS(S1P03S01)
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

      END

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0

```

---

IDCAMS issued messages and a condition code of 8. By using the **SET** command, the condition code value is reset so that the processing can continue as specified.

### Defining APP3 VSAM data sets

The **DEFINE CLUSTER** command is used to create a VSAM data set. The following types of VSAM data sets are available:

- ▶ Indexed: Key Sequenced Data Set (KSDS)
- ▶ Non-indexed: Entry Sequenced Data Sets (ESDS)
- ▶ Numbered: Relative Record Data Sets (RRDS)
- ▶ Linear: Linear Data Sets (LDS)

The following VSAM data sets are available to define for the APP3 application:

- ▶ APP3.PARTS: A KSDS that appears in the following parts:
  - Base cluster.
  - The data component that holds the data records. A primary key is defined for the program on which to access the data records. The KEYS(8 0) indicates that the length of the primary key is 8 bytes and is at location 0 of the record (the first byte of the data record). The parameters are separated by a space.
  - The index component that provides the index into the data component. The index reduces the search for a record. Instead of reading the data in a sequential manner, the index contains a hierarchical map of the data component; therefore, it can minimize the search steps for the requested data record.
- ▶ APP3.PARTS.LOG is an ESDS data set. There is no index for this VSAM data set.

### IDCAMS commands to allocate the APP3 Non-VSAM data sets

The IDCAMS commands that are shown in Example 3-5 define both VSAM data sets.

*Example 3-5 Defining the VSAM data sets*

---

```

/* DELETE THE APP3.PARTS VSAM KSDS.          */
      /* NOTE: THIS WILL ALSO DELETE ANY DEFINED AIX OR PATH. */
DELETE APP3.PARTS
      SET MAXCC=0
      /* DELETE THE APP3.PARTS.LOG ESDS.          */
DELETE APP3.PARTS.LOG
      SET MAXCC=0
      /* DEFINE THE APP3.PARTS KSDS.
*/
DEFINE CLUSTER (NAME(APP3.PARTS) STORCLAS(S1P03S01) ) -
      DATA (KEYS(8 0) RECORDS(800 100) RECORDSIZE( 90 100) ) -

```

```

INDEX(RECORDS (500 500))
      /* DEFINE THE ESDS.                                     */
IF    LASTCC = 0 -
      THEN DO
      DEFINE CLUSTER (NAME(APP3.PARTS.LOG) STORCLAS(S1P03S01) -
      RECORDS(100 100) RECORDSIZE(80 80) -
      NONINDEXED)
      END

```

The VSAM data sets are now defined.

**Note:** The **DELETE** command for the APP3.PARTS KSDS deletes the associated data and index components and any associated ALTERNATE INDEXes (IBM AIX®) or PATHs.

You can now use the TSO command to list the name component of the APP3.PARTS KSDS, as shown in Figure 3-5.

```

ISPF Command Shell
Enter TSO or Workstation commands below:
===> LISTCAT ENT(APP3.PARTS) NAME NOPREFIX

```

Figure 3-5 TSO LISTCAT for APP3.PARTS cluster

The output from the command is shown in Figure 3-6.

```

CLUSTER ----- APP3.PARTS
IN-CAT --- CATAPP3
DATA ----- APP3.PARTS.DATA
IN-CAT --- CATAPP3
INDEX ----- APP3.PARTS.INDEX
IN-CAT --- CATAPP3
***

```

Figure 3-6 Output from the LISTCAT for APP3.PARTS

The base cluster is APP3.PARTS with the DATA and INDEX components.

### Loading APP3.PARTS with test data

In this scenario, the application developers created test data for you to load into the APP3.PARTS VSAM data set. They included the test data in a sequential data set that is called APP3.PARTS.INPUT. It contains six data records.

To load the contents of the APP3 PARTS.INPUT into the APPS.PARTS VSAM data set, you can use the IDCAMS **REPRO** command, as shown in Example 3-6 on page 26.

*Example 3-6 Loading the APP3.PARTS test data by using REPRO command*

```
//DATALOAD EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INPUT DD DISP=SHR,DSN=APP3.PARTS.INPUT
//OUTPUT DD DISP=OLD,DSN=APP3.PARTS
//SYSIN DD *
/* LOAD THE KSDS WITH SUPPLIED TEST DATA */
 REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
```

There are two Data Definition (DD) statements in the JCL. The INPUT DD card refers to the sequential data set that contains the test data. The OUTPUT DD refers to the VSAM data set into which you want to load the data. The INFILE on the **REPRO** command specifies the DDNAME of the input data set in the JCL. The OUTFILE specifies the DDNAME of the output data set.

The output from the **REPRO** command is shown in Example 3-7. The messages indicate how many records were loaded into the file.

*Example 3-7 REPRO load of the APP3.PARTS*

```
/* LOAD THE KSDS WITH SUPPLIED TEST DATA */
 REPRO INFILE(INPUT) OUTFILE(OUTPUT)
IDC0005I NUMBER OF RECORDS PROCESSED WAS 6
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 00
IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

## Test data layout

The next step is to define the AIXes and the PATHs that are related to the APPS3.PARTS VSAM data set. First, you must understand the test data record layout to help you to define the AIXes and paths.

The layout of the test data is shown in Figure 3-7.

1	2	3	4	5	6	7	8
01234567890123456789012345678901234567890123456789012345678901234567890.....							
PART	WAREHOUSE	PART	COST PER	QUANTITY	SUPPLIER NAME		
NUMBER	LOCATION	NAME	ITEM	ON HAND			
00000127	ROW 9 CELL 20	WASHER	0.040	120000	WALLY'S WASHER WORLD		
00001392	ROW 6 CELL 14	SCREW	0.003	500000	SAM'S SCREWDRIVERS		
00001395	ROW 5 CELL 11	NAIL	0.001	730000	NELLIE'S NICE NAILS		
00001467	ROW 3 CELL 01	BOLT	0.110	005000	BOB'S BOLT BONANZA		
00002256	ROW 3 CELL 05	NUT1	0.040	002000	NEIL'S NUT HUT		
00002257	ROW 3 CELL 06	NUT2	0.055	003000	NED'S NEIGHBORLY NUTS		

*Figure 3-7 Test data record layout*

The top two layers show the offset into the data record. The record begins at offset 0 and not 1. The next two lines contain the field names.

**Note:** The top four lines are included only for your information and are not part of the test data in the data set.



The test data set contains six records. The offsets and field names are listed in Table 3-2.

Table 3-2 Offsets and field names of test data record

Offset	Field Name
0	PART NUMBER
9	WAREHOUSE LOCATION
24	PART NAME
34	COST PER ITEM
44	QUANTITY ON HAND
55	SUPPLIER NAME

The IDCAMS commands and keywords that are used to define the cluster are shown in Figure 3-5 on page 24. Notice the KEYS(8 0) values. This parameter specifies the primary key. It overrides the values that might be specified on the DATACLASS parameter.

The data records are stored in order of the primary key. Programs that are accessing the data set by using the primary key can locate the data record more quickly. Therefore, the PART NUMBER is the primary key in the APP3.PARTS VSAM KSDS. The index is built based on the primary key.

### Defining an AIX and a Path

In this scenario, suppose that the developers ask you to provide access to the data by using the PART NAME as a key for some programs, and that other programs access the data by using the SUPPLIER NAME as the key. How is this access provided?

The answer is to define an alternative index by using the AIX. This index does not replace the original index; instead, it is another index that is built on a different key. A cluster can include more than one AIX that is related to it, as shown in Figure 3-8.

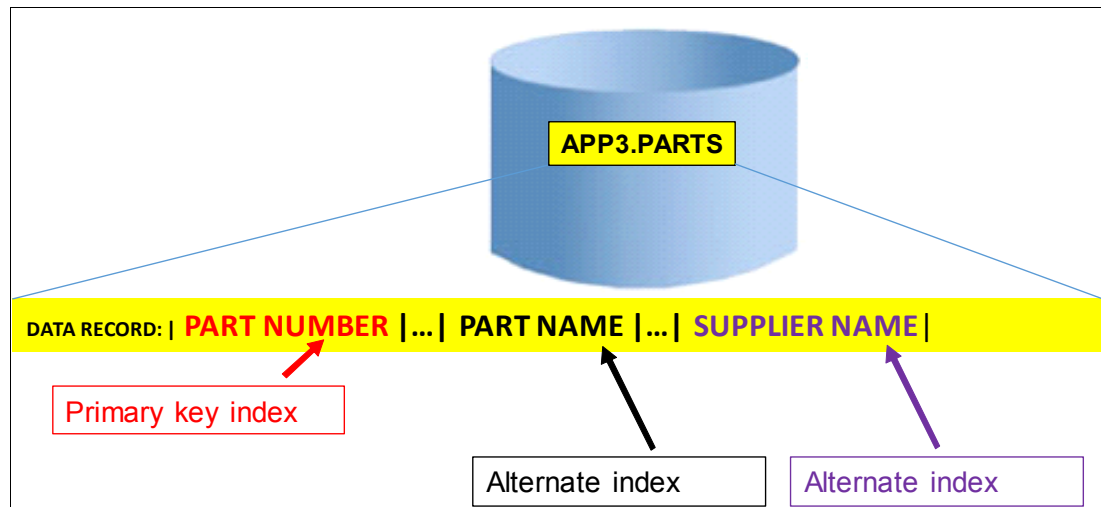


Figure 3-8 Alternative index scenario

When you define an AIX, you relate it to the VSAM cluster for which it is to be used. You also must define a path because the PATH creates an entry name that is related to an AIX name, which indicates that you want to access the cluster via an AIX, as shown in Figure 3-9.

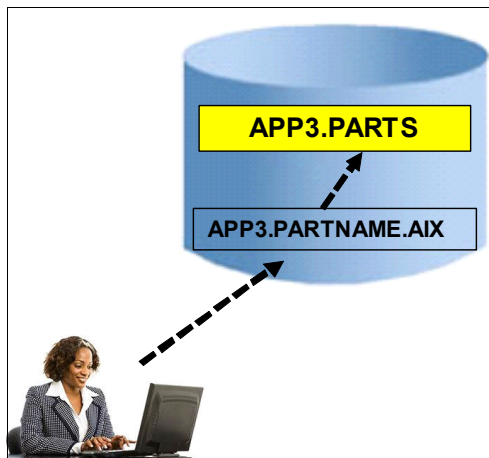


Figure 3-9 Path relating the AIX to the cluster

When the path is opened to process data records, the alternative index and the base cluster are opened and the program accesses the data by the AIX.

### IDCAMS commands to define the AIX and Path

The IDCAMS commands to define the two AIXes and two PATHs are shown in Example 3-8.

#### Example 3-8 AIX and PATH definitions

---

```

DELETE APP3.PARTSUPP.PATH
DELETE APP3.PARTSUPP.AIX
DELETE APP3.PARTNAME.PATH
SET MAXCC=0
DELETE APP3.PARTNAME.AIX
SET MAXCC=0
DEFINE ALTERNATEINDEX (NAME(APP3.PARTNAME.AIX) -
RELATE(APP3.PARTS) KEYS(6 24) RECORDS(800 100) UPGRADE )
IF LASTCC = 0 -
THEN DO
DEFINE PATH (NAME (APP3.PARTNAME.PATH) PATHENTRY(APP3.PARTNAME.AIX) )
END
IF LASTCC = 0 -
THEN DO
DEFINE ALTERNATEINDEX (NAME(APP3.PARTSUPP.AIX) -
RELATE(APP3.PARTS) KEYS(30 55) RECORDS(800 100) NOUPGRADE )
END
IF LASTCC = 0 -
THEN DO
DEFINE PATH (NAME (APP3.PARTSUPP.PATH) PATHENTRY(APP3.PARTSUPP.AIX) )
END

```

---

The **KEYS** parameter values specify its length and offset into the record (see Table 3-2 on page 27). The two fields that you defined as the alternative key for each AIX are the **PART NAME** and the **SUPPLIER NAME**.

The APP3.PARTNAME.AIX specifies the key length as 6 and the offset as 24.

The APP3.PARTSUPP.AIX specifies key length as 30 and the offset as 55.

The application developers in this scenario requested that the APP3.PARTNAME.AIX is upgraded to reflect changed data when the base cluster's records are added to, updated, or erased. This update is done by specifying the UPDGRADE parameter in the **DEFINE** command. The APP3.PARTSUPP.AIX does not need to be upgraded; therefore, the NOUPGRADE parameter was specified.

**Note:** UPGRADE is the default parameter if you do not specify the option that you want to use. Also, you must load the AIX with the necessary data for it to be effective.

## Loading an AIX

You use the **BLDINDEX** command to load an AIX. When you specify the **BLDINDEX** command, IDCAMS opens the base cluster to sequentially read the data records, sorts the information that is obtained from the data records, and builds the alternative index records.

The base cluster's data records are read and information is extracted to form the key-pointer pair. When the base cluster is key sequenced, such as APP3.PARTS, the alternative key value and the data record's prime key value form the key-pointer pair.

The key-pointer pairs are sorted in ascending alternative key order. The sort can be done internally if you specified sufficient virtual storage and access method services perform an internal sort. If you want the sort to be performed externally, you can specify the **EXTERNALSORT** parameter.

If insufficient virtual storage is available to perform the internal sort, IDCAMS defines and uses two sort work files and performs the sort externally. The external sort work file is a VSAM ESDS that is marked as reusable. IDCAMS uses the sort work files to contain most of the key-pointer pairs while it sorts some of them in virtual storage.

Use the following formula to help you calculate the minimum amount of virtual storage you need for an external sort:

$$32768 + ((32768/\text{sort record length}) \times 4)$$

When the key-pointer pairs are sorted into an ascending alternative key order, IDCAMS builds an alternative index record for each key-pointer pair. When all alternative index records are built and loaded into the alternative index, the AIX and its base cluster are closed.

## IDCAMS commands to build the AIX

The IDCAMS commands to build the AIXes are shown in Example 3-9.

*Example 3-9 BLDINDEX for the AIXes*

---

```
/* BUILD THE PART NAME AIX . */
BLDINDEX INDATASET(APP3.PARTS) OUTDATASET(APP3.PARTNAME.AIX)
/* BUILD THE PART SUPPLIER AIX. */
IF LASTCC = 0 -
  THEN DO
    BLDINDEX INDATASET(APP3.PARTS) OUTDATASET(APP3.PARTSUPP.AIX)
  END
```

---

The INDATASET parameters specify the base cluster and the OUTDATASET parameter specifies the AIX to be built.

The setup is now complete.

## 3.3 Maintaining the APP3 project data sets

Several tasks might be needed throughout the project. This section describes those tasks and the IDCAMS statements that are used to perform the tasks.

**Note:** The REPRO Mergecat is outside the scope of this IBM Redpaper publication.

### 3.3.1 Copying data sets

There are different options available to copy the data sets. These options are described next.

#### Copying VSAM to VSAM data sets by using REPRO

If you want to copy your newly defined and loaded APP3.PARTS cluster to another VSAM KSDS as a backup to have a recovery point, you can define a new cluster and copy directly into it. The IDCAMS parameters that are used in this process are shown in Example 3-10.

**Note:** IDCAMS does not use Record Level Sharing (RLS). An RLS keyword is ignored if it is specified on the DD statement that is opened by IDCAMS.

#### Example 3-10 Copying VSAM to VSAM

---

```
/* DELETE THE KSDS.                                     */
/* THIS WILL ALSO DELETE ANY DEFINED AIX OR PATH.      */
DELETE APP3.PARTS.KSDS.COPY
SET MAXCC=0
/* DEFINE THE KSDS.                                     */
DEFINE CLUSTER (NAME(APP3.PARTS.KSDS.COPY) STORCLAS(S1P03S01) ) -
  DATA (KEYS(8 0) RECORDS(800 100) RECORDSIZE( 90 100) ) -
  INDEX(RECORDS (500 500))
END
/* COPY THE KSDS */
IF LASTCC = 0 -
  THEN DO
    REPRO INDATASET(APP3.PARTS) OUTDATASET(APP3.PARTS.KSDS.COPY)
  IF LASTCC = 0 -
    THEN DO
      LISTC ENT(APP3.PARTS) NAME
      LISTC ENT(APP3.PARTS.KSDS.COPY) NAME
    END
```

---

A new cluster APP3.PARTS.KSDS.COPY is defined and the records in the APP3.PARTS are copied by using the **REPRO** command. The output is shown in Example 3-11.

#### Example 3-11 Copying VSAM to VSAM sysout

---

```
/* DELETE THE KSDS.                                     */
/* THIS WILL ALSO DELETE ANY DEFINED AIX OR PATH.      */
DELETE APP3.PARTS.KSDS.COPY
IDC3012I ENTRY APP3.PARTS.KSDS.COPY NOT FOUND
IDC3009I ** VSAM CATALOG RETURN CODE IS 8 - REASON CODE IS IGG0CLEG-42
IDC0551I ** ENTRY APP3.PARTS.KSDS.COPY NOT DELETED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8
```

---

```

SET MAXCC=0

/* DEFINE THE KSDS. */
DEFINE CLUSTER (NAME(APP3.PARTS.KSDS.COPY) STORCLAS(S1P03S01) ) -
  DATA (KEYS(8 0) RECORDS(800 100) RECORDSIZE( 90 100) ) -
  INDEX(RECORDS (500 500))
IDC0508I DATA ALLOCATION STATUS FOR VOLUME BH50E2 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME BH50E2 IS 0
IDC0512I NAME GENERATED-(D) APP3.PARTS.KSDS.COPY.DATA
IDC0512I NAME GENERATED-(I) APP3.PARTS.KSDS.COPY.INDEX
IDC0181I STORAGECLASS USED IS S1P03S01
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

END
/* COPY THE KSDS */
IF LASTCC = 0 -
  THEN DO

  REPRO INDATASET(APP3.PARTS) OUTDATASET(APP3.PARTS.KSDS.COPY)
IDC0005I NUMBER OF RECORDS PROCESSED WAS 6
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IF LASTCC = 0 -
  THEN DO

  LISTC ENT(APP3.PARTS) NAME
CLUSTER ----- APP3.PARTS
  IN-CAT --- CATAPP3
IDCAMS SYSTEM SERVICES
  DATA ----- APP3.PARTS.DATA
  IN-CAT --- CATAPP3
  INDEX ----- APP3.PARTS.INDEX
  IN-CAT --- CATAPP3
IDCAMS SYSTEM SERVICES
  THE NUMBER OF ENTRIES PROCESSED WAS:
    AIX -----0
    ALIAS -----0
    CLUSTER -----1
    DATA -----1
    GDG -----0
    INDEX -----1
    NONVSAM -----0
    PAGESPACE -----0
    PATH -----0
    SPACE -----0
    USERCATALOG -----0
    TAPELIBRARY -----0
    TAPEVOLUME -----0
    TOTAL -----3

  THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  LISTC ENT(APP3.PARTS.KSDS.COPY) NAME
CLUSTER ----- APP3.PARTS.KSDS.COPY

```

```

IN-CAT --- CATAPP3
DATA ----- APP3.PARTS.KSDS.COPY.DATA
IN-CAT --- CATAPP3
INDEX ----- APP3.PARTS.KSDS.COPY.INDEX
IN-CAT --- CATAPP3
IDCAMS  SYSTEM SERVICES
        THE NUMBER OF ENTRIES PROCESSED WAS:
            AIX -----0
            ALIAS -----0
            CLUSTER -----1
            DATA -----1
            GDG -----0
            INDEX -----1
            NONVSAM -----0
            PAGESPACE -----0
            PATH -----0
            SPACE -----0
            USERCATALOG -----0
            TAPELIBRARY -----0
            TAPEVOLUME -----0
            TOTAL -----3
        THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

END

```

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0

---

Consider the following points:

- ▶ The use of the **Delete** command removes any previous version of the copies cluster.
- ▶ The use of the **Define** command defines the new cluster.
- ▶ The use of the **REPRO** command copies the records across. The record count is 6, as was previously loaded.
- ▶ A filtered **LISTCAT** command is run for both data sets to provide a minimum check on both clusters.

### Copying VSAM to NONVSAM data sets by using REPRO

You can copy the APP3.PARTS cluster to a NONVSAM file as a backup, as shown in Example 3-12.

*Example 3-12 Copying a VSAM data set to a non-VSAM backup*

---

```

//BACKUP01 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//BACKUP DD DSN=APP3.PARTS.BACKUP,DISP=(,CATLG,DELETE),
//        DCB=BLKSIZE=32760,
//        SPACE=(CYL,(1,1))
//SYSIN DD *
        REPRO INDATASET(APP3.PARTS) OUTFILE(BACKUP)
/*

```

---

The backup then can be used to reload the cluster.

This technique is frequently used to reorganize VSAM data sets. If the APP3.PARTS cluster was to hold hundreds or thousands of records and feature high activity levels, such as record inserts and deletes, you can expect to see Control Interval (CI) splits and Control Area (CA) splits. However, this split decreases the efficiency of the KSDS data set. You can resolve this issue by using the **REPRO** command to copy the data set to a sequential file and then copy it back, which means that the CIs and CAs are rebuilt and the efficiency restored.

NONVSAM data sets can also be copied to NONVSAM data sets. However, consider the following points:

- ▶ Do not use the **REPRO** command to copy PDS or PDSEs because the information that is stored in the directories is not copied. However, you can copy a PDS member.
- ▶ You do not see the record count when compressed data sets are copied.

### Copying all records in the data set

Although it is advisable to copy all of the records in the data set if you are creating a backup, you might want to copy only some of the records based on your own criteria.

The following parameters can be used to help you select records for a partial copy:

- ▶ For a KSDS:
  - FROMKEY(key): Specifies the key of the first record you want copied. You can specify generic keys (a portion of the key, followed by \*).
  - TOKEY(key): Specifies where copying is to end in the data set that is being copied. You can specify only one of these parameters for a copy operation. The location where copying is to end must follow the location where it is to begin.
- ▶ For a RRDS/VRRDS:
  - FROMNUMBER(number): Specifies the relative record number of the first record you want copied.
  - TONUMBER(number): Specifies the relative record number of the last record you want copied.
- ▶ For KSDS and ESDS or their components:
  - FROMADDRESS(address): Specifies the relative byte address (RBA) of the first record you want copied. The RBA value must be the beginning of a logical record. If you specify this parameter for key-sequenced data, the records are copied in physical sequential order instead of in logical sequential order. This parameter cannot be specified when either of the following conditions exist:
    - When the data set is being accessed through a path.
    - For a key-sequenced data set with spanned records if any of those spanned records are to be accessed.
  - TOADDRESS(address): Specifies the RBA of the last record you want copied. Unlike FROMADDRESS, the RBA value does not need to be the beginning of a logical record. The entire record that contains the specified RBA is copied.

**Note:** The same two restrictions apply for a path and spanned records, as in the FROMADDRESS parameter.

- ▶ SKIP(number): Specifies the number of logical records to skip before copying records.
- ▶ COUNT(number): Specifies the number of logical records you want copied. This setting is not be specified when you access the data set through a path.

**Note:** Sequential Access Method (SAM) data sets can use only SKIP and COUNT.

## Using the EXPORT and IMPORT commands for backups

The use of the **EXPORT** and **IMPORT** commands are an alternative to the use of the **REPRO** command.

The use of the **EXPORT** command exports a VSAM cluster or an AIX, a backup copy of a catalog also can be created.

**EXPORT** copies VSAM information from the catalog and includes it in the portable data set. The **IMPORT** command uses the copied catalog information to re-create the data set before loading it with the exported records.

Use caution when you are selecting your **EXPORT** parameters because these parameters can indicate that you are moving your cluster or AIX elsewhere or possibly archiving it and the original cluster is to be deleted. Pay attention to whether you specify the following parameters:

- ▶ **TEMPORARY:** Specifies that the cluster, alternative index, or catalog is *not* to be deleted from the original system. The object in the original system is marked as **TEMPORARY** to indicate that another copy exists and that the original copy can be replaced by using an **IMPORT** command.
- ▶ **PERMANENT:** Specifies that the cluster or alternative index is to be deleted from the original system. Its storage space is freed. If its retention period did not yet expire, you must also include the **PURGE** parameter.

## IDCAMS commands to create a backup by using EXPORT

The output for the **SYSIN** that is used for exporting the **APP3.PARTS** cluster while keeping the original cluster is shown in Example 3-13.

*Example 3-13 EXPORT APP3.PARTS SYSIN*

---

```
/* BACKUP THE APP3.PARTS CLUSTER AND KEEP THE ORIGINAL CLUSTER */
EXPORT APP3.PARTS -
  OUTFILE (BACKUP) -
  TEMPORARY
```

---

The **SYSOUT** output from the export is shown in Example 3-14.

*Example 3-14 EXPORT APP3.PARTS output*

---

```
IDCAMS  SYSTEM SERVICES                                TIME: 11:36:52

  /* BACKUP THE APP3.PARTS CLUSTER AND KEEP THE ORIGINAL CLUSTER */
  EXPORT APP3.PARTS -
  OUTFILE (BACKUP) -
  TEMPORARY
IDC0005I NUMBER OF RECORDS PROCESSED WAS 6
IDC0594I PORTABLE DATA SET CREATED SUCCESSFULLY ON 08/02/16 AT 11:36:52
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

---



## IDCAMS commands to restore by using IMPORT

The **IMPORT** command to restore APP3.PARTS is shown in Example 3-15.

### Example 3-15 IMPORT APP3.PARTS SYSIN

---

```
/* RESTORE APP3.PARTS FROM EXPORTED BACKUP */
IMPORT -
INFILE(BACKUP) -
OUTDATASET(APP3.PARTS)
```

---

The SYSOUT output from the import is shown in Example 3-16.

### Example 3-16 IMPORT APP3.PARTS output

---

```
IDCAMS  SYSTEM SERVICES                                TIME: 12:05:40

/* RESTORE APP3.PARTS FROM EXPORTED BACKUP */
IMPORT -
INFILE(BACKUP) -
OUTDATASET(APP3.PARTS)
IDC0604I DATA SET BEING IMPORTED WAS EXPORTED ON 08/02/16 AT 11:36:52
IDC0550I ENTRY (R) APP3.PARTNAME.PATH DELETED
IDC0550I ENTRY (D) APP3.PARTNAME.AIX.DATA DELETED
IDC0550I ENTRY (I) APP3.PARTNAME.AIX.INDEX DELETED
IDC0550I ENTRY (G) APP3.PARTNAME.AIX DELETED
IDC0550I ENTRY (R) APP3.PARTSUPP.PATH DELETED
IDC0550I ENTRY (D) APP3.PARTSUPP.AIX.DATA DELETED
IDC0550I ENTRY (I) APP3.PARTSUPP.AIX.INDEX DELETED
IDC0550I ENTRY (G) APP3.PARTSUPP.AIX DELETED
IDC0550I ENTRY (D) APP3.PARTS.DATA DELETED
IDC0550I ENTRY (I) APP3.PARTS.INDEX DELETED
IDC0550I ENTRY (C) APP3.PARTS DELETED
IDC0181I STORAGECLASS USED IS S1P03S01
IDC0508I DATA ALLOCATION STATUS FOR VOLUME BH50E2 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME BH50E2 IS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

---

The use of the **IMPORT** command deleted the base cluster and all its associated components. The APP3.PARTS cluster is re-created along with its DATA and INDEX components by using the information from the export.

**Note:** The AIX and the PATH components were not re-created by using the **IMPORT** command; therefore, you must redefine and reload them.

## 3.4 Obtaining information about the APP3 data sets

This section describes tasks that might be needed throughout the project and the IDCAMS statements that are used to perform those tasks.

### 3.4.1 Using the LISTCAT command

Parameters are available to refine your report to more specific information. If you are running the **LISTCAT** command via JCL and do not enter any parameters, the entire the catalog is listed. However, if you run the command via TSO/E and do not include any parameters, the TSO/E prefix (userid) becomes the HLQ and only those entries that match the userid are listed.

**Note:** Because catalog management does not maintain statistics of its own cluster entry, the information you see might not be accurate.

The abbreviation for the **LISTCAT** command is **LISTC**.

You can include the following parameters in the command:

- ▶ **ENTRIES(entrnames):** This parameter limits the information to the entry name that is specified, as shown in Example 3-17.

*Example 3-17 LISTC ENTRY output*

---

```
LISTC ENT(APP3.PROJECT.JCL) HISTORY NOPREFIX
NONVSAM ----- APP3.PROJECT.JCL
      IN-CAT --- CATAPP3
      HISTORY
      DATASET-OWNER-----(NULL)      CREATION-----2016.215
      RELEASE-----2          EXPIRATION-----0000.000
      ACCOUNT-INFO------(NULL)
      SMSDATA
      STORAGECLASS ---S1P03S01      MANAGEMENTCLASS---(NULL)
      DATACLASS  -----(NULL)      LBACKUP ---0000.000.0000
IDCAMS  SYSTEM SERVICES
      THE NUMBER OF ENTRIES PROCESSED WAS:
      AIX -----0
      ALIAS -----0
      CLUSTER -----0
      DATA -----0
      GDG -----0
      INDEX -----0
      NONVSAM -----1
      PAGESPACE -----0
      PATH -----0
      SPACE -----0
      USERCATALOG -----0
      TAPELIBRARY -----0
      TAPEVOLUME -----0
      TOTAL -----1
      THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

---

- ▶ **LEVEL(h1q):** Limits to one or more qualifiers.

- ▶ CATALOG(*catalog name*): Limits to the specified catalog.
- ▶ LIBRARY(*library name*): Limits to the name of the tape library for which tape volume entries are to be listed.
- ▶ ALIAS: Only the alias entries are listed.
- ▶ USERCATALOGS: Lists the catalog connectors.

### Limiting the amount of information

Because the output from the use of the **LISTCAT** command can be verbose, parameters can be added to make the information more specific.

The following parameter options are available:

- ▶ By information that is wanted:
  - NAME: Lists entry names and names of associations, such as data, index, or alias.
  - HISTORY: Lists entry type, owner ID, creation date, expiration date, and release.
  - VOLUME HISTORY: Volume serial numbers and device types that are allocated to the entries.
  - ALLOCATION VOLUME: Detailed information about the allocation.
  - ALL: Specifies that all fields are to be listed, including fields that are not listed under the NAME, HISTORY, VOLUME, and ALLOCATION options, such as CA-RECLAIM and fields under ATTRIBUTES, and STATISTICS.
- ▶ By Entry Type: ALIAS, ALTERNATEINDEX, CLUSTER, DATA, GENERATIONDATAGROUP, INDEX LIBRARYENTRIES, NONVSAM, PAGESPACE, PATH, USERCATALOG, and VOLUMEENTRIES
- ▶ By Date:
  - CREATION(*days*): Listed only if it was created a specified number of days ago or earlier (*days+n*).
  - EXPIRATION(*days*): Listed only if the entry expires in the number of days or earlier (*days+n*).
- ▶ By Catalog: CATALOG(*catalog name*).

## 3.4.2 Using the PRINT commands

You can use the **PRINT** command to print all or some records in a data set. In terms of selecting ranges, it is similar to the **REPRO** command. You can print the records in Character, Dump, or HEX format. The **PRINT** command for all of the records in the APP3.PARTS data set in character format is shown in Example 3-18.

*Example 3-18 PRINT command*

---

```

IDCAMS  SYSTEM SERVICES                                TIME: 22:30:30

      PRINT INDATASET(APP3.PARTS) CHARACTER                00050000
IDCAMS  SYSTEM SERVICES                                TIME: 22:30:30
LISTING OF DATA SET -APP3.PARTS
KEY OF RECORD - 00000127
00000127 ROW 9 CELL 20 WASHER    0.040    120000    WALLY'S WASHER WORLD
KEY OF RECORD - 00001392
00001392 ROW 6 CELL 14 SCREW    0.003    500000    SAM'S SCREWDRIVERS
KEY OF RECORD - 00001395
00001395 ROW 5 CELL 11 NAIL     0.001    730000    NELLIE'S NICE NAILS
KEY OF RECORD - 00001467

```

```

00001467 ROW 3 CELL 01 BOLT      0.110    005000    BOB'S BOLT BONANZA
KEY OF RECORD - 00002256
00002256 ROW 3 CELL 05 NUT1     0.040    002000    NEIL'S NUT HUT
KEY OF RECORD - 00002257
00002257 ROW 3 CELL 06 NUT2     0.055    003000    NED'S NEIGHBORLY NUTS
IDC0005I NUMBER OF RECORDS PROCESSED WAS 6
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDCAMS  SYSTEM SERVICES                                TIME: 22:30:30

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0

```

---

### 3.4.3 Checking the KSDS structural integrity

Use the **EXAMINE** command to analyze and report on the structural integrity of the index and data components of a (KSDS) and of a variable-length relative record data set cluster (VRRDS). The **EXAMINE** command also can be used to analyze and report on the structural integrity of the basic catalog structure (BCS) of a catalog.

The results of the use of an **EXAMINE** command on APP3.PARTS with the index and data components being tested is shown in Example 3-19.

*Example 3-19 Examining the APP3.PARTS KSDS*

---

```

EXAMINE NAME(APP3.PARTS) -
      INDEXTEST -
      DATATEST
IDC01700I INDEXTEST BEGINS
IDC11773I          1 KEYS PROCESSED ON INDEX LEVEL  1, AVERAGE KEY LENGTH:
0.0
IDC11774I CURRENT INDEX CISIZE IS  512, RECOMMENDED MINIMUM INDEX CISIZE IS  512
IDC01724I INDEXTEST COMPLETE - NO ERRORS DETECTED
IDC01701I DATATEST BEGINS
IDC01709I DATATEST COMPLETE - NO ERRORS DETECTED
IDC01708I 1 CONTROL INTERVALS ENCOUNTERED
IDC01710I DATA COMPONENT CONTAINS 6 RECORDS
IDC01711I DATA COMPONENT CONTAINS 0 DELETED CONTROL INTERVALS
IDC01712I MAXIMUM LENGTH DATA RECORD CONTAINS 100 BYTES
IDC01722I 66 PERCENT FREE SPACE
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0

```

---

## 3.5 Removing the APP3 project

After the project is completed, the application developers request that you to remove the project data sets, including the alias and user catalog.

### 3.5.1 Delete command

You can delete individual elements, such as data sets, or you can use a mask facility to delete a group of elements that match the mask that you provide.

The following options and parameters are available to delete the various elements:

- ▶ SCRATCH or NOSCRATCH:
  - SCRATCH removes the VVR or NVR and associated VTOC entries.
  - NOSCRATCH performs an uncatalog action only.
- ▶ ERASE or NOERASE:
  - ERASE is valid with the SCRATCH option only. Consider the following points:
    - The AIX components are to be overwritten with binary zeros when deleted.
    - More processing is required for the overwriting.
    - The option also can be set in an IBM RACF® profile.
  - NOERASE does not perform the overwrite. The RACF options can override the NOERASE option so that an ERASE is performed.

#### Deleting the APP3 project

There are different ways of approaching this process, but it is suggested that you delete the project by using the following process:

1. Delete all the APP3 project data sets.
2. Delete the alias.
3. Ensure that the user catalog is empty.
4. Delete the user catalog.

#### IDCAMS commands to restore by using IMPORT

The **DELETE** commands that are used to remove the APP3 application are shown in Example 3-20.

*Example 3-20 Removing the APP3 application*

---

```
/*                                                    */
/* DELETE THE APP3 DATA SETS          */
/*                                                    */
DELETE APP3.PARTS CLUSTER
DELETE APP3.PARTS.LOG CLUSTER
DELETE APP3.PROGRAMS.SOURCE NONVSAM
DELETE APP3.PROGRAMS.LOADLIB NONVSAM
DELETE APP3.PROJECT.JCL NONVSAM
DELETE APP3 ALIAS
DELETE CATAPP3 USERCATALOG
```

---

Your first mission was now completed successfully.

## 3.6 Useful things to know

Anomalies can happen. Although everything was done correctly, an anomaly still exists. This section helps you with those issues. If you experience any other types of anomalies, send the feedback to us and we will include them in this paper. For more information about sending feedback, see “Comments welcome” on page xi.

### 3.6.1 Missing comments in your SYSPRINT

In your SYSIN comments, you might include comments in lowercase, as shown in Example 3-21.

*Example 3-21 SYSIN including lowercase*

---

```
//LCTEST01 JOB ,MSGCLASS=H
//STEP001 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
/* this is a lowercase test */
LISTC ENT(APP3) HISTORY NOPREFIX
/*
//
```

---

After the job was run, an extract of the SYSPRINT is shown in Example 3-22.

*Example 3-22 Lowercase not shown in the comments*

---

```
IDCAMS SYSTEM SERVICES

/* ..... */
LISTC ENT(APP3) HISTORY NOPREFIX
ALIAS ----- APP3
IN-CAT --- MCAT.BH5CAT
HISTORY
RELEASE-----2 CREATION-----2016.217
```

---

This issue is corrected by changing the print train default parameter value, which is set to PN. The value can be changed to TN, which is the character set for text printing 120 characters and displaying lowercase comments. You specify this parameter by using the **PARM** command.

#### **PARM** command

The **PARM** command specifies processing options to be used during execution. These options remain in effect until they are changed by another **PARM** command. You can also use these options in the **PARM** field of an **EXEC** statement, which is in the JCL. The syntax of the **PARM** command is shown in Example 3-23 on page 41.

*Example 3-23 PARM command*

---

Command Parameters

```
PARM          [TEST({[TRACE]
                [AREAS(areaid[ areaid...])]
                [FULL((dumpid[ begin[ count]])
                [(dumpid...)...])]|
                OFF})]
                [GRAPHICS(CHAIN(chain)|TABLE(mname))]
                [MARGINS(leftmargin rightmargin)]
```

---

A **PARM** is added to the **EXEC** statement, as shown in Example 3-24.

*Example 3-24 Adding the PARM by using the EXEC statement*

---

```
//LCTEST02 JOB ,MSGCLASS=H
//STEP001 EXEC PGM=IDCAMS,PARM='GRAPHICS(CHAIN(TN))'
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
/* this is a lowercase test */
LISTC ENT(APP3) HISTORY NOPREFIX
/*
//
```

---

The job is run and the relevant SYSPRINT is shown in Example 3-25.

*Example 3-25 Lowercase text that is displayed by using PARM on the EXEC*

---

```
IDCAMS SYSTEM SERVICES
GRAPHICS(CHAIN(TN))
IDCAMS SYSTEM SERVICES
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

/* this is a lowercase test */
LISTC ENT(APP3) HISTORY NOPREFIX
ALIAS ----- APP3
IN-CAT --- MCAT.BH5CAT
HISTORY
RELEASE-----2          CREATION-----2016.217
```

---

The **PARM** command can be entered in the **SYSIN**, as shown in Example 3-26.

*Example 3-26 SYSIN instream PARM command*

---

```
//LCTEST03 JOB ,MSGCLASS=H
//STEP001 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
PARM GRAPHICS(CHAIN(TN))
/* this is a lowercase test */
LISTC ENT(APP3) HISTORY NOPREFIX
/*
//
```

---

The resulting SYSPRINT is the same as shown in Example 3-25.

The job fails if you enter the IDCAMS function commands in lowercase, as shown in Example 3-27.

*Example 3-27 Commands in lowercase fail*

---

```
PARM GRAPHICS(CHAIN(TN))
IDCAMS SYSTEM SERVICES
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

/* this is a lowercase test */
listc ent(app3) history noprefix
IDC3219I VERB NAME 'listc' UNKNOWN
IDC3202I ABOVE TEXT BYPASSED UNTIL NEXT COMMAND. CONDITION CODE IS 12
```

---

### 3.6.2 DYNAMBR parameter

In our APP3 project scenario, we used the **ALLOCATE** and **DEFINE** commands to create and allocate data sets dynamically, which means there were no DD statements in the JCL. All of the data sets were allocated by using **SYSIN** in-stream commands. If you are provisioning a many data sets, you might need to use the **DYNAMBR** parameter.

Dynamic allocation allows a job to acquire resources as the resources are needed and release them immediately after use. The resources are a ddname-data set combination with its volumes and devices.

One reason to use dynamic allocation is that you might not know all of the device requirements for a job before the job is run. Another reason is that it allows the system to use resources more efficiently; that is, the system can acquire resources before they are used and then release those resources immediately after use.

To dictate the number of resources to be held in anticipation of reuse, use the following code:

```
//stepname EXEC PGM=IDCAMS,DYNAMNBR=n
```

Specify *n* as a decimal number 0 - 3273, minus the number of DD statements in the step.

**Note:** The limit of 3273 is based on the number of single unit DD statements for a 64 K task input/output table (TIOT). This limit can be different depending on the installation-defined TIOT size. The 32 K size is the default TIOT size. The limit for a 32 K TIOT is 1635. (In a JES3 system, the installation might further reduce the limit.) The default value is 0.

The system uses the sum of this number and the number of DD statements (including any DD statements that are in a cataloged or in-stream procedure that is called by the step) in the step to establish a control limit for tracking resources that it is holding in anticipation of reuse.

**Note:** If this control limit is reached and another dynamic allocation is requested, the request is not honored unless resources can be unallocated so that the control limit is not exceeded.



### 3.6.3 UNIQUE KEY

When a key is selected for an AIX, the field to which that key refers in the base cluster might not be unique. A sample data set record layout is listed in Table 3-3.

Table 3-3 Base and AIX keys

Applicant ID	Applicant first name	Applicant city
10001	Mary	New York
10002	John	Denver
10003	Sally	New York
10004	David	Chicago

The base cluster is the Applicant ID. Each applicant for this application is given a unique identifier. The second field in each record contains each applicant's first name, and the third field lists the applicant's city.

If an AIX is set up with the APPLICANT CITY as the key, a duplicate key exists (see Table 3-3); therefore, the key is not unique.

When the AIX is defined, the UNIQUEKEY | NONUNIQUEKEY parameter option is available.

This parameter shows whether more than one data record (in the base cluster) can contain the same key value for the alternative index.

UNIQUEKEY points each alternative index key to only one data record. When the alternative index is built and more than one data record contains the same key value for the alternative index, the BLDINDEX processing ends with an error message.

NONUNIQUEKEY points a key value for the alternative index to more than one data record in the base cluster. The alternative index's key record points to a maximum of 32768 records with non-unique keys. When NONUNIQUEKEY is included, the maximum record size is large enough to allow for alternative index records that point to more than one data record. This setting is the default setting.

During BLDINDEX processing when the key-pointer pairs are sorted into ascending alternative key order, IDCAMS builds an AIX record for each key-pointer pair. If the NONUNIQUEKEY attribute is used and more than one key-pointer pair features the same alternative key values, the alternative index record contains the alternative key value followed by the pointer values in ascending order. If the UNIQUEKEY attribute is used, each alternative key value must be unique.

## 3.7 Provisioning jobs

The examples that are used to set up the APP3 project are set up individually and run manually. That is, each job runs after you submit it. The jobs perform individual roles. In this section, we introduce you to the following methods you might use to set up the jobs:

- ▶ JES2 JOBGROUPS
- ▶ A procedure that features instream SYSIN and variables

### 3.7.1 JES2 JOBGROUPs

The JOBGROUP option that was introduced in z/OS V2.2 allows you to set up a schedule for the jobs to run and include dependencies on each other. Because the use of JOBGROUPs requires new section and data area within checkpoint data sets, it is possible to use this function only when running on checkpoint level z22.

JOBGROUPs allow you to define a schedule that ensures the jobs are run in the correct sequences and any jobs that you identify as running concurrently with others run as such. Figure 3-10 shows a schematic to define the APP7 project much as we defined the APP3 project. The JOBGROUP is named APP7JOBG.

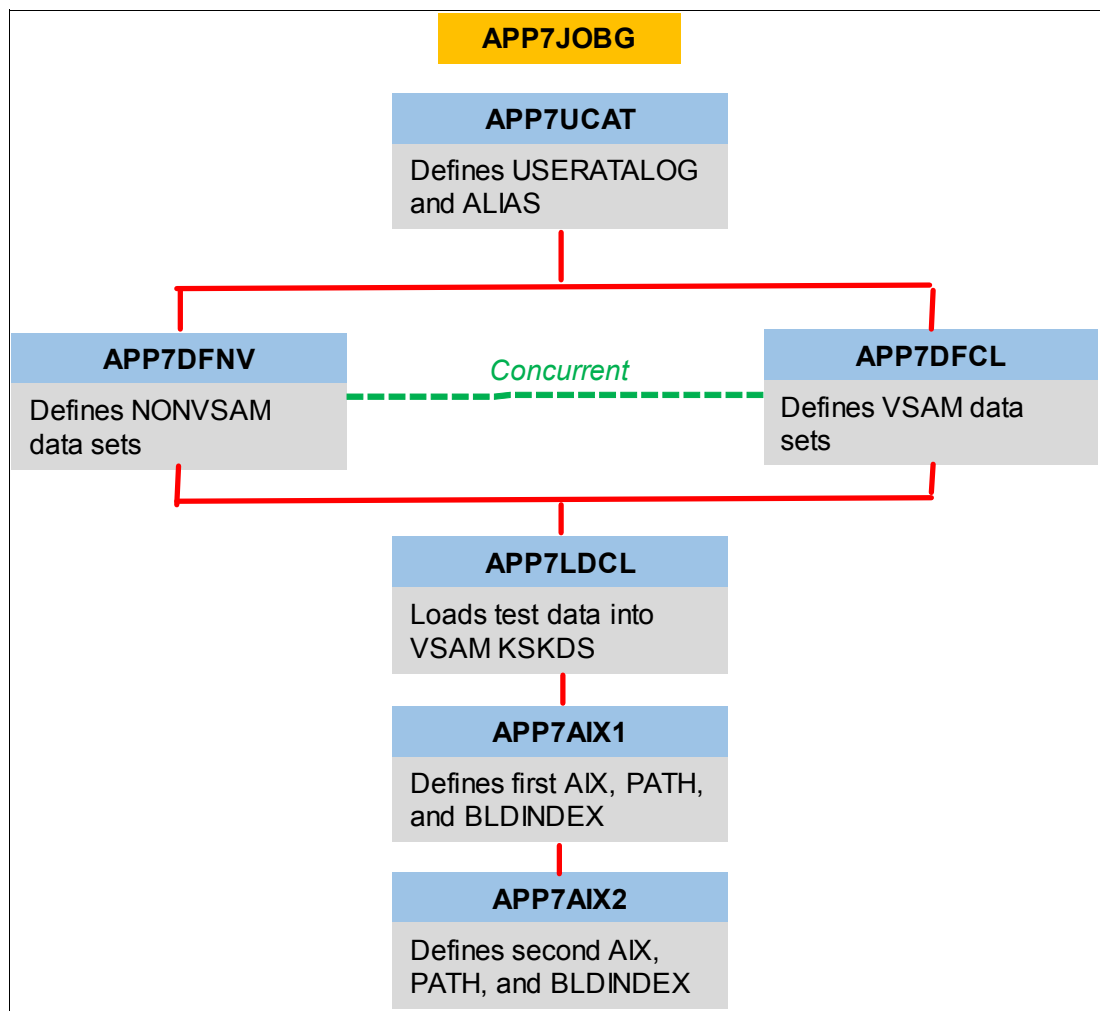


Figure 3-10 APP7JOBG JOBGROUP

Figure 3-10 on page 44 also shows the structure of the APP7JOBG. The jobs roles are included below the job names of the jobs that make up the JOBGROUP. The relationship between the jobs is shown in the following examples:

- ▶ APP7UCAT is the first job to run.
- ▶ APP7DFNV and APP7DEFC run if job APP7UCAT completes with a return code 0. Both jobs can run concurrently.
- ▶ APP7LDCL runs if APP7DFNV and APP7DEFC complete with a return code of 0. Although this job loads only the VSAM KSDS with test data, steps can be added to also load the NONVSAM data sets with PDS members, such as program sources and the JCL to compile and link-edit them. If you choose to do use this configuration, you can encounter concurrent load jobs for each data set.
- ▶ APP7AIX1 runs if job APP7LDCL completes with a return code 0.
- ▶ APP7AIX2 runs if job APP7AIX1 completes with a return code 0. The AIX jobs cannot run together because they serialize on the APP7.PARTS KSDS.

## Defining the JOBGROUP

You must define the JOBGROUP and submit it before submitting the individual jobs. The JOBGROUP must be created before the jobs are submitted for the JOBGROUP or jobs relationship to be recognized by JES. JOBGROUPs are submitted by using an internal reader. The JCL and JOBGROUP that was set up for project APP7 is shown in Example 3-28. (The actual JOBGROUP is defined in the SYSUT1 instream data.)

### Example 3-28 APP7 project JOBGROUP

---

```
//APP7JOBG JOB ,MSGCLASS=H,NOTIFY=KWRES04
//SUBMITJG EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DATA,DLM=@@
//APP7JOBG JOBGROUP
//APP7UCAL GJOB
//APP7DFNV GJOB
// AFTER NAME=APP7UCAL,
// WHEN=(RC=0)
// CONCURRENT NAME=APP7DFCL
//APP7DFCL GJOB
// AFTER NAME=APP7UCAL,
// WHEN=(RC=0)
//APP7LDCL GJOB
// AFTER NAME=(APP7DFNV,APP7DFCL),
// WHEN=(RC=0)
//APP7AIX1 GJOB
// AFTER NAME=APP7LDCL,
// WHEN=(RC=0)
//APP7AIX2 GJOB
// AFTER NAME=APP7AIX1,
// WHEN=(RC=0)
//APP7JOBG ENDGROUP
@@
//SYSUT2 DD SYSOUT=(A,INTRDR)
//SYSIN DD DUMMY,DCB=BLKSIZE=80
//
```

---

For more information about the use of JOBGROUPs, see *z/OS V2R2: JES2, JES3, and SDSF*, SG24-8287, which is available for download at this website:

<http://www.redbooks.ibm.com/redbooks/pdfs/sg248287.pdf>

### **Concurrent JOBS**

When concurrent jobs are used, ensure that you set a value on the GRPDEF definition. Use the **\$DGRPDEF** command and check the CONCURRENT\_MAX value (default is 0), as shown in Example 3-29.

*Example 3-29 Checking the CONCURRENT\_MAX value*

---

```
$DGRPDEF
$HASP732 GRPDEF 010
$HASP732 GRPDEF ZJCNUM=1000,ZJCFREE=975,ZJCWARN=80,
$HASP732          JOBGROUP_JOB_MAX=2000,CONCURRENT_MAX=20
```

---

The CONCURRENT\_MAX value specifies how many concurrences you can have in a JOBGROUP. To modify the limit, use the **\$TGRPDEF, CONCURRENT\_MAX=20** command.

### **Declaring job relationships to JOBGROUP**

Each job that you include in the JOBGROUP must declare its relationship to the JOBGROUP. The JCL SCHEDULE parameter declares the relationship, as shown in Example 3-30.

*Example 3-30 JOB to JOBGROUP affinity*

---

```
//APP7UCAL JOB ,MSGCLASS=H,NOTIFY=KWRES04
//          SCHEDULE JOBGROUP=APP7JOBG
```

---

Each job in the JOBGROUP must declare this relationship.

### **SDSF JG option**

Example 3-31 shows the JOBGROUP output via the SDSF JG option.

*Example 3-31 JOBGROUP output*

---

```
USERID KWRES04 IS ASSIGNED TO THIS JOB.
$HASP1300 APP7UCAL registered to job group APP7JOBG
$HASP1301 APP7UCAL in job group APP7JOBG queued for execution
$HASP1300 APP7DFNV registered to job group APP7JOBG
$HASP373 APP7UCAL STARTED - INIT 1 - CLASS A - SYS SC74
$HASP1300 APP7DFCL registered to job group APP7JOBG
$HASP1300 APP7LDCL registered to job group APP7JOBG
$HASP1300 APP7AIX1 registered to job group APP7JOBG
$HASP395 APP7UCAL ENDED - RC=0000
$HASP1301 Concurrent set containing job APP7DFCL in job group APP7JOBG queued for
execution
$HASP1300 APP7AIX2 registered to job group APP7JOBG
$HASP1201 Concurrent set containing job APP7DFCL in job group
APP7JOBG is entering execution
$HASP373 APP7DFCL STARTED - WLM INIT - SRVCLASS DFLT_MG - SYS SC74
$HASP373 APP7DFNV STARTED - WLM INIT - SRVCLASS DFLT_MG - SYS SC74
$HASP395 APP7DFCL ENDED - RC=0000
$HASP395 APP7DFNV ENDED - RC=0000
$HASP1301 APP7LDCL in job group APP7JOBG queued for execution
$HASP373 APP7LDCL STARTED - INIT 1 - CLASS A - SYS SC74
$HASP395 APP7LDCL ENDED - RC=0000
```

```

$HASP1301 APP7AIX1 in job group APP7JOBG queued for execution
$HASP373 APP7AIX1 STARTED - INIT 2 - CLASS A - SYS SC74
$HASP395 APP7AIX1 ENDED - RC=0000
$HASP1301 APP7AIX2 in job group APP7JOBG queued for execution
$HASP373 APP7AIX2 STARTED - INIT 1 - CLASS A - SYS SC74
$HASP395 APP7AIX2 ENDED - RC=0000
$HASP1304 job group APP7JOBG is complete

```

---

Example 3-32, Example 3-33 on page 48, Example 3-34 on page 48, Example 3-35 on page 49, Example 3-36 on page 50, and Example 3-37 on page 50 show the individual jobs that made up the JOBGROUP.

*Example 3-32 APP7UCAL job*

---

```

//APP7UCAL JOB ,MSGCLASS=H,NOTIFY=KWRES04
//          SCHEDULE JOBGROUP=APP7JOBG
//*
/* +++ THIS JOB IS PART OF THE APP7JOBG JOBGROUP +++
/* =====
/*
//DEFUCAL EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
/* DEFINE THE CATALOG AND ALIAS FOR THE APP7 PROJECT */
/* NOTE: */
/* CHECK IF THE CATALOG ALREADY EXISTS, IF SO END THE RUN AND */
/* INVESTIGATE IF OK TO DELETE THE EXISTING CATALOG */
/* */
LISTC ENT(CATAPP7) NAME
IF LASTCC > 0 -
  THEN DO
    SET MAXCC=0
  DEFINE -
    USERCATALOG -
      (NAME (CATAPP7) -
        STORCLAS(S1P03S01) -
        CYLINDERS(1 1))
END

IF LASTCC = 0 -
  THEN DO
    DEFINE ALIAS(NAME(APP7) RELATE(CATAPP7))
  END
/*
//

```

---

*Example 3-33 APP7DFNV job*

---

```
//APP7DFNV JOB ,MSGCLASS=H,NOTIFY=KWRES04
//          SCHEDULE JOBGROUP=APP7JOBG
//*
/* +++ THIS JOB IS PART OF THE APP7JOBG JOBGROUP +++
/* =====
/*
//DEFNV    EXEC PGM=IDCAMS,DYNAMNBR=10
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
DELETE APP7.PROGRAMS.SOURCE NONVSAM
DELETE APP7.PROGRAMS.LOADLIB NONVSAM
DELETE APP7.PROJECT.JCL      NONVSAM
SET MAXCC=0

ALLOC DSNAME('APP7.PROGRAMS.SOURCE') NEW RECFM(F B) LRECL(80) -
BLKSIZE(27920) DSORG(PO) DIR(50) SPACE(10,2) CYLINDERS -
STORCLAS(S1P03S01)

IF LASTCC = 0 -
  THEN DO
    ALLOC DSNAME('APP7.PROGRAMS.LOADLIB') NEW RECFM(U) LRECL(0) -
    BLKSIZE(32760) DSORG(PO) DIR(50) SPACE(5,1) CYLINDERS -
    STORCLAS(S1P03S01)
  END

IF LASTCC = 0 -
  THEN DO
    ALLOC DSNAME('APP7.PROJECT.JCL') NEW RECFM(F B) LRECL(80) -
    BLKSIZE(27920) DSORG(PO) DIR(50) SPACE(5,1) CYLINDERS -
    STORCLAS(S1P03S01)
  END
/*
//
```

---

*Example 3-34 APP7DFCL job*

---

```
//APP7DFCL JOB ,MSGCLASS=H,NOTIFY=KWRES04
//          SCHEDULE JOBGROUP=APP7JOBG
//*
/* +++ THIS JOB IS PART OF THE APP7JOBG JOBGROUP +++
/* =====
/*
//DEFCL    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
/* APPX IS COMMON HLQ FOR ALL APPN PROJECTS
//TESTDATA DD DISP=OLD,DSN=APPX.PARTS.TESTDATA
//SYSIN DD  *
/*
/* DEFINE THE VSAM DATA SETS
/* -----
/*
/* DELETE THE KSDS.
/*
```

```

/* NOTE: THIS WILL ALSO DELETE ANY DEFINED AIX OR PATH. */
DELETE APP7.PARTS
SET MAXCC=0

/* DEFINE THE KSDS. */
DEFINE CLUSTER (NAME(APP7.PARTS) STORCLAS(S1P03S01) ) -
  DATA (KEYS(8 0) RECORDS(800 100) RECORDSIZE( 90 100) ) -
  INDEX(RECORDS (300 300))

IF LASTCC = 0 -
  THEN DO

/* DELETE THE ESDS. */
DELETE APP7.PARTS.LOG
SET MAXCC=0

/* DEFINE THE ESDS. */
DEFINE CLUSTER (NAME(APP7.PARTS.LOG) STORCLAS(S1P03S01) -
  RECORDS(100 100) RECORDSIZE(80 80) -
  NONINDEXED)
END
/*
//

```

---

*Example 3-35 APP7LDCL job*

```

//APP7LDCL JOB ,MSGCLASS=H,NOTIFY=KWRES04
//          SCHEDULE JOBGROUP=APP7JOBG
//*
/* +++ THIS JOB IS PART OF THE APP7JOBG JOBGROUP +++
/* =====
/*
//LOADDATA EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
/*      APPX IS COMMON HLQ FOR ALL APPN PROJECTS
//TESTDATA DD DISP=OLD,DSN=APPX.PARTS.TESTDATA
//SYSIN DD *
          REPRO INFILE(TESTDATA) OUTDATASET(APP7.PARTS)
          PRINT INDATASET(APP7.PARTS)
/*
//

```

---

*Example 3-36 APP7AIX1 job*

---

```
//APP7AIX1 JOB ,MSGCLASS=H,NOTIFY=KWRES04
//          SCHEDULE JOBGROUP=APP7JOBG
//*
//* +++ THIS JOB IS PART OF THE APP7JOBG JOBGROUP +++
//* =====
//*
//DEFAIX1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
/* DEFINE THE PART NAME AIX. */
  DEFINE ALTERNATEINDEX (NAME(APP7.PARTNAME.AIX) -
    RELATE(APP7.PARTS) KEYS(6 24) RECORDS(800 100) UPGRADE )
  END

  IF LASTCC = 0 -
    THEN DO
/* DEFINE THE PART NAME PATH. */
  DEFINE PATH (NAME (APP7.PARTNAME.PATH) -
    PATHENTRY(APP7.PARTNAME.AIX) )
  END

  IF LASTCC = 0 -
    THEN DO
/* BUILD THE PART NAME AIX. */
BLDINDEX INDATASET(APP7.PARTS) -
  OUTDATASET(APP7.PARTNAME.AIX)
  END
  PRINT INDATASET(APP7.PARTNAME.AIX)
/*
//
```

---

*Example 3-37 APP7AIX2 job*

---

```
//APP7AIX2 JOB ,MSGCLASS=H,NOTIFY=KWRES04
//          SCHEDULE JOBGROUP=APP7JOBG
//*
//* +++ THIS JOB IS PART OF THE APP7JOBG JOBGROUP +++
//* =====
//*
//DEFAIX2 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
/* DEFINE THE PART SUPPLIER AIX. */
  DEFINE ALTERNATEINDEX (NAME(APP7.PARTSUPP.AIX) -
    RELATE(APP7.PARTS) KEYS(30 55) RECORDS(800 100) NOUPGRADE)
  END

  IF LASTCC = 0 -
    THEN DO
/* DEFINE THE PART SUPPLIER PATH. */
  DEFINE PATH (NAME (APP7.PARTSUPP.PATH) -
    PATHENTRY(APP7.PARTSUPP.AIX) )
```



```

        END

        IF LASTCC = 0 -
          THEN DO
            /* BUILD THE PART SUPPLIER AIX. */
            /* BUILD THE PART SUPPLIER AIX. */
            BLDINDEX INDATASET(APP7.PARTS) -
            OUTDATASET(APP7.PARTSUPP.AIX)
            END
            PRINT INDATASET(APP7.PARTSUPP.AIX)
          /*
          //

```

---

### 3.7.2 Using a procedure with instream SYSIN and variables

In z/OS V2.1 it became possible to use variables in instream SYSIN control statements within a procedure. The APP9 project environment is created by using the same attributes and data sets as the APP3 and APP7 projects.

The JCL submitted to build the projects is shown in Example 3-38.

*Example 3-38 JCL that is used to build the APP9 environment*

```

//APP9BLD1 JOB ,MSGCLASS=H,NOTIFY=KWRES04
//BLDAPP9 EXEC APPNBLD1,PROJHLQ=APP9
//

```

---

The JCL starts a procedure that is named APPNBLD1. This procedure is a JCL proc with instream SYSIN. It uses one symbolic variable that is named PROJHLQ. The value of PROJHLQ is set to APP9. Example 3-39 shows the APPNBLD1 procedure, which was stored in a PROCLIB.

*Example 3-39 APPNBLD1 procedure*

```

//APPNBLD1 PROC PROJHLQ=APPN
//          EXPORT SYMLIST=(PROJHLQ)
//          SET PROJHLQ=&PROJHLQ
//SETUP001 EXEC PGM=IDCAMS
//* APPX IS COMMON HLQ FOR ALL APPN PROJECTS
//TESTDATA DD DISP=OLD,DSN=APPX.PARTS.TESTDATA
//SYSPRINT DD SYSOUT=A
//SYSIN DD *,SYMBOLS=JCLONLY
LISTC ENT(CAT&PROJHLQ) NAME
  IF LASTCC > 0 -
    THEN DO
      DEFINE -
        USERCATALOG -
          (NAME (CAT&PROJHLQ) -
            STORCLAS(S1P03S01) -
            CYLINDERS(1 1))
        END

      IF LASTCC = 0 -
        THEN DO
          DEFINE ALIAS(NAME(&PROJHLQ) RELATE(CATAPP5))

```

```

        END
        IF LASTCC = 0 -
            THEN DO
                /*                                     */
                /* DEFINE THE VSAM DATA SETS          */
                /* -----                            */
                /*                                     */
                /* DELETE THE KSDS.                    */
                /* NOTE: THIS WILL ALSO DELETE ANY DEFINED AIX OR PATH. */
                DELETE &PROJHLQ..PARTS
            END
            SET MAXCC=0

                /* DEFINE THE KSDS.                    */
                DEFINE CLUSTER (NAME(&PROJHLQ..PARTS) STORCLAS(S1P03S01) ) -
                    DATA (KEYS(8 0) RECORDS(800 100) RECORDSIZE( 90 100) ) -
                    INDEX(RECORDS (500 500))

                IF LASTCC = 0 -
                    THEN DO
                        REPRO INFILE(TESTDATA) OUTDATASET(&PROJHLQ..PARTS)
                    END
                IF LASTCC = 0 -
                    THEN DO
                        /* DEFINE THE PART NAME AIX.                    */
                        DEFINE ALTERNATEINDEX (NAME(&PROJHLQ..PARTNAME.AIX) -
                            RELATE(&PROJHLQ..PARTS) KEYS(6 24) RECORDS(800 100) UPGRADE )
                        END

                        IF LASTCC = 0 -
                            THEN DO
                                /* DEFINE THE PART NAME PATH.                    */
                                DEFINE PATH (NAME (&PROJHLQ..PARTNAME.PATH) -
                                    PATHENTRY(&PROJHLQ..PARTNAME.AIX) )
                                END

                                IF LASTCC = 0 -
                                    THEN DO
                                        /* BUILD THE PART NAME AIX.                    */
                                        BLDINDEX INDATASET(&PROJHLQ..PARTS) -
                                            OUTDATASET(&PROJHLQ..PARTNAME.AIX)
                                        END

                                        IF LASTCC = 0 -
                                            THEN DO
                                                /* DEFINE THE PART SUPPLIER AIX.                    */
                                                DEFINE ALTERNATEINDEX (NAME(&PROJHLQ..PARTSUPP.AIX) -
                                                    RELATE(&PROJHLQ..PARTS) KEYS(30 55) RECORDS(800 100) NOUPGRADE)
                                                END

                                                IF LASTCC = 0 -
                                                    THEN DO
                                                        /* DEFINE THE PART SUPPLIER PATH.                    */
                                                        DEFINE PATH (NAME (&PROJHLQ..PARTSUPP.PATH) -
                                                            PATHENTRY(&PROJHLQ..PARTSUPP.AIX) )
                                                    END

```

```

        END

        IF LASTCC = 0 -
          THEN DO
        /* BUILD THE PART SUPPLIER AIX. */
          BLDINDEX INDATASET(&PROJHLQ..PARTS) -
          OUTDATASET(&PROJHLQ..PARTSUPP.AIX)
          END
        /* DELETE THE ESDS. */
          DELETE &PROJHLQ..PARTS.LOG
          SET MAXCC=0

        /* DEFINE THE ESDS. */
          DEFINE CLUSTER (NAME(&PROJHLQ..PARTS.LOG) STORCLAS(S1P03S01) -
            RECORDS(100 100) RECORDSIZE(80 80) -
            NONINDEXED)

        /* ALLOCATE THE NONVSAM DATA SETS */

          DELETE &PROJHLQ..PROGRAMS.SOURCE
          DELETE &PROJHLQ..PROGRAMS.LOADLIB
          DELETE &PROJHLQ..PROJECT.JCL
          SET MAXCC=0

        /* ALLOCATE THE SOURCE PDS */
        ALLOC DSNAME('&PROJHLQ..PROGRAMS.SOURCE') NEW RECFM(F B) LRECL(80) -
          BLKSIZE(27920) DSORG(PO) DIR(50) SPACE(10,2) CYLINDERS -
          STORCLAS(S1P03S01)

        IF LASTCC = 0 -
          THEN DO
        /* ALLOCATE THE LOAD LIBRARY */
          ALLOC DSNAME('&PROJHLQ..PROGRAMS.LOADLIB') NEW RECFM(U) LRECL(0) -
          BLKSIZE(32760) DSORG(PO) DIR(50) SPACE(5,1) CYLINDERS -
          STORCLAS(S1P03S01)
          END

        IF LASTCC = 0 -
          THEN DO
        /* ALLOCATE THE JCL LIBRARY */
          ALLOC DSNAME('&PROJHLQ..PROJECT.JCL') NEW RECFM(F B) LRECL(80) -
          BLKSIZE(27920) DSORG(PO) DIR(50) SPACE(5,1) CYLINDERS -
          STORCLAS(S1P03S01)
          END
        /*

```

---

## 3.8 Provisioning

The scenarios that were described in this chapter show you that IDCAMS can be part of a provisioning process. Other entities that are beyond the scope of IDCAMS can be required by the project, but the VSAM and NONVSAM data sets can be allocated and possibly test data loaded.

Members also can be loaded to contain program sources, instructions, documentation, or instructions about how to use the environment. There are many possibilities and they depend on the local standards and procedures.

The data sets provisioning also can be part of a larger workflow for setting up application environments for the development, testing, and training purposes.

## 3.9 Thank you

Thank you to everyone who contributed to this paper. For more information about sending feedback, see “Comments welcome” on page xi. We look forward to the next paper we produce by working with people such as you.

# Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this paper.

## IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Note that some publications that are referenced in this list might be available in softcopy only:

- ▶ *A Practical Guide to ICF Catalogs*, SG24-8262
- ▶ *VSAM Demystified*, SG24-6105

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft, and other materials at the following website:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Other publications

The following publications are also relevant as further information sources:

- ▶ *Access Method Services*, SC23-846
- ▶ *DFSMS Managing Catalogs*, SC23-6853
- ▶ *DFSMS Using data sets*, SC23-6855

## Online resources

The following SHARE website also is relevant as a further information source:

<http://share.org>

## Help from IBM

IBM Support and downloads:

[ibm.com/support](http://ibm.com/support)

IBM Global Services:

[ibm.com/services](http://ibm.com/services)







REDP-5389-00

ISBN 0738455555

Printed in U.S.A.

Get connected

