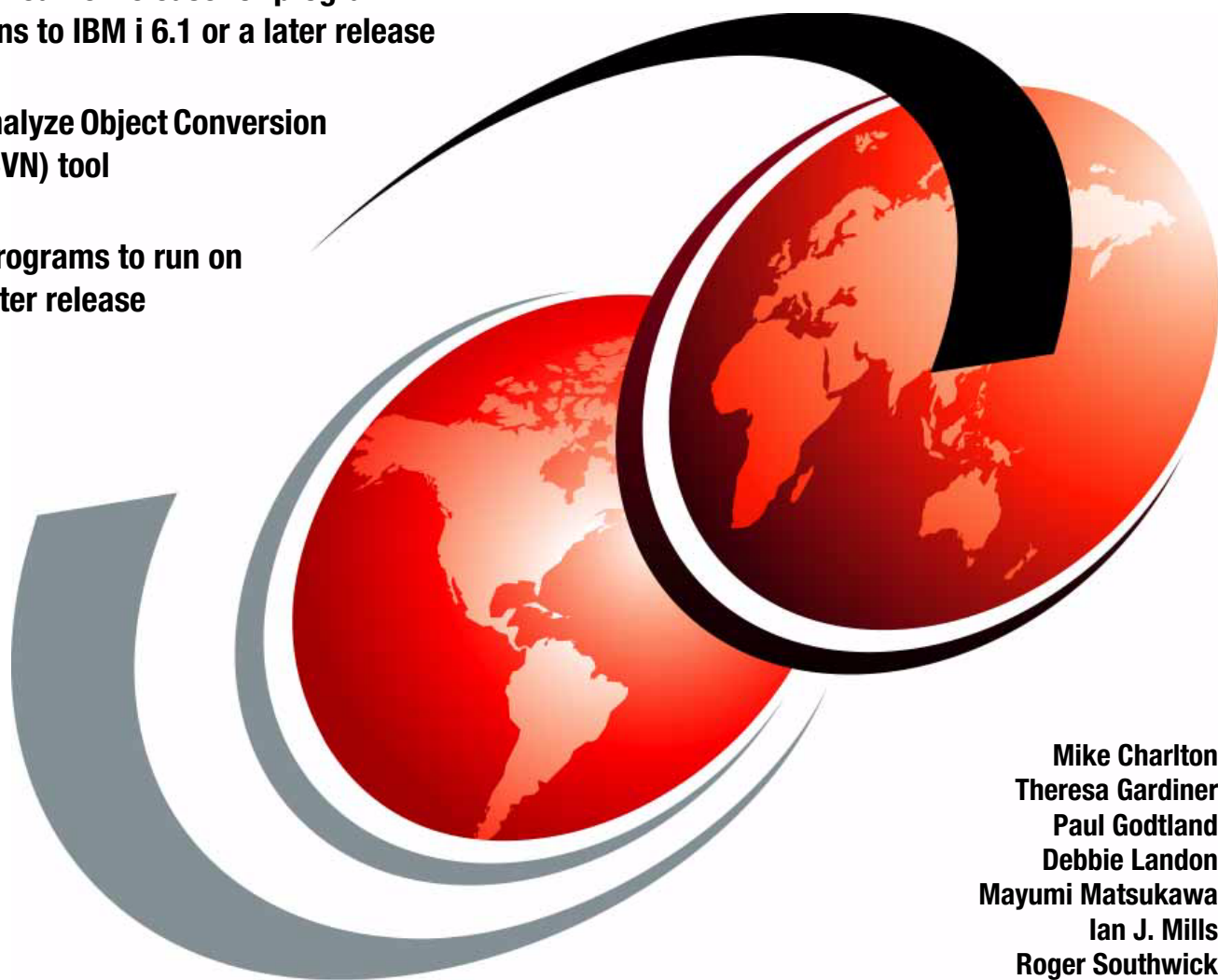


IBM i Program Conversion: Getting Ready for 6.1 and Beyond

Prepare an earlier release for program conversions to IBM i 6.1 or a later release

Use the Analyze Object Conversion (ANZOBJCVN) tool

Convert programs to run on 6.1 or a later release



Mike Charlton
Theresa Gardiner
Paul Godtland
Debbie Landon
Mayumi Matsukawa
Ian J. Mills
Roger Southwick



International Technical Support Organization

**IBM i Program Conversion:
Getting Ready for 6.1 and Beyond**

March 2010

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

Second Edition (March 2010)

This edition applies to IBM i version 6.1 and later.

© Copyright International Business Machines Corporation 2010. 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

Notices	v
Trademarks	vi
Preface	vii
The team who wrote this paper	viii
Now you can become a published author, too!	x
Comments welcome	x
Stay connected to IBM Redbooks	x
Summary of previous versions	xiii
Summary of second edition versions	xiii
March 2010, second edition of this paper	xiii
Summary of changes made to first edition versions	xiii
April 2009 update	xiv
March 2009 update	xiv
November 2008 update	xiv
May 2008, first edition of this paper	xiv
Summary of draft versions	xv
January 2008 draft update	xv
November 2007 draft update	xv
October 2007 draft update	xvi
July 2007 draft update	xvi
Chapter 1. Overview of the program conversion process	1
Chapter 2. Introduction to program conversion	3
2.1 Key values of IBM i	4
2.2 What program conversion is	4
2.3 Why program conversion	6
2.3.1 Integrity	6
2.3.2 Performance improvements	7
2.3.3 Functional improvements	8
Chapter 3. Preparing for conversions	9
3.1 Analyzing objects	10
3.1.1 Terminology definitions	10
3.1.2 Required PTFs	11
3.1.3 Command usage notes	12
3.1.4 Conversion time estimates	13
3.1.5 Alternate method for estimating conversion times	16
3.2 Collecting information	17
3.2.1 Parameters of the ANZOBJCVN *COLLECT command	18
3.2.2 Examples of collection commands	21
3.2.3 Files used for data collections	22
3.3 Creating reports	23
3.3.1 Parameters of the ANZOBJCVN *REPORT command	25
3.3.2 Examples of report commands	26
3.3.3 Understanding the conversion reports	30
3.4 Programs that will not convert	33

3.5 Querying analyzed data	36
3.6 Considerations for application developers	37
3.6.1 Will your application convert	37
3.6.2 Other decision criteria	38
3.6.3 Adaptive code generation	39
3.6.4 Fixes and testing	39
3.7 Other object conversions	40
3.7.1 Java programs	40
3.7.2 Integrated file system names	43
3.7.3 Spooled files	43
3.8 Removing unsupported products	44
Chapter 4. Program conversion methods	45
4.1 Overview of program conversion	46
4.2 Conversion using STROBJCVN	47
4.2.1 STROBJCVN usage notes	48
4.2.2 Parameters of the STROBJCVN command	50
4.2.3 STROBJCVN OPTION(*CHECK)	52
4.2.4 STROBJCVN OPTION(*CONVERT)	55
4.3 Conversion of program objects during restore	57
4.4 Conversion the first time a program is run or called	59
4.5 Conversion times and how to improve them	59
Appendix A. MRI PTFs	61
Appendix B. File field definition tables	65
QAIZACVN: information for analyzed libraries	66
QAIZADIR: information for analyzed directories	69
QAIZAOBJ: information for analyzed integrated file system objects	70
QAIZASPL: information for analyzed spooled files	72
Appendix C. IBM software skip-ship products requiring conversion	73
Related publications	75
Other publications	75
Online resources	75
How to get Redbooks	75
Help from IBM	76

Notices

This information was developed for products and services offered in the U.S.A.

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 give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: 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 states 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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.

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 the names and addresses used by an actual business enterprise 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.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AS/400®	Lotus Enterprise Integrator®	Redbooks®
DB2®	Lotus®	Redpaper™
Domino®	MQSeries®	Redbooks (logo)  ®
eServer™	OS/400®	Sametime®
i5/OS®	PartnerWorld®	System i5®
IBM®	POWER5™	System i®
Integrated Language Environment®	POWER6®	WebSphere®
iSeries®	QuickPlace®	
Language Environment®	Quickr™	

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Preface

IBM® delivered IBM i 6.1 in March 2008. With 6.1, IBM exploits the capabilities of the Machine Interface (MI) architecture to significantly improve programs. Programs can benefit from better performance, a range of new operating system and processor capabilities, and even stronger system integrity. To enable these improvements, all MI programs created for previous releases must be converted to run on 6.1 or a later release. MI programs include integrated language environment (ILE) and original program model (OPM) programs.

To convert a program, its creation data, which is a subset of observability, must be available. MI programs retain creation data by default, so that most programs can be converted, going all the way back to programs that were originally created for System/38. Even if an option was chosen to remove creation data from external access, Licensed Internal Code (LIC) can still access the creation data if the target releases of the program and its constituent modules are V5R1 or later. Thus a program lacks sufficient creation data for conversion only if the program or at least one of its modules was created for IBM OS/400® V4R5 or an earlier release *and* creation data was explicitly removed.

You can run the Analyze Object Conversion (ANZOBJCVN) command on V5R4 or V5R3 to help plan for your upgrade. The ANZOBJCVN command identifies programs that will not convert to run on 6.1 or later releases, if any, and estimates the times that are required for the program conversions. It also provides information about two other types of conversions: integrated file system names and spooled files. You can access the ANZOBJCVN command by loading and applying the appropriate PTFs that are listed in this paper.

This IBM Redpaper™ publication is intended for system administrators and field technicians to help them understand and prepare for upgrading or migrating to 6.1 or a later release. It explains what the program conversion is and why programs are being created or recreated on 6.1 or a later release. It then provides detailed steps for using the ANZOBJCVN tool on V5R3 or V5R4 to prepare for object conversions. Finally, it discusses the program conversion methods for converting programs to run on 6.1 or a later release.

Note: The first edition of this paper was entitled *IBM i5/OS Program Conversion: Getting Ready for i5/OS V6R1*, REDP-4293-00. The title of this second edition has been slightly modified to *IBM i Program Conversion: Getting Ready for 6.1 and Beyond*, REDP-4293-01, to reflect the new naming conventions for IBM i.

The team who wrote this paper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Rochester Center.



Mike Charlton works at the IBM Hursley Laboratories in the U.K. as an IBM i Platform Specialist. He is a former high school mathematics teacher. He joined IBM in 1998 as a Software Support Specialist working in the IBM AS/400® support center in the United Kingdom (U.K.), where he specialized in work management, performance, and performance analysis tools. He was promoted to Advisory level in 2002.



Theresa Gardiner is a Staff Software Engineer in the Rochester IBM i Support Center. She started her career with IBM in 1977 as a Customer Engineer in Milwaukee, Wisconsin. She shifted from hardware to software support in the late 1980s. When Theresa moved to the Rochester Support Center, she started on the Language and Utilities team, of which she is currently the team leader. She worked at the Rochester Support Center during the CISC-to-RISC conversion time frame. She is a support expert for the S36E and S38E languages and utilities.



Paul Godtland is a Senior Software Engineer in Rochester, Minnesota who designs and implements IBM i Licensed Internal Code. He was the technical lead for the MI program conversion project. Most of his 30+ years with IBM have been focused on the System/38, AS/400, IBM eServer™ iSeries®, System i®, and IBM i family of systems. His past roles include membership on teams that designed and implemented the Integrated Language Environment® (ILE), the AS/400 CISC-to-RISC conversion, and teraspace. Paul also helped achieve the initial AS/400 C2 security rating and is involved with ongoing system integrity improvements.



Debbie Landon is an IBM Certified Senior IT Specialist in the IBM ITSO, Rochester Center. Debbie has been with IBM for 26 years, working first with the S/36 and then the AS/400, which has since evolved to the IBM i platform. Before joining the ITSO in November of 2000, Debbie was a member of the IBM PartnerWorld® for Developers iSeries team, supporting IBM Business Partners in the area of Domino® for iSeries.



Mayumi Matsukawa is an IT Specialist for the IBM i Technical Support team in Japan. She joined IBM in 1998 and has eight years experience working with IBM i. Her areas of expertise include Domino and related Lotus® products, LPAR, clustering, cross-site mirroring (XSM), and DB2® for i5/OS®. Mayumi provides technical support and education for IBM specialists, Business Partners, and clients.



Ian J. Mills is an advisory IT Software Support Specialist in the iSeries support center in the U.K. He joined IBM directly after college in 1995 as a Software Support Specialist and was promoted to Advisory level in 2001. In addition to working in support, he has been directly engaged in more than 200 system upgrade contracts, as well as numerous system recoveries for customers throughout the United Kingdom and Ireland. Ian re-certified as an i5/OS specialist in March 2007. Since July 2007, he has been an IBM Availability Manager in the UK. Ian is a graduate of the University of Portsmouth and has a degree in IT.



Roger Southwick is a Staff Software Engineer at IBM in Rochester, Minnesota. He started with IBM in 1987 on Release 1 of the AS/400. Since 1996 he has enhanced and supported the Integrated Language Environment binder, and in 2009 he accepted technical ownership of the ANZOBJCVN and STROBJCVN commands. He holds BSEE and MCIS degrees from the University of Minnesota.

Thanks to the following people for their contributions to this project:

Margaret Fenlon
Kary Garvey
Ian Jarman
Rodney Johnson
Sherri McMeeking
Laurie Miller
Gerri Passe
Rita Roque
Gottfried Schimunek
Michelle Schlicht
Tom Severson
Judy Trousdell
Kevin Vette
IBM Rochester development team

Rodolfo Segoviano Rayas
IBM Guadalajara, Mexico

Al Barsa, Jr.
Barsa Consulting Group, LLC

Thomas Gray
Joanna Pohl-Misczyk
Craig Schmitz
Jenifer Servais
ITSO, Rochester Center

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

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

- ▶ Send your comments in an e-mail to:

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/pages/IBM-Redbooks/178023492563?ref=ts>

- ▶ 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>

Summary of previous versions

Here we provide a history of the various technical changes that were made to this IBM Redpaper publication since the initial draft version was made available in July 2007. Since then the first edition (*IBM i5/OS Program Conversion: Getting Ready for i5/OS V6R1*, REDP-4293-00) was published in May 2008 and the second edition (*IBM i Program Conversion: Getting Ready for 6.1 and Beyond*, REDP-4293-01) in March 2010.

Note: Minor corrections and editorial changes are not identified.

Summary of second edition versions

The second edition of this paper (REDP-4293-01) was published in March 2010. In this section we describe the series of technical changes that were made in this second published edition of this paper.

March 2010, second edition of this paper

In addition to the progressive changes made to the first edition of this paper, this second edition includes the following information:

- ▶ Global change of references of “V6R1” to “6.1 or a later release”
- ▶ The ANZOBJCVN command now has new code PTFs that are listed in 3.1.2, “Required PTFs” on page 11
- ▶ Updated information in “Conversion time estimates using multiple threads” on page 15
- ▶ Updated certain query examples in 3.5, “Querying analyzed data” on page 36
- ▶ Updated information in 3.6.3, “Adaptive code generation” on page 39
- ▶ Updated information in “Multi-threading” on page 48 and “Limits to monitoring progress” on page 49. Also added a new sub-section entitled “Converting for different processors” on page 49
- ▶ Added additional information to 4.5, “Conversion times and how to improve them” on page 59

Summary of changes made to first edition versions

The first edition of this paper (REDP-4293-00) was published in May 2008. In this section we describe the series of technical changes that were made in the first published edition of this paper.

April 2009 update

This publication includes the following updates from April 2009:

- ▶ 3.1.2, “Required PTFs” on page 11
The ANZOBJCVN command has new code PTFs listed in this section. The main change is that conversion time estimates can now take into account the consideration of using multiple threads.
- ▶ 3.1.4, “Conversion time estimates” on page 13
This section now includes information about how to specify multiple threads for a conversion time estimate.

March 2009 update

This publication includes the following updates from March 2009:

- ▶ 3.1.5, “Alternate method for estimating conversion times” on page 16
A new section was added that provides information about an alternate method for estimating conversion times.
- ▶ Conversion times Note boxes on pages 16, 17, and 23
The Conversion times note boxes were updated with more detailed information about the various factors that can affect the conversion time estimates.

November 2008 update

This publication includes the following updates from November 2008:

- ▶ 3.1.2, “Required PTFs” on page 11
Updated PTF information in the Important box
- ▶ 3.3.2, “Examples of report commands” on page 26
Updated first example

May 2008, first edition of this paper

In addition to the progressive changes made to the draft versions of this paper, this first edition includes the following information:

- ▶ 3.1.2, “Required PTFs” on page 11
The ANZOBJCVN command now has new code PTFs that are listed in this section. The main change is that conversion time estimates now take into account partitions that have less than one full processor.
- ▶ 3.1.4, “Conversion time estimates” on page 13
This section now includes information about how to specify a partial processor when providing estimates for a different system than the one on which the ANZOBJCVN command is run.
- ▶ Chapter 4, “Program conversion methods” on page 45
This chapter now includes more discussion about the concurrent processing potential of each of the three conversion methods. In addition, information was added about making sure that you have installed PTF MF44237 before starting conversions of any kind on a

V6R1 system. This PTF prevents rare occurrences of a system hang during program conversion.

Summary of draft versions

Prior to the release of this first edition, this paper was made available in draft mode with the preview of i5/OS V6R1 in July 2007. In this section we describe the series of technical changes that were made in previous versions of the paper. Minor corrections and editorial changes are not identified.

January 2008 draft update

The fourth draft of this paper included the following information:

- ▶ “Preface” on page vii
Updated the preface and abstract with new introductory text.
- ▶ Chapter 1, “Overview of the program conversion process” on page 1
Created this chapter to provide a brief summary of the process to help plan and implement program conversion. This information was in an appendix in earlier drafts of this paper.

November 2007 draft update

The third draft of this paper included the following information:

- ▶ Chapter 3, “Program conversion methods”
We added a recommendation to remove unsupported products on V6R1 before upgrading.
This chapter includes the following changes:
 - 3.2.1, “Parameters of the ANZOBJCVN *COLLECT command”
 - Added information about the QSYS2 library, in that no action is required
 - Enhanced the information in the Subtree parameter discussion
 - 3.2.2, “Examples of collection commands”
More information was added about the *CVNPRB parameter, and the associated window captures were updated.
 - 3.4, “Programs that will not convert”
Additional information was added.
 - 3.5, “Querying analyzed data”
Information was added about how queries are required to report on the ANZOBJCVN results if the files of the command have been moved to another library.
 - 3.6, “Considerations for application developers”
More information was added about adaptive code generation (ACG).
 - 3.7.3, “Spooled files”
This section was updated with advice to perform spooled file housekeeping before upgrading.
- ▶ Chapter 4, “Program conversion methods”

This chapter includes the following updates:

- 4.1, “Overview of program conversion” (new section added)
- 4.2, “Conversion using STROBJCVN” (content revised)
Information about multi-threading was added to 4.2.1, “STROBJCVN usage notes”.
- 4.3, “Conversion of program objects during restore” (content revised)
- 4.5, “Conversion times and how to improve them” (new section added)

▶ Appendix A, “MRI PTFs”

The Analyze Object Conversion (ANZOBJCVN) CL command is distributed as PTFs for V5R3 and V5R4. In this draft version of the paper, we provided the PTF numbers for the different national languages.

▶ Appendix B, “File field definition tables”

This appendix was changed to provide more of a quick reference guide.

▶ Appendix C, “IBM software skip-ship products requiring conversion”

This new appendix provides a list of skip-ship IBM program products that must be converted when upgrading to V6R1.

October 2007 draft update

The second draft of this paper included the following information:

▶ Chapter 2, “Preparing for conversions”

- We added information for the new *CVNPRB parameter of the ANZOBJCVN *REPORT command. This new report parameter provides a detailed report of program objects that will not convert for each library specified in the previous library collection.
- We added a description of the new *ALLAVL parameter of the ANZOBJCVN *REPORT command, which replaced the older default *ALL parameter. Specifying the *ALLAVL parameter causes reports to be generated for each type of information that has been collected.

▶ Appendix A, “MRI PTFs”

The Analyze Object Conversion (ANZOBJCVN) CL command is distributed as PTFs for V5R3 and V5R4. This version of this paper contained only the PTF numbers for the *updated* versions of the English PTFs for V5R3 (SI28425) and V5R4 (SI28415).

July 2007 draft update

The first draft of this paper included the following information:

- ▶ Chapter 1, “Introduction to program conversion”
- ▶ Chapter 2, “Preparing for conversions”
- ▶ Chapter 3, “Program conversion methods”

This chapter contains only minimum information about the conversion process and methods.

▶ Appendix A, “MRI PTFs”

The Analyze Object Conversion (ANZOBJCVN) CL command is distributed as PTFs for V5R3 and V5R4. This draft version of the paper contained only the PTF numbers for the *original* versions of the English PTFs for V5R3 (SI27515) and V5R4 (SI25550).

- ▶ Appendix B, “Program conversion summary”
- ▶ Appendix C, “File field definitions”



Overview of the program conversion process

In this chapter, we summarize the process for you to plan and implement a program conversion. It serves as a brief and concise guide to the steps required for the program conversion process. The steps highlighted in this overview contain references to other chapters in this paper for more in-depth information. We recommend that you read this entire paper to completely understand the program conversion required for 6.1 or a later release.

Note: Planning for program conversion starts with your V5R3 or V5R4 system, with conversion occurring after you upgrade to 6.1 or a later release. If you plan to upgrade from V5R3 or V5R4 directly to a release later than 6.1, you still must perform program conversions after the upgrade. However, if you performed program conversions on 6.1, you do not need to perform these conversions again when moving from 6.1 to a later release.

The following steps briefly summarize the process for you to plan and implement program conversion.

Step 1: Apply V5R4 or V5R3 PTFs to enable the ANZOBJCVN command

In most cases, if you order the MRI PTF for your preferred national language, the requisite code PTFs are also automatically shipped. Refer to 3.1.2, “Required PTFs” on page 11.

Step 2: Collect information about objects on your system

Run the ANZOBJCVN OPTION(*COLLECT) command. Each collection of a given type replaces the data in the collection file or files for that type. Therefore, carefully consider the information provided in 3.2, “Collecting information” on page 17, and particularly in 3.2.3, “Files used for data collections” on page 22.

Step 3: Review and understand collected information

Run the ANZOBJCVN OPTION(*REPORT) command, and optionally run your own queries over the collected data. Consider the following reports:

- ▶ Report on programs that do not convert with the *CVNPRB parameter
- ▶ Report summary information, such as estimated conversion times for each library, with the *LIBSUM parameter

See 3.3, “Creating reports” on page 23, and particularly 3.3.3, “Understanding the conversion reports” on page 30.

Step 4: Contact your application provider, if necessary

If you find that you have purchased an application that will not convert and you do not have the source code to recompile it, contact the application provider to get a version of the application that can run on 6.1 or a later release. See 3.4, “Programs that will not convert” on page 33.

Step 5: Remove unsupported products, and then upgrade to 6.1 or later

Remove unsupported licensed program products (LPP) and other unsupported products before migrating to 6.1 or a later release. See 3.8, “Removing unsupported products” on page 44.

After all of the preparatory tasks for object conversion are completed, refer to *Installing, Upgrading, or Deleting i5/OS and Related Software*, SC41-5120-10, at the following Web site:

<http://www.ibm.com/support/docview.wss?uid=pub1sc41512010>

Step 6: Convert objects on 6.1 or later release

Choose the combination of conversion options that best suits your situation. Refer to the following sections:

- ▶ 4.2, “Conversion using STROBJCVN” on page 47
- ▶ 4.3, “Conversion of program objects during restore” on page 57
- ▶ 4.4, “Conversion the first time a program is run or called” on page 59



Introduction to program conversion

Program conversion is required for all systems that upgrade to IBM i 6.1 or a later release. Use the information in this paper and the tools provided for i5/OS V5R3 and V5R4 to prepare for this conversion. Preparing well before you intend to upgrade promotes a smooth transition.

Overview: For a brief overview about how to prepare for an upgrade, see Chapter 1, “Overview of the program conversion process” on page 1.

2.1 Key values of IBM i

The IBM i operating environment integrates a trusted combination of relational database, security, Web services, networking, and storage management capabilities. This integration promotes a simplicity of operation that is particularly appreciated by smaller businesses. IBM i also provides a broad and highly stable middleware foundation for efficiently deploying business processing applications, with support for over 5,200 solutions from more than 2,500 ISVs. IBM i solutions are offered through an extensive, highly skilled worldwide network of IBM Business Partners that is backed by the trusted services and support infrastructure of IBM.

IBM i has a highly scalable and virus-resistant architecture with a proven reputation for exceptional business resiliency. For large enterprises, business process resilience and the ability to meet and exceed demanding service levels of multiple business units is a top priority. Over many years and in many businesses, IBM i has established a reputation as one of the top operating environments in the industry for resilient application deployment. Companies routinely trust the IBM i operating environment to deploy their most critical business applications.

IBM i offers a variety of expansion options to help a company maximize return on its IT investments. Most important to nondisruptive growth is the IBM i technology independent Machine Interface (MI). Proven over many years and technology generations, the IBM i MI has protected applications from changing hardware devices and processor generations, even enabling applications to be upgraded to new technology without recompilation.

2.2 What program conversion is

To take advantage of the major improvements in system design and processor capabilities, 6.1 and later releases require all applications from a release prior to 6.1 that use the MI to be re-created before they can be run. These MI programs include Integrated Language Environment (ILE) and Original Program Model (OPM) programs. The conversion replaces existing program objects, but each program object retains attributes such as the name, library, and owning user profile. This conversion is a one-time process on each object. All MI programs with sufficient creation data can be converted, independently of the creation release. In fact, even programs initially created for System/38 can be converted.

IBM i Portable Application Solutions Environment (PASE) and Java™ programs are not based on MI. Therefore, they are not directly affected. However, Java programs in directories will also undergo a simple conversion. See 3.7.1, “Java programs” on page 40, for details.

Programs have a new format beginning with 6.1. Therefore, programs that were created for earlier releases must be converted on 6.1 or a later release before they can run. Similarly, programs created on 6.1 or a later release must be converted before they can run on V5R4 or an earlier release.

Software design and hardware used by the IBM i platform are now at a point where significant improvements to vital system aspects, such as integrity, performance, and functionality, can be provided. This is done by taking advantage of the system architecture to transform programs. See 2.3, “Why program conversion” on page 6, for more details.

Program conversion or transformation can occur in three different ways:

- ▶ During program restore or application installation if the application was generated for a release prior to 6.1
- ▶ Explicitly scheduled
- ▶ When a program is first run

You can freely mix and match conversion methods to suit your own situation. See Chapter 4, “Program conversion methods” on page 45, for details about the conversion methods.

To help customers prepare for program conversion, the Analyze Object Conversion (ANZOBJCVN) command is available in the form of PTFs that can be applied to V5R3 and V5R4 systems. (These two releases can be directly upgraded to 6.1.) By using the ANZOBJCVN command, clients can analyze their systems to help determine whether there are program objects that will not convert. It also provides an estimate of the time required for program conversion.

We recommend that you use the ANZOBJCVN CL command prior to upgrading to 6.1 or a later release. By using this tool before upgrading an existing system or before migrating to new hardware running 6.1 or a later release, adequate planning can be completed and any necessary proactive steps can be taken. If a program will not convert, it must be recompiled, or a new version of the program must be acquired from your software provider.

Use the ANZOBJCVN command on all V5R3 and V5R4 i5/OS logical partitions (LPARs) that you intend to upgrade or migrate to 6.1 or a later release. Also consider including independent auxiliary storage pools (ASPs) in the planning stage, because objects that require conversion might reside on them. For additional details about using the ANZOBJCVN command, refer to Chapter 3, “Preparing for conversions” on page 9.

Object conversion also occurs during a restore to V5R3 or V5R4 systems for programs that are created on 6.1 or later but with an earlier target release. See 3.1.2, “Required PTFs” on page 11, for the PTFs that are required on previous releases.

Figure 2-1 shows when program conversion occurs or is required. For more information about how IBM i 6.1 or a later release can take advantage of IBM POWER6® or later processors, see 2.3.2, “Performance improvements” on page 7, and 3.6.3, “Adaptive code generation” on page 39.

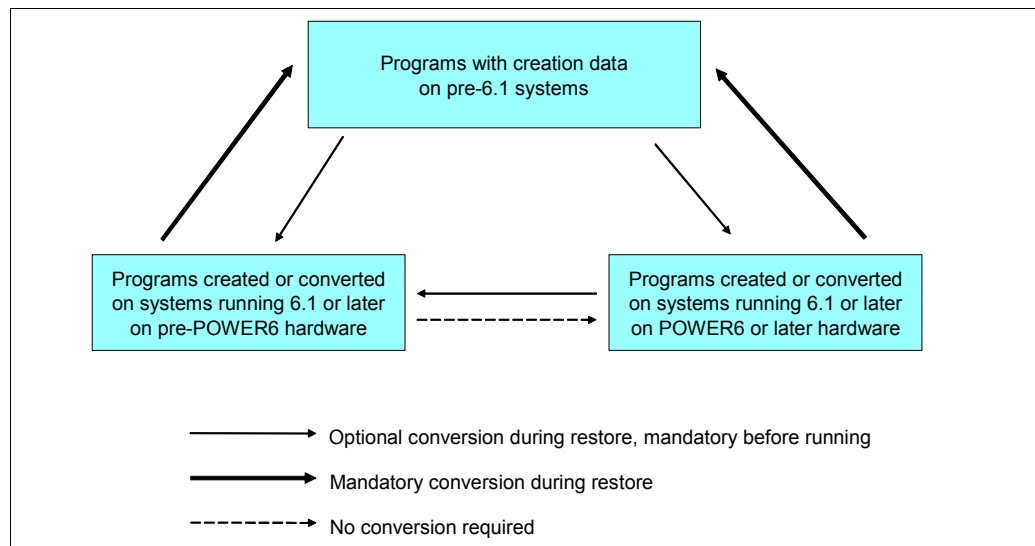


Figure 2-1 Program conversion chart

Historical perspective

This conversion is the third in the history of the MI architecture. The first conversion occurred when moving from System/38 to OS/400 V1R1 in 1988. The second conversion occurred for OS/400 V3R6 in 1995, when upgrading from 48-bit to 64-bit addresses and running a different hardware instruction set. This conversion is simpler than the previous ones. For example, most other types of objects are not changing this time. Program conversions are also much faster now.

2.3 Why program conversion

Programs are being created or recreated starting on 6.1 because several key system aspects can then be significantly enhanced. Enhancements to integrity, performance, and function are included. Existing programs are converted and newly-created programs get the same benefits.

In addition to those tangible results, program conversion is a way of flexing the technical vitality muscles of the MI architecture. Even though the MI architecture is not new, it provides refresh and improvement capabilities that are among the best in the industry. The combination of refreshing all MI programs and retaining their creation data gives IBM i users extreme flexibility for dealing with future hardware, system design, and industry-wide changes.

2.3.1 Integrity

A system has integrity when it performs only the intended or designed functions correctly. An immediate result of re-creating all MI programs is that any unsupported alterations (for example, changing the hardware instruction stream with a service tool) are eradicated. Because they are gone, they have no chance of interfering with the intended uses of your system.

Further, the system's unique ability to remove any potential future code corruption is extended. Now any MI-based application that can run on your system can be refreshed at any time. For example, if you are worried about an application that you want to purchase from a vendor with whom you are not yet familiar, you can convert the program as you install it.

Conversion re-creates a program from its MI architecture constructs, so that system integrity cannot be attacked. A converted program conforms to MI semantics, which only allow defined operations on supported object types. Further, unlike many other systems, you do not have to worry about a continual race to update your virus scanning software before someone sends a new form of attack program. Instead of worrying about which sequences of hardware instructions might cause problems, you can refresh a program from its higher level MI operations, letting the system generate appropriate hardware instructions.

Of course, the MI is typically enhanced for each release, but these changes become part of the coherent whole. Each system release brings improvements, but no requirement to update to be safe from attacks developed since the last release shipped.

Another important integrity improvement beginning with 6.1 is that system state programs must be part of IBM i in order to be loaded. In past releases, you could optionally allow system state programs to be loaded, regardless of whether they were part of the operating system. System state was designed for operating system use. Therefore, in the past, customers could inadvertently undermine the operating system integrity. That led to situations where programs that ran with system state, but were not designed to interoperate correctly

with the operating system, caused problems as severe as a system crash. It was as though those other programs could withdraw money from the IBM i metaphorical bank account of operating system resources, eventually leading to an overdraft.

2.3.2 Performance improvements

Performance improvements were a key driver for program conversion. Commonly run operations were improved, performance pitfalls were removed, new create-time options were added to allow extra processing focus on two aspects of performance, and future performance improvements were enabled.

Activation (readying a program to run), procedure (function) calls, and pointer use can now all run faster. The amount of improvement depends upon the extent to which these operations affect each application's overall performance.

Teraspace

Teraspace, a large contiguously addressable area local to each job, is now implemented with direct hardware support instead of software emulation. In one severe example on a prior release, an ISV application stressed the software emulation so much that the application's initial performance was degraded six times overall. When the problem was identified, the application was re-created by using a Licensed Internal Code Option (LICOPT) that eliminated most of the overhead. Now that teraspace has direct hardware support, this potential performance pitfall has been eliminated, without any special create-time considerations.

Even for programs that do not suffer from emulation overhead, there are general benefits from direct hardware use. For example, a commonly used, memory-intensive IBM i PASE application's performance improved by more than 15%. This occurred even though IBM i PASE programs are not converted, because IBM i PASE and MI programs interact using teraspace. A general improvement in memory handling efficiency also benefits ILE users of teraspace.

New create-time options

Two new create-time options are now offered on the Create Service Program (CRTSRVPGM) and Create Program (CRTPGM) CL commands for programs that target 6.1 or later. Neither option requires any source code changes. The ARGOPT parameter further optimizes all the calls between procedures in the service program or program that was created by using that option. Performance benefits up to 20% overall have been measured in call-intensive applications.

Service program activation can now also be deferred by using the *DEFER qualifier on the BNDSRVPGM parameter. Deferred service program activation occurs only when needed, and not when the program that binds to these service programs is activated. Thus you can choose not to pay a start-up price for functions that are only rarely used in mainline processing.

Adaptive code generation

Adaptive code generation is another performance enhancement starting in 6.1 that is particularly interesting to clients who purchase the newest hardware. In the past, the code generated for programs created for a given operating system release could use only a least common denominator of hardware features. That allowed programs to run on any hardware supported by the release, but it meant that some new hardware features could not be exploited for several releases.

In contrast, MI programs now, by default, use the available features of the hardware on which they are created. For example, an MI program that targets 6.1 or a later release and is created on a system based on POWER6 processors includes hardware instructions or other hardware features that first became available on POWER6. If these programs are moved to a system based on an older processor, the programs are automatically converted, so that they use only hardware features available on all supported processors.

See the *System i ILE Concepts Version 6 Release 1*, SC41-5606, manual for more information about the new create-time options and adaptive code generation ability:

<http://publib.boulder.ibm.com/infocenter/iserics/v6r1m0/topic/books/sc415606.pdf>

Also see 3.6.3, “Adaptive code generation” on page 39.

2.3.3 Functional improvements

Functional improvements starting with 6.1 are centered on the theme of making it easier to develop and deploy applications. Perhaps the biggest change is that now all MI programs are capable of using teraspace. In the past, you had to be careful to avoid passing teraspace addresses to programs that were not teraspace capable. This constraint made it more difficult to take advantage of teraspace in places where a large contiguously addressable area would have been more efficient, or where use of 8-byte, untagged pointers to teraspace would have improved the performance of a C or C++ program. Now teraspace addresses can be handled by all MI programs, so that teraspace use is much easier.

Other functional improvements include the ability for each job to allocate about 100 times more teraspace, with these allocations strictly local to a job. In the past under some conditions, the smaller allocations could inadvertently be referenced between jobs. For example, testing might not detect erroneous between-job attempts to use this inherently job-local resource, allowing a potential application problem to escape to clients.

Another functional enhancement is the addition of thread-local static storage for MI programs. This storage class can make thread-enabled programming simpler. RPG, C++, and C languages all offer new syntax for accessing this storage class. RPG uses “THREAD(*CONCURRENT)” in the H Spec to specify thread-local use for an entire program. Alternatively, RPG individual variables can be designated to use thread local storage by adding the 'static(*allthread)' D Spec qualifier. C and C++ offer the “__thread” qualifier on individual variable declarations. See the *System i ILE Concepts Version 6 Release 1*, SC41-5606, manual at the following Web site for more information:

<http://publib.boulder.ibm.com/infocenter/iserics/v6r1m0/topic/books/sc415606.pdf>

The development of applications has also been eased by improving coverage for Performance Explorer (PEX) and trace functions. Programs no longer must be recompiled with special options before using these system functions.



Preparing for conversions

Important: The processes explained in this chapter are for use prior to an upgrade from i5/OS V5R3 or V5R4 to IBM i 6.1 or a later release. Use of the analyze process is not mandatory, but is highly recommended.

In this chapter we discuss the process of planning for object conversions. The object types that require conversion are programs in libraries, Java programs in the integrated file system, the names of some files in the integrated file system, and spooled files. While conversions do not occur until after upgrade, the planning for conversions should be completed on V5R3 or V5R4. Conversion planning should be completed on all systems and partitions that you intend to upgrade or migrate to 6.1 or a later release.

When preparing for conversion, use the guidance in this paper in conjunction with the information provided in the following resources:

- ▶ *Installing, Upgrading, or Deleting i5/OS and Related Software*, SC41-5120-10
<http://www.ibm.com/support/docview.wss?uid=pub1sc41512010>
- ▶ System i i5/OS Memo to Users Version 6 Release 1
<http://publib.boulder.ibm.com/infocenter/iserics/v6r1m0/topic/rzaq9/rzaq9.pdf>
- ▶ System i Upgrade planning Web page
<http://www-304.ibm.com/systems/support/i/planning/upgrade/index.html>

3.1 Analyzing objects

For i5/OS versions V5R3 and V5R4, PTFs are available that provide the Analyze Object Conversion (ANZOBJCVN) command. With this tool, clients can analyze objects and generate reports based on that analysis. The reports are then available in spooled files on the system from which the command is run.

The ANZOBJCVN command is a tool that helps you identify potential conversion difficulties, if any, and estimates the time required for program conversion. It also provides information about two other types of conversions: integrated file system names and spooled files.

The following types of objects in libraries can be analyzed and reported on by the ANZOBJCVN command:

- ▶ Programs (*PGM), specifically original program model (OPM) and Integrated Language Environment (ILE)
- ▶ Service programs (*SRVPGM)
- ▶ Modules (*MODULE)
- ▶ SQL packages (*SQLPKG)

In most cases, the objects analyzed by the ANZOBJCVN command can be converted for use on 6.1 or later. See Chapter 4, “Program conversion methods” on page 45, for more details about the conversion process.

The following types of objects in directories can also be analyzed and reported on by the ANZOBJCVN command:

- ▶ Stream files with attached Java programs that were optimized to run on the i5/OS operating system
Only stream files in the “root” (/), QOpenSys, and user-defined file systems are analyzed. See 3.7.1, “Java programs” on page 40, for more details.
- ▶ Names of integrated file system objects in the “root” (/) file system as well as user-defined file systems created with option CASE(*MONO)
See 3.7.2, “Integrated file system names” on page 43, for more details.

Spooled file data is also collected by the ANZOBJCVN command to provide information that might be useful when planning system upgrades. By default, conversion of spooled files occurs during a system IPL or independent auxiliary storage pool (ASP) vary on operation. For details about spooled file conversion, see 3.7.3, “Spooled files” on page 43.

3.1.1 Terminology definitions

You should be familiar with the following terminology, which is used in the reports that are generated by the ANZOBJCVN command:

State An attribute that controls how a machine interface (MI) program is run. It can be either a user state or system state. This state attribute controls whether the program has any access to system domain objects at high security levels. State also determines the type of access (no access, read only, or read and write) that a program has to user domain spaces. Application programs run with a user state, where most operating system programs run with a system state.

- Digital signatures** A validation that is used to ensure that an object has not been modified and that it came from the expected source. Most often, digital signatures are used as an object integrity check before the object is restored onto a system. Digital signatures can optionally be applied to several types of objects, including programs. They must be removed during program conversion because the program is changed, thus invalidating any previous digital signature.
- Profiling** Advanced optimization of applications that uses data-specific information to further improve program performance. A profiled program is reorganized according to how frequently its control-flow paths are run when operating on data that is considered typical of the data it will process when in production.

3.1.2 Required PTFs

The ANZOBJCVN command is distributed as PTFs for V5R3 or V5R4. For this function, you must order an MRI PTF. By ordering the appropriate MRI PTF for your language, the requisite code PTFs are also ordered. For example, for English (2924), order MRI PTF SI28425 for V5R3 or MRI PTF SI28415 for V5R4. For the complete list of the MRI PTFs for V5R3 and V5R4, refer to Appendix A, “MRI PTFs” on page 61.

Superseded PTFs: This document lists PTF numbers that were available at the time of writing. We recommend that you check for more recent PTFs on the following PTF Cover Letters Search Web site:

[http://www-912.ibm.com/a_dir/as4ptf.nsf/\\$\\$search?openform](http://www-912.ibm.com/a_dir/as4ptf.nsf/$$search?openform)

On the PTF Cover Letters Search Web page, perform the following steps:

1. Enter the PTF number in the Search field and click **Search**.
2. On the PTF Cover Letters Search results page, click the PTF number.
3. On the following PTF details page, if a more recent PTF is available, it will be listed in the header information in the Superseded by row.

Table 3-1 shows the code PTFs for the ANZOBJCVN command. Check whether you have the latest code PTF, and if not, order the code PTF separately.

Table 3-1 Code PTFs for ANZOBJCVN command on i5/OS

i5/OS release	Code PTFs
V5R3M0	SI35094
V5R3M5	SI35094
V5R4M0	SI37535, SI37648
V5R4M5	SI37535, SI37648

Table 3-2 lists the PTFs that are required for restoring objects that were created on 6.1 or later to an earlier i5/OS release.

Table 3-2 Required PTFs for restoring objects to an earlier i5/OS release

i5/OS release	Required PTF
V5R3M0	MF41354
V5R3M5	MF41734
V5R4M0	MF40520
V5R4M5	MF42655

Table 3-3 lists additional PTFs that are required to restore Java programs that were created on IBM i 6.1 or later to an earlier i5/OS release.

Table 3-3 Required PTFs for restoring Java programs to an earlier i5/OS release

i5/OS release	Required PTF
V5R3M0	SI27764
V5R3M5	SI27764
V5R4M0	SI27765
V5R4M5	SI27765

3.1.3 Command usage notes

Consider the following notes about the usage of the ANZOBJCVN command:

- ▶ You must have all object (*ALLOBJ) special authority to run this command.
- ▶ This command, especially the *COLLECT stage, can be long running. For this reason, we suggest that you run this command in a batch job.
- ▶ Vary on any primary or secondary independent ASPs (ASPs 33-255) referenced by this command. Ensure that they have a status of *Available* before you run this command.
- ▶ Mount any user-defined file systems referenced by this command before running this command.
- ▶ This command is intended to be run in one of two ways:
 - Having only one active job with ANZOBJCVN working with libraries and directories at the same time
 - Having two active jobs running ANZOBJCVN at the same time, one job running ANZOBJCVN only for libraries and the other running ANZOBJCVN only for directories
- ▶ If the QUSRSYS library is deleted after information is collected by using ANZOBJCVN, the information will be lost because the files that contain the collected data reside in the QUSRSYS library. See 3.2.3, “Files used for data collections” on page 22, for details about the files that are used by the ANZOBJCVN command.

Attention: Ensure that the QUSRSYS library is not ahead of the QSYS library in the library list when the ANZOBJCVN command is run. Otherwise the ANZOBJCVN command will fail.

3.1.4 Conversion time estimates

The estimated times for the conversion of program model objects might not be close approximations, depending on several factors. Two important factors are the hardware capabilities that are available to the partition on which the conversion occurs and the conversion methods that you choose to use.

By default, the estimates of conversion times are based in part on the processing capacity of the partition on which the ANZOBJCVN command is run. The first-call conversion method might only use a single processor at a time. Therefore, for partitions with at least one full processor, the estimates are based on the CPW rating of a single processor. This means that if you use a different conversion method after you upgrade (for example, the STROBJCVN command) on a partition with more than one processor, conversion times are likely to be shorter than the estimates. For partitions with less than one full processor, the estimate takes into account the fraction of the processor that is currently available to the partition.

Conversion time estimates for a different system

By default, the ANZOBJCVN command bases its conversion time estimates on the system model on which it is being run. The process to estimate the conversion times for a different system model is to create a data area called QIZAFTCD in the QUSRSYS library. This data area must contain the processor feature code of the target system model. You can also supply the number of processors used, in units of .01 processors.

To create the data area, use the Create Data Area (CRTDTAARA) CL command, for example:

```
CRTDTAARA DTAARA(QUSRSYS/QIZAFTCD) TYPE(*CHAR) LEN(10) VALUE('7456 0050')
```

In this example, processor feature code 7456, with one half (.5) of one processor, is used as the base for conversion time estimation.

To display the processor feature code of a system:

1. Type the following Display Hardware Resource (DSPHDWRSC) command and press Enter:

```
DSPHDWRSC TYPE(*PRC)
```

- On the Display Processor Resources display (Figure 3-1), type option 7 (Display resource detail) next to the Main Card Enclosure resource and press Enter.

Display Processor Resources				
Type options, press Enter.				
7=Display resource detail				
Opt	Resource	Type-model	Status	Text
7	CEC01	9406-520	Operational	Main Card Enclosure
	PN01	28E5	Operational	System Control Panel
	MP01	5229	Operational	System Processor Card
	MP02	5229	Operational	System Processor Card
	PV01	7456	Operational	Processor Capacity Card
	SP01	28D7	Operational	Service Processor Card
	BCC01		Operational	Bus Adapter
	BCC02	5229	Operational	Bus Adapter
	BCC03	28E7	Operational	Bus Adapter
	BCC04	28E7	Operational	Bus Adapter
	BCC05	5229	Operational	Bus Adapter
	MS01	30D5	Operational	2048MB Main Storage Car
				More...
F3=Exit F5=Refresh F6=Print F12=Cancel				

Figure 3-1 Displaying the processor resources

- On the Display Resource Detail display (Figure 3-2), you see the processor feature. In this example, the processor feature code is 7456.

Display Resource Detail	
Resource name	: CEC01
Text	: Main Card Enclosure
Type-model	: 9406-520
Serial number	: 10-8A36C
Part number	:
Processor feature code	: 7456
Processor feature	: 8955
Location: U9406.520.108A36C	
Bottom	
Press Enter to continue.	
F3=Exit F5=Refresh F6=Print F12=Cancel	

Figure 3-2 Displaying the resource details

If the QIZAFTCD data area is not created or contains incorrect data, the ANZOBJCVN command defaults to the system processor on which the command is being run for the time conversion estimations.

If the system on which the ANZOBJCVN command is being run cannot be upgraded to 6.1, and the data area does not exist or contains incorrect data, the default value for the processor

type shown in Table 3-4 is used to estimate conversion times. One full processor is assumed for this default value.

Table 3-4 Default feature code for the QIZAFTCD data area

Type	Model	Feature Code
520	0904	7454

Important: If the QIZAFTCD data area is created or its values are changed, any previous collection must be redone in order to reflect the new time estimation.

Conversion time estimates using multiple threads

By default, the ANZOBJCVN command bases its estimates on single-threaded conversion rates, which might occur if you rely on first-call conversion. If you intend to use a faster conversion method, such as STROBJCVN, you can ask ANZOBJCVN to adjust its overall estimates based on the number of threads that might concurrently perform conversions. To do this, determine the current processing capacity of the system by typing the Work with System Activity (WRKSYSACT) command on a command line and press Enter. On the Work with System Activity display (Figure 3-3), you can see the current processing capacity of the system.

```

Work with System Activity
                                04/20/09 16:35:06
Automatic refresh in seconds . . . . . 5
Elapsed time . . . . . : 00:00:02   Average CPU util . . . . . : .0
Number of CPUs . . . . . : 2       Maximum CPU util . . . . . : .1
Overall DB CPU util . . . . . : .0   Minimum CPU util . . . . . : .0
                                      Current processing capacity: 1.00

Authorization Type . . . :
Type options, press Enter.
  1=Monitor job  5=Work with job

Job or
Opt Task      User      Number  Thread  Pty   CPU   Total Sync  Total Async  DB
      QPADEV0006 DALANDON 013031 00000023 1     .1    10    0     0     .0

Bottom
F3=Exit  F10=Update list  F11=View 2  F12=Cancel  F19=Automatic refresh

```

Figure 3-3 Display the current processing capacity of the system

Processing capacity is expressed in units of .01 processors, so one full processor is 100. Multiply the number of threads by the processing capacity. For example, using a processing capacity of 0.50 with two threads per processor during a conversion, you multiply 2 times 50 (2 x 50 = 100). Then create a data area by using the Create Data Area (CRTDTAARA) CL command, called QIZAFTCD:

```
CRTDTAARA DTAARA(QUSRSYS/QIZAFTCD) TYPE(*CHAR) LEN(10) VALUE('7456 0100')
```

Important: If the QIZAFTCD data area is created or its values are changed, any previous collection must be redone in order to reflect the new time estimation.

For another example, using a processing capacity of 3.00 with two threads per processor, you multiply 2 times 300 (2 x 300 = 0600). This corresponds to a total of six threads during STROBJCVN.

A value of two threads per processor has been found to make good use of the system during STROBJCVN. Values higher than two threads per processor might not scale linearly.

How ANZOBJCVN adjusts estimates for processors and threads

The value (processors x threads) is used during ANZOBJCVN OPTION(*COLLECT) as a simple dividing factor for the estimated conversion time for each program. This adjustment to the estimate is valid when considering the total conversion time of a large number of programs. However, converting a single program at a time will not benefit from multithreading, so conversion of a single program will have a higher conversion time (the single-threaded time) than what is shown by ANZOBJCVN.

Because changes to data area QIZAFCD require running the ANZOBJCVN collection again, you might prefer to use the value 0100 for (processors x threads) when collecting data, then use SQL queries to try various values of processors and the multithreading level. See 3.5, “Querying analyzed data” on page 36, for examples.

3.1.5 Alternate method for estimating conversion times

Conversion times: Conversion times reported with the ANZOBJCVN command are estimates. Actual conversion times might vary from the estimate. Conversion time is affected by various system factors, by program attributes, and by the conversion method used. Pertinent system factors include CPU speed, priority of the job, disk arms attached, memory available, and the number of processors. Important program attributes include program size and whether or not the program is optimized.

By default, estimates assume the slowest conversion method, first-call conversion. Use of the Start Object Conversion (STROBJCVN) command will likely require less conversion time than the ANZOBJCVN estimate, especially for partitions with multiple processors that also have many small, unoptimized programs. While conversion estimates are often fairly accurate, some situations can give estimates that are several times longer than the actual time needed for conversion.

As noted previously, the conversion time estimates from ANZOBJCVN are only approximations and depend on a variety of factors. If you are concerned about the estimated conversion times, an alternate method is available to measure estimated conversion times. You might want to use this alternate method if the ANZOBJCVN estimates exceed the time period that you have available for converting programs, and you plan to convert programs either during restore or with the Start Object Conversion (STROBJCVN) command.

Another approximation of 6.1 or later conversion times can be obtained by measuring the forced conversion of programs while restoring them on the release in use before the upgrade. This type of conversion does not prepare the programs to run on 6.1 or later release, but it takes a similar amount of time as the conversion for a given set of programs. Conversion can be forced during a restore by using either of the following methods:

- ▶ Specifying the Force Object Conversion (FRCOBJCVN(*YES *ALL)) parameter on the restore command
- ▶ Before the running the restore command, changing the Force Conversion On Restore (QFRCCVNRST) system value to a value of 4, 5, 6, or 7, as appropriate for your situation

When restoring from a save file, four threads are used, allowing four objects to be converted concurrently. This conversion time will be similar to STROBJCVN using four threads.

3.2 Collecting information

The ANZOBJCVN command has two functions, to collect and to report. Initial usage of the command is to collect information and then the report parameter can be used to generate a spooled file listing of information that was previously collected. In this section, we explain how to collect data from the system.

Conversion times: Conversion times reported with the ANZOBJCVN command are estimates. Actual conversion times might vary from the estimate. Conversion time is affected by various system factors, by program attributes, and by the conversion method used. Pertinent system factors include CPU speed, priority of the job, disk arms attached, memory available, and number of processors. Important program attributes include program size and whether or not the program is optimized.

By default, estimates assume the slowest conversion method, first-call conversion. Use of the Start Object Conversion (STROBJCVN) command will likely require less conversion time than the ANZOBJCVN estimate, especially for partitions with multiple processors that also have many small, unoptimized programs. While conversion estimates are often fairly accurate, some situations can give estimates that are several times longer than the actual time needed for conversion.

To start the collection process:

1. From an i5/OS command line, type ANZOBJCVN and press F4 to prompt the command.
2. On the Analyze Object Conversion (ANZOBJCVN) display (Figure 3-4), type *collect in the Option field and press Enter.

```

                                Analyze Object Conversion (ANZOBJCVN)
Type choices, press Enter.
Option . . . . . *collect      *COLLECT, *REPORT
                                                    Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
```

Figure 3-4 Prompting the Analyze Object Conversion (ANZOBJCVN) CL command

3. You are presented with the options for collecting information as shown in Figure 3-5.

```

                                Analyze Object Conversion (ANZOBJCVN)

Type choices, press Enter.

Option . . . . . > *COLLECT      *COLLECT, *REPORT
Libraries to analyze . . . . . *ALLUSR      Name, generic*, *ALLUSR...
ASP device . . . . . *              Name, *, *ALLAVL...
Check spooled files . . . . . *YES        *YES, *NO
Object . . . . . *NONE

Directory subtree . . . . . *ALL          *ALL, *NONE
Target release . . . . . V6R1M0          VxRyMz

                                                                Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display

```

Figure 3-5 The ANZOBJCVN *COLLECT parameters

3.2.1 Parameters of the ANZOBJCVN *COLLECT command

In this section, we discuss the various parameters for collecting information as shown in Figure 3-5:

► Libraries to analyze (LIB)

Collections are run against libraries on your system. Program model objects within the libraries are the subjects for the reports. You can choose to collect information from a specific library, in which case you specify the library name, or from a generic list of libraries by using the usual wild card convention. Additionally, it is possible to collect data against all libraries that are part of the *ALLUSR element of the system.

If you use the *ALLUSR parameter, all user libraries are analyzed. All libraries with names that do not begin with the letter Q are analyzed except for the following libraries:

- #CGULIB
- #COBLIB
- #DFULIB
- #DSULIB
- #RPGLIB
- #SDALIB
- #SEULIB

Although the following Qxxx libraries are provided by IBM, they typically contain user data that changes frequently. Therefore, these libraries are considered user libraries and are also analyzed:

- QGPL
- QGPL38
- QMGTC
- QMGTC2
- QMPGDATA
- QMQMDATA
- QMQMPROC
- QPFRDATA
- QRCL
- QSRVAGT

- QS36F
- QUSER38
- QUSRADSM
- QUSRBRM
- QUSRDIRCL
- QUSRDIRDB
- QUSRINFSKR
- QUSRNOTES
- QUSROND
- QUSRPOSGS
- QUSRPOSSA
- QUSRPYMSVR
- QUSRRDARS
- QUSRSYS
- QUSRVI

QSYS2* and SYSIB*: Libraries QSYS2* and SYSIB* are usually included in the *ALLUSR list of libraries, so that their contents are included in *ALLUSR save operations. However, because the programs in these libraries are replaced during an upgrade, the ANZOBJCVN command does not check their contents when *ALLUSR is specified.

IBM product libraries: You might also want to collect conversion information about individual IBM product libraries. Libraries for IBM skip-ship products, which do not ship a new version for 6.1, are listed in Appendix C, “IBM software skip-ship products requiring conversion” on page 73.

It is also possible to specify *NONE for this parameter. In this case, no library is analyzed. You can specify LIB(*NONE) if you only want to analyze objects that are located in integrated file system directories.

► **ASP device (ASPDEV)**

This parameter specifies the ASP device where storage is allocated for the library and spooled files to be analyzed. If the library is in an ASP that is not part of the thread’s library name space, this parameter must be specified to ensure that the correct library is analyzed.

Ignored for the Object parameter: This parameter does not apply for the objects that are specified in the Object (OBJ) parameter (described later), because the independent ASP (IASP) name is part of the path name of the object.

- * (asterisk)

ASPs that are currently part of the thread’s library name space are searched to find the library. This includes the system ASP (ASP 1), all defined basic user ASPs (ASPs 2-32), and, if the thread has an ASP group, the primary and secondary ASPs in the thread’s ASP group.

- *ALLAVL

All available ASPs are searched. This includes the system ASP (ASP 1), all defined basic user ASPs (ASPs 2-32), and all available primary and secondary ASPs (ASPs 33-255 with a status of *Available*).

- *CURASPGRP

If the thread has an ASP group, the primary and secondary ASPs in the thread's ASP group are searched to find the library. The system ASP (ASP 1) and defined basic user ASPs (ASPs 2-32) are not searched. If no ASP group is associated with the thread, an error message is issued.

- *SYSBAS

The system ASP (ASP 1) and all defined basic user ASPs (ASPs 2-32) are searched to find the library. No primary or secondary ASPs are searched, even if the thread has an ASP group.

- Name

By using the Name option, you can specify the name of the primary or secondary ASP device to be searched. The primary or secondary ASP must have been activated (by varying on the ASP device) and have a status of *Available*. The system ASP (ASP 1) and defined basic user ASPs (ASPs 2-32) are not searched.

- ▶ Check spooled files (SPLFILE)

This parameter specifies whether to estimate the time needed to convert the spooled files on each ASP. You cannot specify LIB(*NONE) if SPLFILE(*YES) is specified. The following two options are possible:

- *YES: Spooled files on each ASP are analyzed.
- *NO: Spooled files on each ASP are not analyzed.

Note: The ASP device selected (ASPDEV) affects the collection of spooled file data.

For more information about spooled files, see 3.7.3, “Spooled files” on page 43.

- ▶ Object (OBJ)

This parameter specifies the path name, or a pattern to match the name, of the integrated file system objects to be analyzed. The object path name can be either a simple name or a name that is qualified with the name of the directory in which the object is located. A pattern can be specified in the last part of the path name. An asterisk (*) matches any number of characters, and a question mark (?) matches a single character. If the path name is qualified or contains a pattern, it must be enclosed in apostrophes.

Pattern matching: Pattern matching from the Object (OBJ) parameter applies only to the first level objects. If the first level object is a directory, pattern matching does not apply to its contents or the contents of its subdirectories.

Symbolic link: If the last component in the path name is a symbolic link, the symbolic link object is analyzed, but the object pointed to by the symbolic link is not analyzed.

Specifying *NONE means that no integrated file system object is analyzed.

- ▶ Directory subtree (SUBTREE)

This parameter specifies whether to analyze the objects within the subtree if the object specified by the Object (OBJ) parameter is a directory. If the parameter *ALL is used, the objects specified by the Object (OBJ) parameter are analyzed, as appropriate. If the object is a directory, its contents, as well as the contents of all of its subdirectories, are analyzed.

Note: This parameter only applies to objects in the integrated file system. It is only valid when the OBJ parameter is specified.

After the command has begun processing a specific directory subtree, the objects that are found and processed might be affected by operations that update the organization of objects within the specified directory tree. This includes the following actions among others:

- Adding, removing, or renaming object links
- Mounting or unmounting file systems
- Updating the effective root directory for the process calling the command
- Updating the contents of a symbolic link

In order to process the directory subtree, the system code can increase the job-scoped maximum number of file descriptors that can be opened during processing. This is done so that the command is not likely to fail due to a lack of descriptors. This job-scoped maximum value is not reset when the command completes.

*NONE can be specified, in which case only the objects specified by the Object (OBJ) parameter are analyzed. If the object is a directory, the directory is analyzed, but its contents are not analyzed.

► Target release (TGTRLS)

This parameter specifies the release to which the data and programs will be migrated or upgraded. This information is used to determine the specific analysis to be performed.

3.2.2 Examples of collection commands

We suggest one of two approaches to collect data. One way is to collect all types of information at once, so that all collection files are populated, by using the ANZOBJCVN command in Figure 3-6. This command collects information about all program objects found in libraries in the *ALLUSR portion of the system (not including exceptions listed in 3.2.1, “Parameters of the ANZOBJCVN *COLLECT command” on page 18), spooled files, and all relevant objects in the integrated file system.

```

                                Analyze Object Conversion (ANZOBJCVN)

Type choices, press Enter.

Option . . . . . > *COLLECT      *COLLECT, *REPORT
Libraries to analyze . . . . . *ALLUSR      Name, generic*, *ALLUSR...
ASP device . . . . . *                Name, *, *ALLAVL...
Check spooled files . . . . . *YES        *YES, *NO
Object . . . . . > /

Directory subtree . . . . . *ALL          *ALL, *NONE
Target release . . . . . V6R1M0          VxRyMz

                                                                Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
```

Figure 3-6 ANZOBJCVN collection on *ALLUSR libraries, the integrated file system, and spooled files

Alternatively, you can split the data collection process into two steps. For example, you can collect data on program model objects in *ALLUSR libraries and on spooled files:

```
ANZOBJCVN OPTION(*COLLECT) LIB(*ALLUSR) SPLFILE(*YES)
```

Then follow this collection by doing a directory object collection:

```
ANZOBJCVN OPTION(*COLLECT) LIB(*NONE) SPLFILE(*NO) OBJ('/')
```

Important: When running the previous ANZOBJCVN command on the entire integrated file system, the diagnostic message CPDB0C8 is expected on the /dev/QASP01 folder. This message is expected when specifying OBJ('/') and this path is skipped.

After collecting the data, you can display the results by using one of the ANZOBJCVN report options. See 3.3, “Creating reports” on page 23.

You can also collect data on individual libraries, on subsets of the integrated file system, or on both at once. However, if you choose this approach, be sure to run the reports that you want to review before you continue with data collection on another library or another subset of the file system. Each collection of a given type replaces the data in the file used to store that type of information. Alternatively, you can copy the collection files to another set of files each time you run a type of collection, such as programs in libraries. You can then run queries on each copied file. See 3.2.3, “Files used for data collections” on page 22, and 3.5, “Querying analyzed data” on page 36.

The following example shows how to collect information about programs in one specific library:

```
ANZOBJCVN OPTION(*COLLECT) LIB(myLib) SPLFILE(*NO)
```

The following example shows a collection from a subset of directories in the integrated file system:

```
ANZOBJCVN OPTION(*COLLECT) LIB(*NONE) SPLFILE(*NO) OBJ('/myPart') SUBTREE(*ALL)
```

It is also possible to combine library and directory object collections, in this case, against only one library and one directory:

```
ANZOBJCVN OPTION(*COLLECT) LIB(aLib) SPLFILE(*NO) OBJ('/aPath') SUBTREE(*NONE)
```

You might also want to collect conversion information about individual IBM product libraries. Libraries for IBM skip-ship products, which do not ship a new version for 6.1, are listed in Appendix C, “IBM software skip-ship products requiring conversion” on page 73.

3.2.3 Files used for data collections

The data collected by the ANZOBJCVN command is stored in four different files in the QUSRSYS library:

- ▶ The QAIZACVN file in QSYS is the master template file and is not populated with data from collections. However, QAIZACVN in QUSRSYS is the target file for the collection of data analyzed in the libraries.
- ▶ The QAIZADIR file is the target for directory information when the Object parameter is used.
- ▶ The QAIZAOBJ file is the target for integrated file system object information when the Object parameter is used.
- ▶ The QAIZASPL file is the target for information for the spooled file collections.

Each of these files only contains data from the last collection of that type. Subsequent collections will replace the data that is currently held in these files. Unless you collect at one time all the data about a particular type, such as program model objects, you might want to copy these files to another library to do a later analysis of your collections. Collections that are

moved to other libraries can be queried to generate additional reports as required. For details about creating queries, see 3.5, “Querying analyzed data” on page 36. The ANZOBJCVN OPTION(*REPORT) command only runs against these files in the QUSRSYS library and not against copies of these files in other libraries.

Attention: Ensure that the QUSRSYS library is not ahead of the QSYS library in the library list when the ANZOBJCVN command is run. Otherwise the ANZOBJCVN command will fail.

3.3 Creating reports

The ANZOBJCVN command has two functions:

- ▶ To collect
- ▶ To report

Initial usage of the command is to collect information. Then you can use the report parameter to generate a spooled file listing of information that was previously collected. In this section, we explain how to generate reports with the ANZOBJCVN OPTION(*REPORT) command. When the ANZOBJCVN command is run with OPTION(*REPORT), the tool generates a report in a spooled file called QPIZARPT, based on the data of the type last collected with the ANZOBJCVN OPTION(*COLLECT) command.

The QPIZARPT file is shipped with a limit on the maximum number of records. If you choose to print all available reports, which includes detailed reports, or even if you print an object summary report and have a large file system, the report might exceed that maximum number of records. To avoid having to answer a CPA4072 operator inquiry message, you can change the maximum number of records before printing the report by using the following Change Print File (CHGPRTF) command:

```
CHGPRTF FILE(QSYS/QPIZARPT) MAXRCDS(*NOMAX)
```

Conversion times: Conversion times reported with the ANZOBJCVN command are estimates. Actual conversion times might vary from the estimate. Conversion time is affected by various system factors, by program attributes, and by the conversion method used. Pertinent system factors include CPU speed, priority of the job, disk arms attached, memory available, and number of processors. Important program attributes include program size and whether or not the program is optimized.

By default, estimates assume the slowest conversion method, first-call conversion. Use of the Start Object Conversion (STROBJCVN) command will likely require less conversion time than the ANZOBJCVN estimate, especially for partitions with multiple processors that also have many small, unoptimized programs. While conversion estimates are often fairly accurate, some situations can give estimates that are several times longer than the actual time needed for conversion.

To start the report process:

1. From an i5/OS command line, type the ANZOBJCVN CL command and press F4 to prompt the command.
2. On the Analyze Object Conversion (ANZOBJCVN) display (Figure 3-7), type *report in the Option field and press Enter.

```
                                Analyze Object Conversion (ANZOBJCVN)

Type choices, press Enter.

Option . . . . . *report          *COLLECT, *REPORT

                                                                Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
```

Figure 3-7 Prompting the Analyze Object Conversion (ANZOBJCVN) CL command

You are presented with the options for reporting information as shown in Figure 3-8.

```
                                Analyze Object Conversion (ANZOBJCVN)

Type choices, press Enter.

Option . . . . . > *REPORT          *COLLECT, *REPORT
Type of report . . . . . *ALLAVL     *ALLAVL, *CVNPRB, *LIBSUM...
                    + for more values _____

                                                                Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
```

Figure 3-8 The ANZOBJCVN *REPORT command

Note: No other parameters are used when ANZOBJCVN OPTION(*REPORT) is specified. If F9 is used to recall a previous ANZOBJCVN OPTION(*COLLECT) command, users should be aware that additional parameters might appear to be in use. However, only the parameters shown in Figure 3-8 are actually used.

3.3.1 Parameters of the ANZOBJCVN *REPORT command

The Type of report parameter as shown in Figure 3-8 on page 24 includes the following options for generating reports:

▶ *ALLAVL

All summarized and detailed reports available are generated for all objects in the libraries and directories specified in the most recent previous collection for each type. If the Check spooled files (SPLFILE) parameter was set to *YES for a previous library collection, spooled database members information is generated in the Library Summary report.

▶ *CVNPRB

A detailed report of program objects that will not convert is generated for each library specified in the most recent library collection.

▶ *LIBSUM

A summary report is generated for each library specified in the most recent library collection. If the Check spooled files (SPLFILE) parameter was set to *YES during a previous collection, spooled database member information is also generated in this report.

▶ *LIBDTL

A detailed report is generated for each library specified in the most recent library collection.

▶ *OBJSUM

A summary report is generated for each integrated file system directory specified in the most recent object collection.

▶ *OBJDTL

A detailed report is generated for each integrated file system object that is specified in the most recent object collection.

Tips: Use the following tips when running the ANZOBJCVN OPTION(*REPORT) command:

- ▶ You must complete an appropriate collection before asking for a report. Otherwise message CPF00DC is generated with reason code 3.
- ▶ If a collection has been done, but there is no data to report, for example if there are no programs that will fail to convert when a *CVNPRB report is requested, then message CPI1E10 “No printer output to display” is generated.
- ▶ You can combine parameters when generating reports by typing a plus sign (+) on the command prompt of the Type of Report parameter.

We recommend that you use the *CVNPRB and *LIBSUM parameters first. Those two reports only show program model objects that require attention and summary estimates of program and spooled file conversion times. In general, these two reports should provide all the information that you need to prepare for an upgrade.

In contrast, the detailed reports display information about each object. Therefore, a report that is generated by using the *LIBDTL or *OBJDTL parameter might produce very large spooled file output. Even an *OBJSUM report can be quite large, because it displays information about each subdirectory against which data was collected. While detailed reports provide information, such as the conversion time required for a specific program or any file system file names that might be affected, you might want to run queries instead of printing detailed reports. See 3.5, “Querying analyzed data” on page 36.

3.3.2 Examples of report commands

This section provides several examples of using the ANZOBJCVN OPTION (*REPORT) command.

Example 1

The following command shows usage of the ANZOBJCVN OPTION(*REPORT) command:

```
ANZOBJCVN OPTION(*REPORT) RPTTYPE(*CVNPRB *LIBSUM)
```

In this example the Program Conversion Problems (*CVNPRB) and Library Summary (*LIBSUM) reports are generated. Figure 3-9 on page 27 shows the first two pages of the Program Conversion Problems (*CVNPRB) report generated from this command and Figure 3-10 on page 28 shows the first two pages of the Library Summary (*LIBSUM) report.

Analyze Object Conversion - Program Conversion Problems							9/26/08 13:28:17
5722SS1 V5R4M0 060210 RCHAS10							Page 1
Collection Selection Criteria - ANZOBJCVN OPTION(*COLLECT)							
Library	*ALLUSR						
ASPDEV	*						
SPLFILE	*YES						
Objects	/						
Subtree	*ALL						
Target release	V6R1M0						
Report Selection Criteria - ANZOBJCVN OPTION(*REPORT)							
Report type	*CVNPRB						
Libraries	*ALLUSR						
ASPDEV	*						
SPLFILE	*YES						
Objects	/						
Subtree	*ALL						
Target release	V6R1M0						
Analyze Object Conversion - Program Conversion Problems							9/26/08 13:28:17
5722SS1 V5R4M0 060210 RCHAS10							Page 2
Processor Feature Code : 7456							
Auxiliary Storage Pool (ASP) : 00001 ASP Resource Name : *SYSBAS							
-----Counters-----							
Library	Total Objects	Cannot Convert					
MILLSI	2	1					
Object	Type	System Level	Creation Data	Digitally Signed	Profiled	State	
NOCRTDATA	PGM	V5R4M0	NO	NO	NO	*USER	
Auxiliary Storage Pool (ASP) : 00001 ASP Resource Name : *SYSBAS							
-----Counters-----							
Library	Total Objects	Cannot Convert					
WILKE	37	9					
Object	Type	System Level	Creation Data	Digitally Signed	Profiled	State	
BUZRPB	PGM	V2R3M0	NO	NO	NO	*USER	
CBLPARM	PGM	V2R3M0	NO	NO	NO	*USER	
FADEPRDFU	PGM	V3R6M0	NO	NO	NO	*USER	
FAHISTDFUD	PGM	V3R6M0	NO	NO	NO	*USER	
FAHISTDFUE	PGM	V3R6M0	NO	NO	NO	*USER	
HELLO	PGM	V2R2M0	NO	NO	NO	*USER	
LIBTIM	PGM	V2R2M0	NO	NO	NO	*USER	
X42199SR	PGM	V3R6M0	NO	NO	NO	*USER	
X43199SR	PGM	V3R6M0	NO	NO	NO	*USER	

Figure 3-9 ANZOBJCVN Program Conversion Problems report

Analyze Object Conversion - Library Summary Report						9/26/08 13:28:17
5722SS1 V5R4M0 060210 RCHAS10						Page 3
Collection Selection Criteria - ANZOBJCVN OPTION(*COLLECT)						
Library	*ALLUSR					
ASPDEV	*					
SPLFILE	*YES					
Objects	/					
Subtree	*ALL					
Target release	V6R1M0					
Report Selection Criteria - ANZOBJCVN OPTION(*REPORT)						
Report type	*LIBSUM					
Libraries	*ALLUSR					
ASPDEV	*					
SPLFILE	*YES					
Objects	/					
Subtree	*ALL					
Target release	V6R1M0					
Processor Feature Code						7456
Date collected						09/26/08
Total analyzed objects						6,557
Objects that can be converted with no lost attributes						6,545
Objects that can be converted with some lost attributes						1
Total objects to convert						6,546
Total estimated conversion time for library objects						00:39:41

Analyze Object Conversion - Library Summary Report						9/26/08 13:28:17
5722SS1 V5R4M0 060210 RCHAS10						Page 4
Auxiliary Storage Pool (ASP) : 00001						ASP Resource Name : *SYSBAS
	-----Counters-----			-----Symptoms-----		
Library	Total Objects	Need Conversion	No Lost Attributes	Some Lost Attributes	Conversion Time	
CASESTUDY1	2	2	2	0	00:00:00	
CASETUP	12	12	12	0	00:00:01	
CGIDEV2	2	2	2	0	00:00:07	
COMSIPM	3	3	3	0	00:00:01	
CORPSTAGE	2	2	2	0	00:00:00	
DBITSODATA	4	4	4	0	00:00:02	
DBITSODB02	2	2	2	0	00:00:01	
DBITSODB06	3	3	3	0	00:00:01	
DBITSODB07	1	1	1	0	00:00:00	
DBITSODB08	2	2	2	0	00:00:00	
DBITSODB11	4	4	4	0	00:00:02	
DBITSODB16	5	5	5	0	00:00:08	
DBITSOPGMS	6	6	6	0	00:00:00	
DB2QRY	4	4	4	0	00:00:00	
DB2XML	4	4	4	0	00:00:00	
DDSLIB	2	2	2	0	00:00:00	
DXXSAMPLES	32	32	32	0	00:00:07	
DXXTTOOLS	15	15	15	0	00:00:08	

Figure 3-10 ANZOBJCVN Library Summary report

Example 2

Figure 3-11 shows the first two pages of an Object Summary report (*OBJSUM).

```

Analyze Object Conversion - Object Summary Report          10/31/07 15:52:45
5722SS1 V5R4M0 060210 RCHAS61                               Page 1
Collection Selection Criteria - ANZOBJCVN OPTION(*COLLECT)
Library              *ALLUSR
ASPDEV               *
SPLFILE              *YES
Objects              /
Subtree              *ALL
Target release       V6R1M0
Report Selection Criteria - ANZOBJCVN OPTION(*REPORT)
Report type          *OBJSUM
Libraries            *ALLUSR
ASPDEV               *
SPLFILE              *YES
Objects              /
Subtree              *ALL
Target release       V6R1M0
Processor Feature Code . . . . . : 7457
Date collected . . . . . : 10/31/07
Objects that need conversion . . . . . : 156
Total estimated conversion time for directory objects. . . . . : 00:07:15

Analyze Object Conversion - Object Summary Report          10/31/07 15:52:45
5722SS1 V5R4M0 060210 RCHAS61                               Page 2
Directory : /
---Counter-----  -----Symptoms-----
      Need          Unicode          Conversion
      Conversion    Characters        Time
-----
              0              0      00:00:00
Directory : /QOpenSys
---Counter-----  -----Symptoms-----
      Need          Unicode          Conversion
      Conversion    Characters        Time
-----
              0              0      00:00:00
Directory : /QOpenSys/QIBM
---Counter-----  -----Symptoms-----
      Need          Unicode          Conversion
      Conversion    Characters        Time
-----
              0              0      00:00:00
Directory : /QOpenSys/QIBM/ProdData
---Counter-----  -----Symptoms-----
      Need          Unicode          Conversion
      Conversion    Characters        Time
-----
              0              0      00:00:00

```

Figure 3-11 ANZOBJCVN Object Summary Report

Example 3

It is also possible to generate reports based on a combination of directory and library objects, by entering the following command for example:

```
ANZOBJCVN OPTION(*REPORT) RPTTYPE(*LIBSUM *OBJSUM)
```

3.3.3 Understanding the conversion reports

In this section, we guide you through the conversion reports to help you understand the information that is presented in them. We begin by looking at the report shown in Figure 3-12 on page 31. Each report type contains two sections:

- ▶ The *Collection Selection Criteria* section shows the last data collected with the ANZOBJCVN OPTION(*COLLECT) command that is available to be reported. The Library, ASPDEV, and SPLFILE lines are related to the library collections and are updated every time a library collection is issued. The Objects, Subtree, and Target release lines are related to the objects collection and are updated every time an objects collection is issued.
- ▶ The *Report Selection Criteria* section shows the data requested for the report.

The partial report in Figure 3-12 on page 31 shows the following information:

- ▶ The date on which the report was generated
- ▶ The i5/OS version and system name
- ▶ The last data types collected (For additional information, see 3.2.3, “Files used for data collections” on page 22.)
- ▶ The processor feature code used for estimating conversion times

You also see key information at the bottom of the report in regard to the following items:

- ▶ The number of objects analyzed
- ▶ Object conversion with and without lost attributes
- ▶ The total number of objects that will be converted
- ▶ An estimate of the time to convert the objects to the new format

In this example, conversion of the 57 eligible program objects found in library PFREXP is estimated to take 00:00:13, which is zero hours, zero minutes, and 13 seconds. The conversion times shown on the reports compiled with *LIBDTL list the individual program objects with the conversion times listed in seconds only. We noticed that some rounding up and down occurred in the reports that we generated. This was especially true where the cumulative conversion time for all the program objects in any one library was less than one second. In this case, the report for the library in the *LIBSUM report often showed as being 00:00:00.

Conversion times on the *LIBSUM report are reported in hours, minutes, and seconds, for example, HH:MM:SS. Conversion times on the *LIBDTL report for the individual program objects are reported in seconds and parts of seconds only. For example, in Figure 3-12 on page 31 we see that program object BUPAP1CC has a conversion time of 0.124 seconds.

The total conversion time given by the *LIBSUM report is limited to 100 hours (99:59:59). If your report shows this value, you can use SQL queries to find the actual conversion time. See 3.5, “Querying analyzed data” on page 36, for examples. You might also want to make plans to run STROBJCVN with more processors or threads, and run the ANZOBJCVN collection again with these values specified in data area QIZAFTCD. See “Conversion time estimates using multiple threads” on page 15 for more information.

Analyze Object Conversion - Library Details Report						9/21/07 14:35:05	
5722SS1 V5R4M0 060210 RCHAS10						Page 1	
Collection Selection Criteria - ANZOBJCVN OPTION(*COLLECT)							
Library	PFREXP						
ASPDEV	*						
SPLFILE	*YES						
Objects	/						
Subtree	*ALL						
Target release	V6R1M0						
Report Selection Criteria - ANZOBJCVN OPTION(*REPORT)							
Report type	*LIBDTL						
Libraries	PFREXP						
ASPDEV	*						
SPLFILE	*YES						
Objects	/						
Subtree	*ALL						
Target release	V6R1M0						
Analyze Object Conversion - Library Details Report						9/21/07 14:35:05	
5722SS1 V5R4M0 060210 RCHAS10						Page 2	
Processor Feature Code : 7456							
Auxiliary Storage Pool (ASP) : 00001 ASP Resource Name : *SYSBAS							
-----Counters-----				-----Symptoms-----			
Library	Total Objects	Need Conversion	No Lost Attributes	Some Lost Attributes	Conversion Time		
PFREXP	57	57	57	0	00:00:13		
Object	Type	System Level	Creation Data	Digitally Signed	Profiled	State	Conversion Estimated (Sec)
BUPAP1CC	PGM	V3R6M0	YES	NO	NO	*USER	0.124
BUPAP1CCN	PGM	V3R6M0	YES	NO	NO	*USER	0.124
BUPAP1CJ	PGM	V3R6M0	YES	NO	NO	*USER	0.105
BUPAP1CJE	PGM	V3R6M0	YES	NO	NO	*USER	0.115
BUPAP1IN	PGM	V3R6M0	YES	NO	NO	*USER	0.401
BUPAP1INDQ	PGM	V3R6M0	YES	NO	NO	*USER	0.716
BUPAP1INSQ	PGM	V3R6M0	YES	NO	NO	*USER	0.411
BUPAP1ISEQ	PGM	V3R6M0	YES	NO	NO	*USER	0.391
BUPAP1ISER	PGM	V3R6M0	YES	NO	NO	*USER	0.391
BUPAP1I2	PGM	V3R6M0	YES	NO	NO	*USER	0.458
BUPAP1RCN	PGM	V3R6M0	YES	NO	NO	*USER	0.573
BUPAP1RN	PGM	V4R4M0	YES	NO	NO	*USER	0.649
BUPCRTJRN	PGM	V3R6M0	YES	NO	NO	*USER	0.124

Figure 3-12 ANZOBJCVN Library Details Report for library PFREXP

Figure 3-13 on page 33 shows the information that is found later in the Library Details Report concerning the details of some of the objects found in a library. The following information is collected and reported on for each object:

- ▶ Object name
- ▶ Object type
- ▶ System level, which is the earliest release for which an object can be saved
- ▶ Creation data, which indicates if an object has all the required data to be converted to the new release of the operating system

The value N/A means “not applicable,” and is shown for *SQLPKG objects and certain *PGM and *SRVPGM objects used only for SQL purposes. Those objects do not require program conversion, because they are not run as programs.

- ▶ Digitally signed, which indicates whether an object has a digital signature
The program conversion removes any digital signatures that were previously associated with the object because a digital signature added before a program is converted cannot be valid after conversion.
- ▶ Profiling, which indicates if an object will lose the benefits of profiling
If a program had profile data applied previously but its encapsulated profiling data was removed, that program will lose its profiling benefits during program conversion.
- ▶ State, which shows the current state of a program
Programs with a current state of *SYSTEM or *INHERIT will change to *USER during program conversion.
- ▶ Conversion estimate, which shows the estimated conversion time for the program object

Tip: For details about digital signatures, profiling, and program state, see 3.1.1, “Terminology definitions” on page 10.

Object	Type	System Level	Creation Data	Digitally Signed	Profiled	State	Conversion Estimated (Sec)
BUPAP1CC	PGM	V3R6M0	YES	NO	NO	*USER	0.124
BUPAP1CCN	PGM	V3R6M0	YES	NO	NO	*USER	0.124
BUPAP1CJ	PGM	V3R6M0	YES	NO	NO	*USER	0.105
BUPAP1CJE	PGM	V3R6M0	YES	NO	NO	*USER	0.115
BUPAP1IN	PGM	V3R6M0	YES	NO	NO	*USER	0.401
BUPAP1INDQ	PGM	V3R6M0	YES	NO	NO	*USER	0.716
BUPAP1INSQ	PGM	V3R6M0	YES	NO	NO	*USER	0.411
BUPAP1ISEQ	PGM	V3R6M0	YES	NO	NO	*USER	0.391
BUPAP1ISER	PGM	V3R6M0	YES	NO	NO	*USER	0.391
BUPAP1I2	PGM	V3R6M0	YES	NO	NO	*USER	0.458
BUPAP1RCN	PGM	V3R6M0	YES	NO	NO	*USER	0.573
BUPAP1RN	PGM	V4R4M0	YES	NO	NO	*USER	0.649
BUPCRTJRN	PGM	V3R6M0	YES	NO	NO	*USER	0.124
BUPINZFE	PGM	V4R4M0	YES	NO	NO	*USER	0.220
BUPMUNUNE	PGM	V4R4M0	YES	NO	NO	*USER	0.325
CHAINUP_CW	PGM	V3R6M0	YES	NO	NO	*USER	0.124
CLCSTPEX	PGM	V4R4M0	YES	NO	NO	*USER	0.191
CLCSTPEXB	PGM	V3R6M0	YES	NO	NO	*USER	0.153
CLCSTPEXHI	PGM	V3R6M0	YES	NO	NO	*USER	0.181
CLRPLIBALL	PGM	V4R1M0	YES	NO	NO	*USER	0.258
CLRPLIB20	PGM	V3R6M0	YES	NO	NO	*USER	0.201
CLRPLIB25	PGM	V4R1M0	YES	NO	NO	*USER	0.229

Figure 3-13 Objects in library PFREXP, reported by the ANZOBJCVN Library Details report

3.4 Programs that will not convert

The recommended way to determine whether any program objects cannot be converted is to use the *CVNPRB value on the “Type of report” parameter of the ANZOBJCVN OPTION(*REPORT) command. A detailed report of program objects that will not convert is generated for each library specified in the previous collection. See Figure 3-14 on page 34 for an example of a report that is generated by using the *CVNPRB parameter. This report shows that program object NOCRTDATA from library MILLSI will fail conversion, because the required creation data is missing.

It is also possible to create queries to extract data from the files that are used during collection. See 3.5, “Querying analyzed data” on page 36, for an example of a query that lists programs that cannot be converted.

To check individual programs, you can also use the Display Program (DSPPGM) or Display Service Program (DSPSRVPGM) commands to determine whether sufficient creation data exists. For ILE (service) programs, any value except *NO for “All creation data” indicates that enough creation data is available. For OPM programs, any value except *NONE for “Observable information” indicates that enough creation data is available.

Finally, you can also review the detailed library conversion report that is created with the *LIBDTL parameter to determine which program objects cannot be converted. This report has a value of NO in the Creation Data column for these programs.

Programs without creation data: If the ANZOBJCVN command reports programs without creation data, make sure those programs are still being used. For example, they might no longer be in use because they are part of a previous version of an application. If a program without creation data is still in use, then it is necessary to recompile it. Alternatively, if a program was supplied to you without source code, you must replace it with a newer version that contains creation data.

```

Analyze Object Conversion - Program Conversion Problems 10/26/07 11:43:50
5722SS1 V5R4M0 060210 RCHAS10 Page 1
Collection Selection Criteria - ANZOBJCVN OPTION(*COLLECT)
Library *ALLUSR
ASPDEV *
SPLFILE *NO
Objects /
Subtree *ALL
Target release V6R1M0

Report Selection Criteria - ANZOBJCVN OPTION(*REPORT)
Report type *CVNPRB
Libraries *ALLUSR
ASPDEV *
SPLFILE *NO
Objects /
Subtree *ALL
Target release V6R1M0

Analyze Object Conversion - Program Conversion Problems 10/26/07 11:43:50
5722SS1 V5R4M0 060210 RCHAS10 Page 2
Processor Feature Code . . . . . : 7456
Auxiliary Storage Pool (ASP) . . . . . : 00001 ASP Resource Name : *SYSBAS
-----Counters-----
Library      Total      Cannot
            Objects      Convert
-----
MILLSI                2          1

Object      Type      System  Creation  Digitally  Profiled  State
-----
NOCRTDATA  PGM      V4R5M0  NO        NO         NO        *USER
Auxiliary Storage Pool (ASP) . . . . . : 00001 ASP Resource Name : *SYSBAS

```

Figure 3-14 ANZOBJCVN Program Conversion Problems report

Lost attributes

The reports that you generate for your program objects might also indicate that programs will lose attributes during conversion. See the Some Lost Attributes column in the example shown in Figure 3-15.

The following objects are counted for lost attributes:

- ▶ Objects that will lose their digital signatures if YES is displayed under “Digitally Signed.” This change matters when validating an object. It does not affect the object’s function.
- ▶ Objects that will lose their benefits from profiling if YES is displayed under “Profiled.” This change affects the performance of the object.
- ▶ Objects that will drop their state from either *SYSTEM or *INHERIT state to *USER state. This change can affect the object’s function. Applications must not use these states that are reserved for operating system use.

For more information about these attributes, see 3.1.1, “Terminology definitions” on page 10.

Analyze Object Conversion - Library Details Report		9/21/07 14:48:38
5722SS1 V5R4M0 060210 RCHAS10		Page 1
Collection Selection Criteria - ANZOBJCVN OPTION(*COLLECT)		
Library	MILLSI	
ASPDEV	*	
SPLFILE	*YES	
Objects	/	
Subtree	*ALL	
Target release	V6R1M0	
Report Selection Criteria - ANZOBJCVN OPTION(*REPORT)		
Report type	*LIBDTL	
Libraries	MILLSI	
ASPDEV	*	
SPLFILE	*YES	
Objects	/	
Subtree	*ALL	
Target release	V6R1M0	
Analyze Object Conversion - Library Details Report		9/21/07 14:48:38
5722SS1 V5R4M0 060210 RCHAS10		Page 2
Processor Feature Code : 7456		
Auxiliary Storage Pool (ASP) : 00001 ASP Resource Name : *SYSBAS		
-----Counters----- -----Symptoms-----		
Library	Total Objects	Need Conversion
		No Lost Attributes
		Some Lost Attributes
		Conversion Time
MILLSI	293	291
		236
		55
		00:31:38

Figure 3-15 ANZOBJCVN Library Details Report (*LIBDTL) showing lost attributes

3.5 Querying analyzed data

The data collected by the ANZOBJCVN OPTION (*COLLECT) command is stored in one of four files in the QUSRSYS library. (See 3.2.3, “Files used for data collections” on page 22, for details.) It is important to know that each of these files contains only data from the last collection of that type. That is, subsequent collections replace the data that was collected previously in these files.

These files can be queried to generate additional reports as required. For example, enter the STRSQL command and then type the following query to extract the program objects that do not have creation data:

```
SELECT DIOBLI, DIOBNM FROM QUSRSYS/QAIZACVN WHERE DICRTD='0'
```

In this query, the library name (DIOBLI) and name of the object (DIOBNM) fields are selected if the program creation data area indicator (DICRTD) field is set to zero, which indicates that the program object does not have creation data.

For the field definitions of the files created by the ANZOBJCVN command, see Appendix B, “File field definition tables” on page 65.

You might find the following additional queries to be useful in the analysis of your collected data from the ANZOBJCVN OPTION (*COLLECT) command:

- ▶ Query to get file system objects that have a name with an affected character:

```
SELECT QIZAOBJNAM FROM QUSRSYS/QAIZAOBJ WHERE QIZANAMFLG = 1
```

- ▶ Query to get the file system objects that require Java program conversion:

```
SELECT QIZAOBJNAM FROM QUSRSYS/QAIZAOBJ WHERE QIZACVNFLG = 1
```

- ▶ Query to get the full path name for objects that require Java program conversion or the objects that have a name with an affected Unicode character:

```
SELECT O.QIZAOBJNAM, D.QIZADIRNAM1 FROM QUSRSYS/QAIZAOBJ AS O INNER JOIN  
QUSRSYS/QAIZADIR AS D ON D.QIZADIRIDX = O.QIZADIRIDX WHERE O.QIZACVNFLG = 1 OR  
O.QIZANAMFLG = 1
```

- ▶ Query to get the ASPs of the spooled files that have conversion times of more than one second:

```
SELECT SPASP, SPASPN FROM QUSRSYS/QAIZASPL WHERE SPTMCV > 1000000
```

- ▶ Query to get the total conversion time (in seconds; DIOCTM is in milliseconds, so divide by 1000) for *PGM and *SRVPGM objects:

```
SELECT DECIMAL(SUM(DIOCTM))/1000 FROM QUSRSYS/QAIZACVN WHERE (DIOBTP = 'PGM' OR  
DIOBTP = *SRVPGM*)
```

- ▶ For collections performed with data area QIZAFTCD set to 0100 (processors x threads), query to get the adjusted conversion time (in seconds) for 2.0 processors with two threads per processor (QIZAFTCD equivalent value of 0400) instead:

```
SELECT DECIMAL(SUM(DIOCTM))/1000/4 FROM QUSRSYS/QAIZACVN
```


Note: The previously listed queries will produce unreadable output if the CCSID of the job is set to 65535. To correct this problem, you can change the QCCSID system value to something other than 65535, for example 37. Otherwise you can change the queries as follows:

- ▶ Query to get file system objects that have a name with an affected character:

```
SELECT CHAR(QIZAOBJNAM) FROM QUSRSYS/QAIZAOBJ WHERE QIZANAMFLG = 1
```

- ▶ Query to get the file system objects that require Java program conversion:

```
SELECT CHAR(QIZAOBJNAM) FROM QUSRSYS/QAIZAOBJ WHERE QIZACVNFLG = 1
```

- ▶ Query to get the full path name for objects that require Java program conversion or the objects that have a name with an affected Unicode character:

```
SELECT CHAR(O.QIZAOBJNAM), CHAR(D.QIZADIRNAM1) FROM QUSRSYS/QAIZAOBJ AS O  
INNER JOIN QUSRSYS/QAIZADIR AS D ON D.QIZADIRIDX = O.QIZADIRIDX WHERE  
O.QIZACVNFLG = 1 OR O.QIZANAMFLG = 1
```

- ▶ Query to get the ASPs of the spooled files that have conversion times of more than one second:

```
SELECT CHAR(SPASP,SPASP) FROM QUSRSYS/QAIZASPL WHERE SPTMCV > 1000000
```

3.6 Considerations for application developers

If you develop applications for others, whether those applications are sold to your customers or deployed to others within your company, give careful consideration to program conversions.

3.6.1 Will your application convert

The first consideration is whether your programs or service programs can be converted. If they have sufficient creation data and are not altered in unsupported ways, your programs can be converted. In general, MI programs that target V5R1 or a later release can be converted. The constituent modules of such programs must also target V5R1 or later. Programs that target earlier releases can also be converted, as long as the creation data part of their program observability is still available.

More information: See 3.4, “Programs that will not convert” on page 33, for methods to determine which programs cannot be converted.

As discussed earlier, programs that have had their hardware instruction streams altered in unsupported ways, or that have been altered to run system state, cannot be successfully converted. Such programs must be changed to use only supported interfaces. If no supported interface exists for your situation, you can request a new interface by sending e-mail to Rich Diedrich at richd@us.ibm.com.

If you have programs that will not convert for some reason, then you must decide, based on your client set, whether to recompile one or more versions of your software for your clients who want to move to 6.1 or a later release. You can best determine how to support your clients in light of program conversion. For example, you can recompile your application, so that all of its modules and programs have a target release of V5R1 or later. This newly built application can then be converted on client systems to run on 6.1 or later.

Alternatively, you can choose to ship two versions of your product:

- ▶ One that targets 6.1
- ▶ One that targets an earlier release

The same source can be used for both versions, unless you take advantage of new 6.1 create options or new language features. Your application installation code can choose the correct version to install, based on the release at which the client is running. You might compile on both 6.1 and on a previous release. Alternatively, you can compile your application only on 6.1, save it, and restore it to V5R4 or V5R3, where conversion will occur during the restore. If you then also save this pre-6.1 version, you will have two versions that require no conversion, after only compiling your application once.

Conversion assistance: If you need assistance with converting your application to 6.1 or later, contact IBM ISV Enablement for IBM i. Send your name, title, company name, PartnerWorld number, and specific questions by e-mail to iaic@us.ibm.com. You can also use this e-mail address to inform IBM that you have a version of your application that is ready to run on 6.1 or later.

3.6.2 Other decision criteria

If you have confirmed that your MI programs and service programs can be converted, is the conversion time a factor? Again, the ANZOBJCVN command will provide a rough estimate of the time needed to convert a program on a particular machine model. For example, consider whether the conversion time for your programs would cause client concern during installation. As another example, consider whether you have time-out conditions programmed that could be inadvertently tripped if a program in your application is converted before the call to it is carried out.

There are reasons to consider distributing a new version of your application, even if it can be successfully converted in a reasonable amount of time. For example, if you use application profiling and have removed the profiled data, profiling will not be applied during the conversion process.

Another situation to think about is if you apply a digital signature to your programs and prefer to have it available even after the programs have been loaded on a client's system. Digital signatures are removed during program conversion, because the object is changed. Digital signatures are most frequently used to ensure that an object has not been modified and was provided by the expected source, meaning the signature is used most often before programs are loaded. However, your application might also use the digital signature for another reason.

If you decide to recompile and redistribute a new version of your application, you might also want to take advantage of the new create options. See 2.3.2, "Performance improvements" on page 7, for details. The new create options might significantly improve the performance of your application. However, keep in mind that they are available only for programs that target 6.1 or later.

If you have not taken advantage of application profiling, consider it as another way to improve performance. While application profiling is not new, it is an under used feature that has resulted in overall application performance improvements up to 10%. For more information, see the "Advanced optimizations" chapter in the *System i ILE Concepts Version 6 Release 1*, SC41-5606, manual at the following Web site:

<http://publib.boulder.ibm.com/infocenter/iserics/v6r1m0/topic/books/sc415606.pdf>

3.6.3 Adaptive code generation

Whether you decide to recompile and redistribute your application, consider adaptive code generation (ACG). If your application uses Integrated Language Environment (ILE) programs and is created or converted on a system with POWER6 or later technology, it might run faster on such systems than it otherwise would. (Note: Original Program Model (OPM) programs do not take advantage of ACG.) This depends on how extensively your application ends up using hardware features that are found only on newer processors. However, it must then be converted if it is loaded on a system with an older processor, even if the second system is also running 6.1 or later. Your installation instructions should discuss the difference in load times if that sort of conversion is significant for the programs in your application.

If your application does not improve much in performance when using specific POWER6 or later features, then it might be best to create it so that it uses only features that are available on all supported processors. This approach avoids conversion time if the ILE programs are loaded on pre-POWER6 systems running 6.1 or later. You can either create the ILE programs on a system with an older processor or *opt out* of ACG when creating ILE programs on a POWER6 or later processor. The latter can be done by using either of the following methods:

- ▶ Setting the QIBM_BN_CREATE_WITH_COMMON_CODEGEN environment variable to a value of 2 prior to creating ILE modules
- ▶ Specifying the LICOPT('CodeGenTarget=Common') parameter on the CRT* or CHG* commands that you use when building your application

For more information about controlling the use of ACG in your applications, refer to *System i ILE Concepts Version 6 Release 1*, SC41-5606, available on the following Web site:

<http://publib.boulder.ibm.com/infocenter/iserics/v6r1m0/topic/books/sc415606.pdf>

Note: The methods listed above apply only when *creating* ILE programs. A POWER6 or later system that performs conversions (for example, using RSTOBJ FRCOBJCVN(*YES *ALL)) is free to use those hardware features in the converted ILE program.

In contrast, you can take advantage of ACG to optimize your application when the system on which the application was created or converted was not a 6.1 or later POWER6 or later system. In this case, if creation data is observable, run the following command on a 6.1 or later POWER6 or later system:

```
CHGPGM or CHGSRVPGM with FRCRT(*YES)
```

If program creation data is unobservable, restore (from a save file, for example) after first setting the QFRCCVNRST system value to a value of 7, or specify the FRCOBJCVN(*YES *ALL) parameter on the RESTORE command.

3.6.4 Fixes and testing

Keep in mind that conversion can affect your processes not only for handling your application as a whole, but also for distributing fixes or updates. Consider the ramifications listed previously. Also note that the Update Program (UPDPGM) command does not trigger a program conversion, because it neither activates nor calls the program that is being updated. If you use UPDPGM for distributing fixes or updates, you might want to consider changing the parameter used in your RESTORE command to force conversion. For more information see 4.3, "Conversion of program objects during restore" on page 57.

As a general rule, the testing methods that are used to ensure that applications are ready for 6.1 or later should be the same as for any other release.

3.7 Other object conversions

In addition to MI program model objects, the following objects will also be converted when upgrading to 6.1 or later:

- ▶ Java programs
- ▶ Some integrated file system names
- ▶ Spooled files

3.7.1 Java programs

Java programs do not reside in libraries but in directories. All Java programs that are created and stored in the i5/OS integrated file system must be converted. Almost all Java programs should convert without issue. However, some Java programs associated with class files and jar files might encounter problems if they were created for a very early target release. A simple solution is to use the Delete Java Program (DLTJVAPGM) command on these files and allow the Java program to be recreated automatically at run time.

To estimate conversion times, you can run the ANZOBJCVN command against the integrated file system directories that contain Java programs. For example, we ran the following collect and report commands:

```
ANZOBJCVN OPTION(*COLLECT) LIB(*NONE) SPLFILE(*NO) OBJ('/millsi')
ANZOBJCVN OPTION(*REPORT) RPTTYPE(*OBJSUM)
```

Figure 3-16 shows the first two pages of the Object Summary report (*OBJSUM) that is generated for the /millsi directory.

```

Analyze Object Conversion - Object Summary Report          9/21/07 14:56:54
5722SS1 V5R4M0 060210 RCHAS10                               Page 1

Collection Selection Criteria - ANZOBJCVN OPTION(*COLLECT)
Library           MILLSI
ASPDEV            *
SPLFILE          *YES
Objects          /millsi
Subtree          *ALL
Target release    V6R1M0

Report Selection Criteria - ANZOBJCVN OPTION(*REPORT)
Report type       *OBJSUM
Libraries        MILLSI
ASPDEV            *
SPLFILE          *YES
Objects          /millsi
Subtree          *ALL
Target release    V6R1M0

Processor Feature Code . . . . . : 7456
Date collected . . . . . : 09/21/07
Objects that need conversion . . . . . : 33
Total estimated conversion time for directory objects. . . . . : 00:02:29

Analyze Object Conversion - Object Summary Report          9/21/07 14:56:54
5722SS1 V5R4M0 060210 RCHAS10                               Page 2

Directory : /millsi
---Counter-----Symptoms-----
  Need           Unicode           Conversion
  Conversion     Characters         Time
-----
                4                 0      00:00:00

Directory : /millsi/andrew
---Counter-----Symptoms-----
  Need           Unicode           Conversion
  Conversion     Characters         Time
-----
                0                 0      00:00:00

Directory : /millsi/freya
---Counter-----Symptoms-----
  Need           Unicode           Conversion
  Conversion     Characters         Time
-----
                0                 0      00:00:00

```

Figure 3-16 ANZOBJCVN Object Summary Report (*OBJSUM) for the /millsi directory

Figure 3-17 shows the first two pages of the Object Details report (*OBJDTL) of the /millsi directory. Java programs do not rely on MI creation data for conversion, as is the case for program objects in libraries. Also note that the conversion times for Java program objects are likely to be sub-second in most, if not all, cases.

```

Analyze Object Conversion - Object Details Report          9/21/07 15:01:06
5722SS1 V5R4M0 060210 RCHAS10                               Page 1
Collection Selection Criteria ANZOBJCVN OPTION(*COLLECT)

Library      MILLSI
ASPDEV      *
SPLFILE     *YES
Objects     /millsi
Subtree     *ALL
Target release V6R1M0

Report Selection Criteria - ANZOBJCVN OPTION(*REPORT)
Report type  *OBJDTL
Libraries   MILLSI
ASPDEV     *
SPLFILE    *YES
Objects    /millsi
Subtree    *ALL
Target release V6R1M0

Analyze Object Conversion - Object Details Report          9/21/07 15:01:06
5722SS1 V5R4M0 060210 RCHAS10                               Page 2

Processor Feature Code . . . . . : 7456

Directory : /millsi
----Counter---- -----Symptoms-----
      Need          Unicode          Conversion
      Conversion    Characters        Time
-----
                4                0        00:00:00

Conversion      Object
Estimated
(Sec)
-----
0.040          Hi.class
0.048          AJM.class
0.009          Freya.class
0.040          UPFoRbEer.jar

Unicode Characters
-----
There was no data collected for this section

```

Figure 3-17 ANZOBJCVN Object Details Report (*OBJDTL) for the /millsi directory

3.7.2 Integrated file system names

Searches for names occur any time a name is provided on an integrated file system command or API to identify the object being operated on. Searches for names in file systems that are not case-sensitive are affected by the Unicode characters that are defined and the casing rules in effect for that release. When additional characters are defined, and the casing rules are updated to support the new characters, names that contain characters affected by the change may no longer be found.

The IBM i integrated file system stores all names in Unicode and currently supports Unicode Standard 2.0. In 6.1 or later, an automatic conversion runs to update the integrated file system directories in file systems that are not case-sensitive to support Unicode Standard 4.0. The file systems that are not case-sensitive included in this conversion are “root” (/) and user-defined file systems (in any ASP) created with CASE(*MONO).

The ANZOBJCVN tool identifies names of integrated file system objects in file systems that are not case-sensitive that are affected by the Unicode changes. When an object name is identified as affected, it means that either that object, or another object linked in the same parent directory, can be renamed when the automatic name conversion runs in 6.1 or later. Therefore, multiple objects might be identified as affected when only one object is renamed. There is no guarantee as to which affected object is the one that is renamed.

In a file system that is not case-sensitive, the system does not distinguish between uppercase and lowercase characters during searches for names. Any change in the casing rules affects which characters are considered the same when not distinguishing between uppercase and lowercase. Because of this, during the automatic name conversion, names that were not considered to be the same in Unicode Standard 2.0 might now be considered the same in Unicode Standard 4.0. One of the objects must be renamed. It is not predictable which object will be renamed when this occurs. Many factors can affect which object will be renamed when this situation occurs such as the order of the links in the directory and whether one of the objects is damaged.

What should you do for any objects that the ANZOBJCVN command identifies as being affected by the Unicode conversion? Instead of waiting for the automatic name conversion to occur, you might want to change the names of the affected objects prior to the installation of 6.1 or a later release. By doing so, you can handle any name changes in a more controlled fashion. You can change the object names and run the ANZOBJCVN command again to ensure that any possible name issues have been addressed.

Renamed objects: Even if the ANZOBJCVN command identifies no objects as being affected by Unicode name conversion, the automatic name conversion might still rename some objects, due to changes in the Unicode Standard or if any name changes occur on the system before the upgrade.

3.7.3 Spooled files

By default, conversion of spooled files occurs during an IPL of the system or a vary on of an ASP. Spooled file data is collected by the ANZOBJCVN command to provide information that can be useful when planning system upgrades. The estimated time is the expected time for the system to convert spooled files during the installation IPL for SRC C900 2AAA or for a vary on step C900 29B0 of an ASP group.

After running a collection and then a report, you can find details of the spooled file conversion times on the report as shown in the example in Figure 3-18.

Total spooled files to convert :	697
Total estimated conversion time for spooled files :	00:00:06

Figure 3-18 Spooled file conversion estimate

The number of spooled files in the data collected represents all database members in the QSPLxxxx libraries, where xxxx is blank or 0002-0032 or 0033-0255 for an ASP group.

Using the Reclaim Spool Storage (RCLSPLSTG) DAYS(*NONE) command to reduce the number of database members in the QSPLxxxx libraries in SYSBAS is generally not recommended. Reducing the number of available database members in the QSPLxxxx libraries for an ASP group can cause severe spooled file creation performance problems resulting in the inability of the system or application to reach full capacity for an undetermined period of time. See the *Reclaim Spooled File Storage Experience Report* at the following Web address:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/experience/spoolreclaim.pdf>

Delaying conversion can be accomplished by using a data area. However, be aware that a deferred conversion of spooled files results in the existing spooled files being unavailable until they have been converted. For more information about delaying the conversion of spooled files refer to *Installing, Upgrading, or Deleting i5/OS and Related Software*, SC41-5120-10, at the following Web address:

<http://www.ibm.com/support/docview.wss?uid=pub1sc41512010>

Tip: Reduce the number of spooled files kept on your system prior to upgrading the operating system. Longer conversion times occur if large numbers of spooled files are left on output queues. Also consider removing or saving spooled files as part of regular system maintenance, but this is especially relevant prior to performing a system upgrade.

3.8 Removing unsupported products

Conversion is not required before a program can be deleted. However, if a particular product uses an exit program during the delete operation, and that program calls some other program that cannot be converted, then the Delete Licensed Program (DLTLICPGM) command might not complete successfully on 6.1 or later. This problem can be avoided by deleting the unsupported product before upgrading. However, you might want to save the old product before deleting it, in case you need it later for a release prior to 6.1.



Program conversion methods

In this chapter we discuss the various methods of converting programs to run on 6.1 or a later release.

Important: The conversion methods discussed in this chapter must be performed on a system that has been upgraded or migrated from releases V5R4 or earlier to 6.1 or later. These conversions do not need to be performed when migrating from 6.1 to a later release.

We recommend that you prepare for object conversion prior to starting the upgrade or migration process. You might find that some program objects will not convert and will, therefore, be unusable on 6.1 or later. See Chapter 3, “Preparing for conversions” on page 9, for details about how to use the Analyze Object Conversion (ANZOBJCVN) command. This command helps you to determine what actions, if any, are required prior to starting an upgrade or migration.

The ANZOBJCVN command is not available on 6.1 or later. Users who must check which objects require conversion on these systems should use the Start Object Conversion (STROBJCVN) command with the *CHECK option. See 4.2.2, “Parameters of the STROBJCVN command” on page 50, for more details.

4.1 Overview of program conversion

At this stage we expect that you have identified any program objects that will not convert and have made plans to recompile or acquire versions of these objects compatible with your planned upgrade release of 6.1 or later.

During an upgrade to 6.1, many IBM-supplied program objects will be provided in a 6.1-compatible format on your installation media. For those program objects, no conversion is required. However, for some IBM-supplied programs (for example, those that *skip ship* for 6.1) conversion is required.

After a system is upgraded to 6.1 or later, program conversion can be performed. The list of objects, if any, that require conversion consists of two parts:

- ▶ Those objects identified by the ANZOBJCVN command by the process described in Chapter 3, “Preparing for conversions” on page 9.
This includes program objects in user libraries and Java programs attached to files in directories.
- ▶ IBM licensed program products (LPP) that are identified as being skip ship for 6.1.
These are products for which there is not a specific 6.1 version. See Appendix C, “IBM software skip-ship products requiring conversion” on page 73, for a list of these products and the libraries where the products are installed.

Conversion to a 6.1-compatible format is a one-time only operation. For program model objects that require conversion, and depending on the method of conversion selected, the process may render applications unavailable while conversion takes place.

Change dates and object sizes: For program model objects that are converted to a 6.1-compatible format, the change date and time is altered to the time of conversion and the object size might change. These changes can have audit implications.

If you use the Save Changed Objects (SAVCHGOBJ) command, note that conversion affects which objects are saved.

The size changes for program model objects are expected to be small. A single-digit percentage increase is the most likely result over all such objects on a given system. Further, the amount of storage consumed by program model objects is almost always a small part of overall storage usage. Therefore, any storage increases resulting from program conversion should be insignificant. Smaller and very old programs tend to increase by a larger percentage, but the amount of storage they consume is still quite small.

The following methods are available to perform program object conversion:

- ▶ Using the 6.1 STROBJCVN CL command to manually initiate conversion on program objects within libraries
For details see 4.2, “Conversion using STROBJCVN” on page 47.
- ▶ During a restore
For details see 4.3, “Conversion of program objects during restore” on page 57.
- ▶ The first time the program is called
For details see 4.4, “Conversion the first time a program is run or called” on page 59.

Recommendation for using the STROBJCVN command: For objects already on the system, use the STROBJCVN command rather than waiting for a *first call* conversion. When you initiate the conversion process before calling programs, you have the greatest flexibility for controlling the use of machine resources. You also avoid the potential of affecting the runtime behavior of applications that either rely on their own timing constraints or are first called by many users concurrently. If a program requires conversion and is called in multiple jobs at the same time, the first job will perform the conversion while the other jobs wait.

Required PTFs

Before starting conversions of any kind on a 6.1 or later system, make sure that you have applied the following PTFs for your specific release:

- ▶ 6.1.0
 - SI36087
 - SI37071
 - MF48992
 - MF46307
 - MF46307
 - MF46636
 - MF47450
 - MF48880
- ▶ 6.1.1
 - SI36087
 - SI37071
 - MF49004
 - MF47464
 - MF48202
- ▶ 7.1.0
 - SI36960
 - MF48889

We also recommend checking for superseding versions of these PTFs (see the “Superseded PTFs” Note box in 3.1.2, “Required PTFs” on page 11).

4.2 Conversion using STROBJCVN

The 6.1 STROBJCVN command is provided to manually initiate object conversion. Objects are converted from the format used in earlier levels of the operating system to the format required for use on IBM i 6.1.

Important: Although the STROBJCVN command exists on earlier releases of i5/OS, only the IBM i 6.1 and later versions of the command support conversion to a 6.1-compatible format.

With the STROBJCVN command, you have the option to check whether certain objects require conversion. However, running this check option on 6.1 is not a good substitute for running the ANZOBJCVN command on V5R3 or V5R4. It is to your advantage to plan ahead before the upgrade, so you can replace any applications that will not convert and get an estimate of how long conversion will take for the rest of your programs.

The STROBJCVN command does not convert the directories affected by new Unicode characters and casing rules. That conversion is automatically done by the operating system. See 3.7.2, “Integrated file system names” on page 43, for details.

The STROBJCVN command also does not convert spooled files. Conversion is automatically done by the operating system. See 3.7.3, “Spooled files” on page 43, for details.

The STROBJCVN command can check or convert other types of objects that are not included in an ANZOBJCVN analysis, such as files in libraries. See 3.1, “Analyzing objects” on page 10, for a list of object types analyzed by ANZOBJCVN.

4.2.1 STROBJCVN usage notes

The following notes are about usage of the STROBJCVN command:

- ▶ You must have all object (*ALLOBJ) special authority to run this command.
- ▶ Any primary or secondary independent auxiliary storage pool (ASP; ASPs 33-255) referenced by this command must be varied on and have a status of *Available* before running this command.
- ▶ Any user-defined file systems referenced by this command must be mounted before running this command.
- ▶ Objects located in a *read-only* user-defined file system are not converted.

Important: This command with option *CONVERT can be long running. For all systems, but particularly for those with limited interactive (5250 online transaction processing (OLTP)) capability, we recommend that you run this command in a batch job.

It is also recommended that you change the Maximum spooled files system value (QMAXSPLF) to the maximum value before using this command on *ALLUSR. SQL conversions need to create and delete spooled files on every object converted that contains SQL, this can reach the 9999 count on systems with many SQL objects.

Multi-threading

The STROBJCVN command might use multiple threads to convert programs, service programs, and modules. Each thread converts one object at a time, and multiple threads run concurrently. By default, the number of threads is calculated based on the number of available processors. Currently two threads are used for each available full processor. A data area can be used to override the default thread usage, thus specifying that either more or fewer threads are to be used concurrently.

The following command can be used to create the data area prior to running STROBJCVN:

```
CRTDTAARA DTAARA(QGPL/QIZANBRTHD) TYPE(*DEC) LEN(10 0) VALUE(x) TEXT('Number of threads for STROBJCVN')
```

If the data area is used, it applies to all instances of the STROBJCVN command. There is no per-job control mechanism.

For most systems, the default value of two threads per processor is appropriate. The option to change the number of threads is made available for users who want to tailor the behavior of the STROBJCVN command to their system needs.

Conversion of an object occurs within one thread. Using multiple threads will be of benefit when converting a large number of objects, not when converting a single large object.

Tip: The Work with System Activity (WRKSYSACT) command, which is part of the Performance Tools (5761-PT1) product, provides an easy way to monitor active threads.

If you would like to have more dynamic control over the amount of processing capacity used by conversion processing, you can do the following:

1. Set data area QGPL/QIZANBRTHD to specify a single thread for each STROBJCVN command.
2. Issue separate batch jobs to run STROBJCVN for each library individually, starting with those most important to you.
3. Control the number of active jobs within the batch subsystem to achieve the desired level of processor activity.

Limits to monitoring progress

STROBJCVN works in three phases when processing objects in libraries:

1. Obtain a list of all objects to be processed, grouped by library.
2. Perform multi-threaded conversions for all programs in the list. No completion messages are sent by this phase.
3. Once all program conversions are complete, do two things for each object in the list:
 - Send completion messages for programs converted by phase 2.
 - Convert other objects (files, SQL objects).

This permits completion messages to be grouped by library.

Because completion messages are not sent until phase 3, you cannot rely on the messages in the job log to determine the progress of the conversion. You also cannot determine conversion time for an object (or group of objects in a library) based on the timestamp on the completion messages.

Converting for different processors

Beginning with 6.1, the system can use Adaptive Code Generation (ACG) when creating or converting a program. ACG allows the program to use features that are unique to a specific processor. When such a program is run on a processor that does not implement these features, the system will automatically perform conversion to create a compatible program. See 3.6.3, “Adaptive code generation” on page 39, for more information.

It is possible for your computing environment to use a variety of processor hardware. If a program is converted on a newer processor and is then run on an older processor, automatic conversions may occur even if both processors are running the same release of IBM i. Some examples are:

- ▶ Programs in an Independent Auxiliary Storage Pool (IASP) are converted on a system with a POWER6 processor. The IASP is then switched to another system with a POWER5™ processor.
- ▶ Programs are converted on a system with a POWER6 processor. The programs are then saved and restored onto another system with a POWER5 processor.
- ▶ Programs are converted on a system with a POWER6 processor. The system then encounters a hardware failure and is temporarily replaced by a system using an older POWER5 processor.

Tip: If you want to prevent the system from spending time on these automatic conversions, you can run the STROBJCVN command using your earliest processor hardware to ensure that programs are compatible with all your processors.

4.2.2 Parameters of the STROBJCVN command

When using the STROBJCVN command, you are prompted to enter either *CHECK or *CONVERT:

- ▶ Specifying *CHECK initiates an inspection of objects in the specified library or directory to determine whether those objects have already been converted or whether they still require conversion. The *CHECK parameter also provides feedback on whether a program object can be converted. For details see 4.2.3, “STROBJCVN OPTION(*CHECK)” on page 52.
- ▶ Specifying *CONVERT initiates the conversion process on all appropriate objects in the specified library or directory. For details see 4.2.4, “STROBJCVN OPTION(*CONVERT)” on page 55.

In addition, the STROBJCVN command includes the following parameters:

- ▶ Library (LIB)

This parameter specifies the library for which objects are to be checked or converted to the format required for use with 6.1 or later:

- *ALLUSR

All user libraries are selected. All libraries with names that do not begin with the letter Q are selected, except for those detailed in 3.2.1, “Parameters of the ANZOBJCVN *COLLECT command” on page 18.

- *NONE

No library is selected. Specify LIB(*NONE) if you want to convert or check objects that are in directories only.

- Library name

Specifies the name of the library whose objects are to be checked or converted.

- ▶ Object type (OBJTYPE)

This parameter specifies which object types in the library should be converted. If *CHECK is specified for the Option (OPTION) parameter, the Object type parameter is ignored. If *NONE is specified for the Library (LIB) parameter, the Object type parameter is ignored.

- *ALL

All program (*PGM) objects, service program (*SRVPGM) objects, module (*MODULE) objects, and database file (*FILE) objects in the specified library are converted. In addition, the stored SQL information in all *PGM, *SRVPGM, and *SQLPKG objects that contain SQL statements, as well as *PGM and *SRVPGM objects that are used to implement external stored procedures, are converted.

- *FILE

Only database file member objects in the specified library are converted. For a brief description of this conversion see 4.2.4, “STROBJCVN OPTION(*CONVERT)” on page 55.

Note: File objects only must be converted if you are upgrading from V5R3 or earlier.

- *ALLPGM

All program (*PGM) and service program (*SRVPGM) objects in the specified library are converted.

Module (*MODULE) objects are not converted. Modules cannot be run, so they do not need to be converted for applications to run. Modules will be converted as needed when they are bound into a program or service program, for example, with the CRTPGM or UPDSRVPGM commands.

Tip: *ALLPGM is recommended when upgrading from V5R4 or V5R4M5 so that unnecessary file conversions are not performed.

- *SQL

The stored SQL information in all *PGM, *SRVPGM, and *SQLPKG objects that contain SQL statements, as well as in *PGM and *SRVPGM objects that are used to implement external stored procedures, are converted.

- ▶ ASP device (ASPDEV)

This parameter specifies the ASP device where storage is allocated for the library objects to be checked or converted. If the library is in an ASP that is not part of the thread's library name space, this parameter must be specified to ensure that the correct library is used.

Object parameter: The ASP device parameter does not apply to the objects specified in the Object (OBJ) parameter (described later) because the independent ASP name is part of the path name of the object.

- * (asterisk)

ASPs that are currently part of the thread's library name space are searched to find the library. This includes the system ASP (ASP 1), all defined basic user ASPs (ASPs 2-32), and, if the thread has an ASP group, the primary and secondary ASPs in the thread's ASP group.

- *ALLAVL

All available ASPs are searched. This includes the system ASP (ASP 1), all defined basic user ASPs (ASPs 2-32), and all available primary and secondary ASPs (ASPs 33-255 with a status of *Available*).

- *CURASPGRP

If the thread has an ASP group, the primary and secondary ASPs in the thread's ASP group are searched to find the library. The system ASP (ASP 1) and defined basic user ASPs (ASPs 2-32) are not searched. If no ASP group is associated with the thread, an error is issued.

- *SYSBAS

The system ASP (ASP 1) and all defined basic user ASPs (ASPs 2-32) are searched to find the library. No primary or secondary ASPs are searched, even if the thread has an ASP group.

- Name

With this option, you can specify the name of the primary or secondary ASP device to be searched. The primary or secondary ASP must have been activated (by varying on the ASP device) and have a status of *Available*. The system ASP (ASP 1) and defined basic user ASPs (ASPs 2-32) are not searched.

► Object (OBJ)

This parameter specifies the path name, or a pattern to match the name, of files with an attached Java program to be checked or converted. The object path name can be either a simple name or a name that is qualified with the name of the directory in which the object is located. A pattern can be specified in the last part of the path name. An asterisk (*) matches any number of characters, and a question mark (?) matches a single character. If the path name is qualified or contains a pattern, it must be enclosed in apostrophes. To check or convert appropriate objects in the root file system, you should specify a forward slash (/).

Specifying *NONE means that no integrated file Java program is checked or converted.

► Directory subtree (SUBTREE)

This parameter determines the subtree upon which either the check or convert function will be run.

Integrated file system objects: This parameter applies to objects in the integrated file system only. It is valid only when the OBJ parameter is specified.

If *ALL is used, the appropriate objects specified by the OBJ parameter are checked or converted. If the object is a directory, its contents as well as the contents of all of its subdirectories are checked or converted.

After the command has begun processing a specific directory subtree, the objects that are found and processed might be affected by operations that update the organization of objects within the specified directory tree. This includes, but is not limited to, the following operations:

- Adding, removing, or renaming object links
- Mounting or unmounting file systems
- Updating the effective root directory for the process calling the command
- Updating the contents of a symbolic link

In order to process the directory subtree, the system code can increase the job-scoped maximum number of file descriptors that can be opened during processing. This is done so that the command is not likely to fail due to a lack of descriptors. This job-scoped maximum value is not reset when the command completes.

If *NONE is used, the appropriate objects specified by the OBJ parameter are checked or converted. If the object is a directory, it is checked or converted as appropriate, but its contents are not checked or converted.

4.2.3 STROBJCVN OPTION(*CHECK)

If the STROBJCVN command is run with the *CHECK option, objects located by the LIB parameter and OBJ parameter are checked to determine whether they need to be converted. The following types of program model objects in libraries are checked by this command:

- Programs (*PGM)
- Service programs (*SRVPGM)
- Modules (*MODULE)

In the integrated file system, stream files with attached Java programs that were created to run on i5/OS are checked by this command. Only stream files in the “root” (/), QOpenSys, and user-defined file systems (UDFS) are included.

Figure 4-1 shows the STROBJCVN OPTION(*CHECK) command parameters. In this example, the command checks all objects of type *PGM, *SRVPGM, or *MODULE that are in *ALLUSR libraries, but not Java programs in the integrated file system. The STROBJCVN OPTION(*CHECK) command reports on the number of objects that have already been converted, the number of objects that still need to be converted, and those objects that cannot be converted.

```

                                Start Object Conversion (STROBJCVN)

Type choices, press Enter.

Option . . . . . > *CHECK          *CONVERT, *CHECK
Library . . . . . > *ALLUSR        Name, generic*, *ALLUSR...
Object type . . . . . *ALL         *ALL, *FILE, *ALLPGM, *SQL
ASP device . . . . . *            Name, *, *ALLAVL...
Object . . . . . *NONE

Directory subtree . . . . . *ALL   *ALL, *NONE

                                                                Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys

```

Figure 4-1 STROBJCVN OPTION(*CHECK) command

To see the results of the *CHECK process, refer to the session job log. Figure 4-2 shows the message detail from the job log for an object that has not been converted.

```

CPIB0EA  Information          10  06/25/07  09:13:19.272366  QIZAECVN
QSYS      0975  QUIMNDRV  QSYS      060F
Message . . . . . : Object LIBHT451 has not been converted.
Cause . . . . . : Start Object Conversion(STROBJCVN) OPTION(*CHECK) found
object LIBHT451 of type *SRVPGM in library CMC1 that has not been converted
yet to the current release of the operating system. Recovery . . . :
To convert the object to the current release of the operating system, run
STROBJCVN OPTION(*CONVERT) LIB(CMC1).

```

Figure 4-2 Example of message in the job log after running STROBJCVN OPTION(*CHECK)

The command should complete with message CPCB0D5, which summarizes the results of the check. Figure 4-3 shows an example of this message.

```
Additional Message Information

Message ID . . . . . : CPCB0D5      Severity . . . . . : 00
Message type . . . . . : Completion  Time sent . . . . . : 08:58:05
Date sent . . . . . : 11/01/07

Message . . . . . : Checking of objects was successful.
Cause . . . . . : Start Object Conversion(STROBJCVN) OPTION(*CHECK) ended
with the following results.
  7 objects have not been converted yet. 159 objects did not need to be
converted. 166 objects were processed in *ALLUSR.
0 objects have not been converted yet. 0 objects did not need to be
converted. 0 objects were processed in *N.
  Objects did not need to be converted due to one of the following reasons:
  - Some objects were already converted.
  - Some objects do not have all the creation data.
  If *N or 0 appears for any fields, then this information does not apply.

                                                    Bottom

Press Enter to continue.

F3=Exit  F6=Print  F9=Display message details
F10=Display messages in job log  F12=Cancel  F21=Select assistance level
```

Figure 4-3 An example of message CPCB0D5

Notice the part of the additional message information that starts with “0 objects have not been converted yet.” That section looks like a duplicate of the library information preceding it but without the values filled in. Instead, it refers to objects in directories, which were not checked with this command invocation because the Object parameter was set to *NONE.

There is no output parameter to generate a spooled file for the messages issued by this command. However, information can be obtained by generating a spooled file of the session job log with the following Display Job Log (DSPJOBLOG) command:

```
DSPJOBLOG OUTPUT(*PRINT)
```

Tip: If you want to see messages directly on the display, enter the following command from a Command Entry display (CALL QCMD):

```
STROBJCVN OPTION(*CHECK)
```

If you want to write your own program to determine which objects require conversion, the conversion required field returned by the following APIs will be helpful:

- ▶ Retrieve Program Information (QCLRPGMI) API
<http://publib.boulder.ibm.com/infocenter/series/v6r1m0/index.jsp?topic=/apis/qclrpgmi.htm>
- ▶ Retrieve Service Program Information (QBNRSPGM) API
<http://publib.boulder.ibm.com/infocenter/series/v6r1m0/index.jsp?topic=/apis/qbnrspgm.htm>
- ▶ Retrieve Module Information (QBNRMODI) API
<http://publib.boulder.ibm.com/infocenter/series/v6r1m0/index.jsp?topic=/apis/qbnrmodi.htm>

4.2.4 STROBJCVN OPTION(*CONVERT)

When the STROBJCVN command is run with the *CONVERT option, it initiates the object conversion process. Program model objects that do not require conversion are not converted. The following types of program model objects in libraries are converted by this command, as needed:

- ▶ Programs (*PGM)
- ▶ Service programs (*SRVPGM)
- ▶ Modules (*MODULE)

SQL data is also converted by this command. Programs, service programs, and SQL package (*SQLPKG) objects can contain SQL statements and data used for externally stored procedures. Converting SQL data contained within a program object is handled separately from converting the program itself.

In the integrated file system, stream files with attached Java programs that were created to run on i5/OS are also converted by this command. Only stream files in the “root” (/), QOpenSys and user-defined file system are included.

Important: STROBJCVN OPTION(*CONVERT) OBJTYPE(*ALL) also processes various types of file objects in libraries. If these files were not converted for V5R4 (for example, if you upgraded to 6.1 from V5R3), then these file objects also undergo a type of conversion.

STROBJCVN OPTION(*CONVERT) OBJTYPE(*ALL) must always take action that prompts file conversion when it is necessary. Even if file conversions have already been done, which is likely if you are upgrading from V5R4, STROBJCVN OPTION(*CONVERT) can run a long time on libraries that have large numbers of files, even though no conversion is being done. It is important to note that file processing time is *not* included in the ANZOBJCVN time estimates.

If you are upgrading from V5R4 we recommend that you run STROBJCVN (*CONVERT) OBJTYPE(*ALLPGM) to convert only programs and service programs.

If you are upgrading from V5R3 and have significant numbers of files in libraries, first run STROBJCVN (*CONVERT) OBJTYPE(*ALLPGM) to convert only programs and service programs, then run STROBJCVN (*CONVERT) OBJTYPE(*ALL) to convert files. (Once programs are converted, they will not be converted again.)

Figure 4-4 shows an example of the STROBJCVN command with the *CONVERT option. In this case, the command initiates conversion for objects in *ALLUSR libraries, but not for Java programs attached to files in the integrated file system. It uses OBJTYPE(*ALLPGM) as recommended for upgrades from V5R4.

```

                                Start Object Conversion (STROBJCVN)

Type choices, press Enter.

Option . . . . . *CONVERT      *CONVERT, *CHECK
Library . . . . . *ALLUSR      Name, *ALLUSR, *NONE
Object type . . . . . *ALLPGM  *ALL, *FILE, *ALLPGM, *SQL
ASP device . . . . . *         Name, *, *ALLAVL...
Object . . . . . *NONE

Directory subtree . . . . . *ALL      *ALL, *NONE

                                                                Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys

```

Figure 4-4 STROBJCVN OPTION(*CONVERT) used to convert program objects in *ALLUSR libraries

Java programs in the integrated file system can be converted by using the Object (OBJ) parameter. The object path name can be either a simple name or a qualified name. A pattern using wildcard characters can be specified in the last part of the name. Specifying the / will address the root file system.

Figure 4-5 shows an example of the STROBJCVN command with the *CONVERT option. In this example, the command initiates conversion for all Java programs that require conversion that are in the root file system.

```

                                Start Object Conversion (STROBJCVN)

Type choices, press Enter.

Option . . . . . *CONVERT      *CONVERT, *CHECK
Library . . . . . > *NONE      Name, *ALLUSR, *NONE
Object type . . . . . *ALL      *ALL, *FILE, *ALLPGM, *SQL
ASP device . . . . . *         Name, *, *ALLAVL...
Object . . . . . > '/'

Directory subtree . . . . . *ALL      *ALL, *NONE

                                                                Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys

```

Figure 4-5 STROBJCVN OPTION(*CONVERT) to convert Java programs in the root file system

STROBJCVN OPTION(*CONVERT) should complete with a CPCB0E2 message that summarizes the activity. Figure 4-6 shows an example of this message.

```
Additional Message Information

Message ID . . . . . : CPCB0E2      Severity . . . . . : 00
Message type . . . . . : Completion
Date sent . . . . . : 11/01/07      Time sent . . . . . : 22:27:44

Message . . . . . : Conversion of objects was successful.
Cause . . . . . : Start Object Conversion(STROBJCVN) OPTION(*CONVERT) ended with the
following results.

2 objects in library QAPTL have been processed. Some objects may not have been converted
because they were converted at an earlier time. They are included in this count. 0 objects
in *N have been processed. Some objects may not have been converted because they were
converted at an earlier time. They are included in this count. If *N or hex zero appears for
any fields, then this information does not apply.

                                                                    Bottom

Press Enter to continue.

F3=Exit  F6=Print  F9=Display message details
F10=Display messages in job log  F12=Cancel  F21=Select assistance level
```

Figure 4-6 Example of message CPCB0E2 that summarizes the STROBJCVN *CONVERT activity

4.3 Conversion of program objects during restore

With the system value Force conversion on restore (QFRCCVNRST), the user can specify whether to convert the following objects during a restore:

- ▶ Program (*PGM)
- ▶ Service program (*SRVPGM)
- ▶ SQL package (*SQLPKG)
- ▶ Module (*MODULE)

The possible values for QFRCCVNRST are 0 to 7. The default value is 1. A value of 2 is sufficient to initiate the conversion of objects requiring conversion to a 6.1-compatible format.

This system value can also prevent some objects from being restored. An object that is specified to be converted by the system value, but cannot be converted because it does not contain sufficient creation data, is not restored.

Reminder: If you change the system value QFRCCVNRST, remember to reset it to the value that you were previously using, after the restore activity.

The *SYSVAL value for the Force object conversion (FRCOBJCVN) parameter on the restore commands (RST, RSTLIB, RSTOBJ, and RSTLICPGM) uses the value of the QFRCCVNRST system value. Figure 4-7 shows the Force object conversion parameter on the Restore Library (RSTLIB) command.

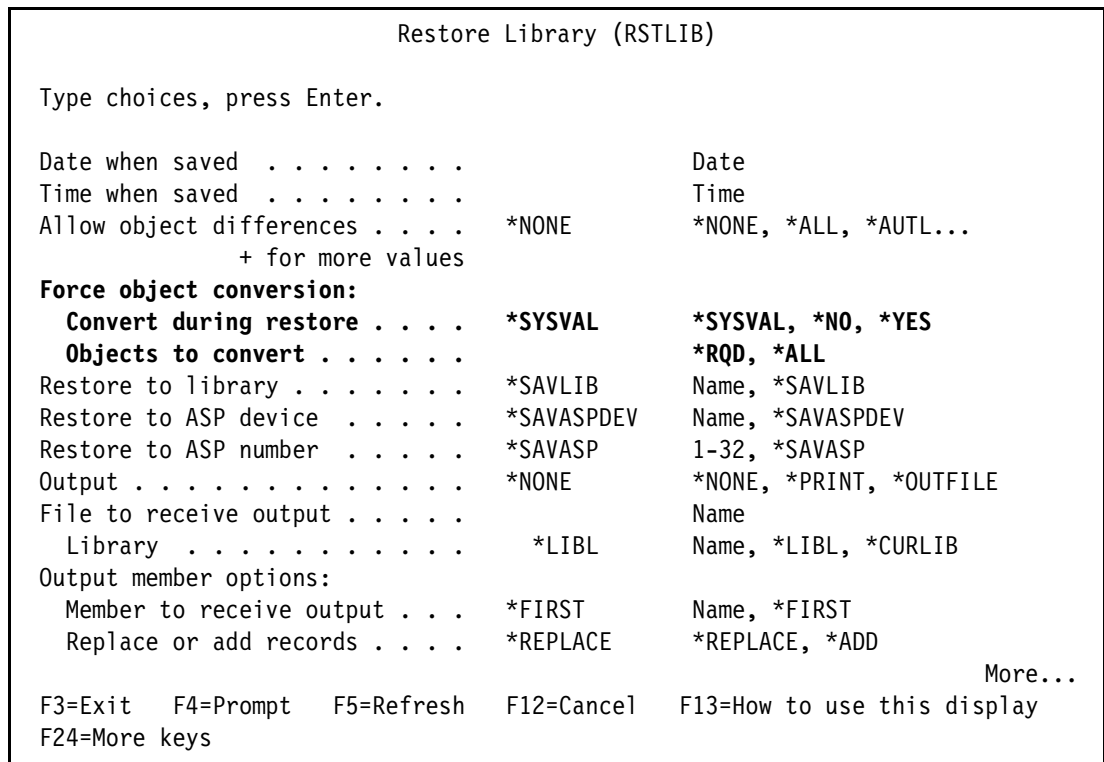


Figure 4-7 Force object conversion parameter on the Restore Library (RSTLIB) command

By changing the system value QFRCCVNRST, the user can turn on and turn off conversion for the entire system. However, on the restore commands, the FRCOBJCVN parameter can override the system value in some cases, for example:

- ▶ Specifying *YES and *ALL on the FRCOBJCVN parameter overrides all settings of the QFRCCVNRST system value.
- ▶ Specifying *YES and *RQD on the FRCOBJCVN parameter is the same as specifying a value of 2 for this system value and can override the QFRCCVNRST system value when it is set to 0 or 1.

Notes: The Load and Run (LODRUN) command, which is used to install certain products, uses restore commands but does not override the QFRCCVNRST system value. Products that require conversion and that use LODRUN for installation can be converted upon installation by setting the system value of QFRCCVNRST to 2 or greater.

Program model objects saved from 6.1 or later with a target release of V5R4 or V5R3 are converted out of the 6.1 format during the restore on the target system. See 3.1.2, “Required PTFs” on page 11, for prerequisite PTFs. After these program model objects have been restored on a V5R4 or V5R3 system, no further user action is required.

Conversions during restore might occur concurrently, three or four at a time, depending on the device that is used for the restore media. For example, restoring from a save file will perform

four conversions at a time. This conversion method is likely to make more effective use of processor capabilities than the first call conversion method.

4.4 Conversion the first time a program is run or called

Program objects that have not been converted during restore or by running the STROBJCVN command, are converted automatically when the program is first run or called. For large programs, this process can be long running. For clients who are concerned about potential performance issues arising from long conversion times, we recommend that you convert program objects before production usage by using the STROBJCVN command. See 4.2, “Conversion using STROBJCVN” on page 47, for details.

Important: First call conversion might be the slowest conversion method, particularly on partitions that have more than one processor available. Actual results will vary depending on the call order and number of jobs that concurrently call programs that have not yet been converted. This is another reason to choose one of the other conversion methods.

4.5 Conversion times and how to improve them

Conversion times reported with the ANZOBJCVN command are estimates. The estimate is based on the processor feature of the system. Actual conversion times might vary from the estimate. Various factors can affect the conversion time, including:

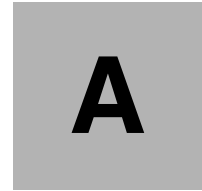
- ▶ Number of processors
- ▶ Disk arms attached
- ▶ Available memory
- ▶ Priority of conversion job
- ▶ Other jobs on system
- ▶ Number of concurrent conversion jobs

In most cases, it is anticipated that object conversion will take a short period of time and will be easily accomplished. However, in some cases, conversion estimates might be longer than you prefer. Use the following suggestions to potentially help reduce the actual conversion time:

- ▶ The STROBJCVN command with the *CONVERT option can be long running. For all systems, but particularly for those with limited interactive (5250 OLTP) capability, we recommend that you run this command in a batch job.
- ▶ For partitions with more than one processor, STROBJCVN is likely to use the opportunity for concurrent conversions most effectively. Conversion during restore is the next most effective method, with conversion during first call likely to be the least effective for using multiple processors.
- ▶ Consider running separate jobs for the conversion of objects in libraries and the integrated file system. If many objects in many libraries require conversion, or if you have many file objects in libraries, regardless of whether they require conversion, consider running more than one job to process objects in libraries.
- ▶ Use STROBJCVN OBJTYPE(*ALLPGM) when upgrading from V5R4 to avoid processing file objects that do not require conversion.

- ▶ If you are upgrading from V5R3, consider running separate jobs for libraries that contain many files or files with many members. Files in libraries might also undergo a form of conversion that can take a significant amount of time.
- ▶ Best performance is achieved if no other user jobs are active on the system while conversion takes place.
- ▶ Because only a single library (aside from *ALLUSR) can be specified on the STROBJCVN command, if objects in many libraries that are not part of *ALLUSR require conversion, it might be useful to write a simple CL program to submit multiple batch jobs, each of which converts a single library.
- ▶ In a logically partitioned environment, consider using dynamic logical partitioning to temporarily increase the number of available processors and the amount of available memory prior to starting any conversions with STROBJCVN. Use data area QGPL/QIZANBRTHD to specify the desired number of threads for each invocation of STROBJCVN or allow the system to use the default of two threads per full processor.
- ▶ Testing has demonstrated that Lotus Domino, and in particular objects in QDOMINOxxx libraries, might take a long time to convert. Remove all old versions of Domino in advance of the upgrade and consider running a separate batch job for the conversion of objects in QDOMINOxxx libraries.
- ▶ If you have a restricted period of time in which the conversions must run, and the estimated conversion time from ANZOBJCVN exceeds the allotted time, consider first converting libraries that contain high-use programs by using the LIB parameter on STROBJCVN. Then run STROBJCVN LIB(*ALLUSR) to convert the remaining objects.

If STROBJCVN does not complete before users must sign on, objects will be converted at first use. If the job running STROBJCVN is consuming too much processor resource when users are signed on, it can be reduced in priority to run in the background or it can be ended.
- ▶ If you plan to convert objects and save them for distribution to pre-POWER6 systems running at 6.1 or later, you might want to convert the objects on a pre-POWER6 system. Converting on a POWER6 or later system allows the machine to use new hardware features in the converted programs. Moving these programs to a pre-POWER6 system could require another conversion. See 3.6.3, “Adaptive code generation” on page 39, for basic information about ACG.
- ▶ If programs that were created or converted on a POWER6 or later system are restored to a pre-POWER6 system, the STROBJCVN command can be used after the restore to perform any necessary conversions.



MRI PTFs

The Analyze Object Conversion (ANZOBJCVN) command is distributed as PTFs for V5R3 or V5R4. For this function, you must order an MRI PTF. By ordering the appropriate MRI PTF for your language, the requisite code PTFs are also ordered. However, there are some circumstances under which you must separately order code PTFs. For additional considerations about required PTFs see 3.1.2, “Required PTFs” on page 11.

Table A-1 contains a complete listing of all the MRI PTFs for i5/OS V5R3 and V5R4.

Table A-1 MRI PTFs for ANZOBJCVN command

Product load	Language	V5R3	V5R4
2924	English (upper/lower case)	SI28425	SI28415
2902	Estonian	SI28472	SI28416
2903	Lithuanian	SI28473	SI28417
2904	Latvian	SI28474	SI28418
2905	Vietnamese	SI28475	SI28419
2906	Laotian	SI28476	SI28420
2909	Belgian English	SI28477	SI28421
2911	Slovenian	SI28478	SI28422
2912	Croatian	SI28479	SI28423
2913	Macedonian	SI28480	SI28424
2914	Serbian (Cyrillic)	SI28482	SI28426
2922	Portuguese	SI28483	SI28427
2923	Dutch	SI28484	SI28428
2925	Finnish	SI28485	SI28429
2926	Danish	SI28486	SI28431

Product load	Language	V5R3	V5R4
2928	French	SI28488	SI28432
2929	German	SI28489	SI28433
2930	Japanese (upper/lower case)	Not available	SI30602
2931	Spanish	SI28490	SI28435
2932	Italian	SI28491	SI28436
2933	Norwegian	SI28492	SI28437
2937	Swedish	SI28493	SI28438
2938	English Uppercase for DBCS (KANJI)	SI28494	SI28439
2939	German MNCS	SI28495	SI28440
2940	French MNCS	SI28496	SI28441
2942	Italian MNCS	SI28497	SI28442
2950	English Uppercase	SI28498	Not available
2954	Arabic	SI28499	SI28443
2956	Turkish	SI28500	SI28444
2957	Greek	SI28501	SI28445
2958	Icelandic	SI28502	SI28446
2961	Hebrew	SI28503	SI28447
2962	Japanese (DBCS)	SI28504	SI28448
2963	Dutch MNCS	SI28505	SI28449
2966	Belgian French	SI28506	SI28450
2972	Thai	SI28512	SI28451
2974	Bulgarian	SI28513	SI28452
2975	Czech	SI28514	SI28453
2976	Hungarian	SI28515	SI28454
2978	Polish	SI28516	SI28455
2979	Russian	SI28517	SI28456
2980	Portuguese	SI28518	SI28457
2981	Canadian French MNCS	SI28519	SI28458
2984	English Upper/Lower for DBCS	SI31794	SI28459
2986	Korean (DBCS)	SI28521	SI28460
2987	Chinese, Traditional	SI28522	SI28461
2989	Chinese, Simplified	SI28523	SI28462
2992	Romanian	SI28524	SI28463
2994	Slovakian	SI28525	SI28464

Product load	Language	V5R3	V5R4
2995	Albanian	SI28527	SI28465
2996	Portuguese MNCS	SI28528	SI28466
2998	Farsi	SI28529	SI28467



B

File field definition tables

In this appendix, we describe the fields in the files that are generated by the Analyze Object Conversion (ANZOBJCVN) command. You can use this field information to help define queries to better understand the provided data.

The data collected by the ANZOBJCVN command is stored in four different files in the QUSRSYS library:

- ▶ File QAIZACVN in QSYS is the master template file and does not get populated with data from collections. However, QAIZACVN in QUSRSYS is the target file for the collection of data analyzed in libraries.
- ▶ File QAIZADIR is the target for directory information when the Object parameter is used.
- ▶ File QAIZAOBJ is the target for integrated file system object information when the Object parameter is used.
- ▶ File QAIZASPL is the target for information for spooled file collections.

Important: Each of these files contains only data from the last collection. Therefore, subsequent collections *will replace* the data that was collected previously in these files. You may want to copy these files to another library to do a later analysis of your collections. Also, these files can be queried to generate additional reports as required. For details about creating queries see 3.5, “Querying analyzed data” on page 36.

QAIZACVN: information for analyzed libraries

Table B-1 lists the field definitions of the QAIZACVN file in the QUSRSYS library. The QAIZACVN file is the target file for the collection of data analyzed in libraries.

Table B-1 QAIZACVN file field definitions

Field name	Description	Attribute
DIOBLI	Library name.	Char(10)
DIOBNM	Name of the object.	Char(10)
DIOBTP	Object type.	Char(7)
DIOBAT	Object attribute.	Char(9)
DIOBSZ	Object size in bytes.	Packed (15,0)
DINOSZ	Not currently used.	Packed (15,0)
DIOBTX	Text description.	Char(50)
DISTIN	Status indicator: ▶ 0 = Normal ▶ 1 = Locked ▶ 2 = Damage	Char(1)
DIOBOW	Object owner.	Char(10)
DIOBAS	Object ASP.	Char(5)
DIASPN	ASP resource name.	Char(10)
DISTGF	Storage freed: ▶ 0 = No ▶ 1 = Yes	Char(1)
DICRTD	Program creation data area indicator: ▶ 0 = No ▶ 1 = Yes ▶ 2 = Not applicable	Char(1)
DICOMP	Earliest release for which object can be saved.	Char(7)
DISFIL	Source file.	Char(10)
DISLIB	Source library.	Char(10)
DISMBR	Source member.	Char(10)
DISMBRE	Source member existence: ▶ 0 = No ▶ 1 = Yes	Char(1)
DIPPID	Program product.	Char(7)
DIREAS	Reason code.	Char(7)
DILCEN	Change century: ▶ 0 = Indicates 19xx ▶ 1 = Indicates 20xx	Char(1)
DILDAT	Change date. The format is mmddy.	Char(6)

Field name	Description	Attribute
DILTIM	Change time.	Char(6)
DIUCEN	Last used century: ▶ 0 = Indicates 19xx ▶ 1 = Indicates 20xx	Char(1)
DIUDAT	Last used date. The format is mmddy.	Char(6)
DIUDTC	Usage data collected: ▶ 0 = No ▶ 1 = Yes	Char(1)
DIDUSD	Days used count.	Packed(5,0)
DICUCR	Century use count reset: ▶ 0 = Indicates 19xx ▶ 1 = Indicates 20xx	Char(1)
DIDUCR	Date use count reset. The format is mmddy.	Char(6)
DIALCH	Allow change by program: ▶ 0 = No ▶ 1 = Yes	Char(1)
DISCEN	Save century: ▶ 0 = Indicates 19xx ▶ 1 = Indicates 20xx	Char(1)
DISDAT	Save date. The format is mmddy.	Char(6)
DISTIM	Save time.	Char(6)
DIACEN	Save active century: ▶ 0 = Indicates 19xx ▶ 1 = Indicates 20xx	Char(1)
DIADAT	Save active date. The format is mmddy.	Char(6)
DIATIM	Save active time.	Char(6)
DIRCEN	Restore century: ▶ 0 = Indicates 19xx ▶ 1 = Indicates 20xx	Char(1)
DIRDAT	Restore date. The format is mmddy.	Char(6)
DIRTIM	Restore time.	Char(6)
DISCMD	Save command.	Char(10)
DISAVF	Save file.	Char(10)
DISAVL	Save file library.	Char(10)
DIFLAB	File label.	Char(17)
DISDEV	Save device: ▶ 0 = Save file ▶ 1 = Tape ▶ 2 = Diskette	Char(1)
DISEQN	Sequence number.	Zoned(8,0)

Field name	Description	Attribute
DIVOL1	Saved volume 1. The tape, diskette or optical volumes used to save the object. A maximum of ten volumes can be saved.	Char(6)
DIVOL2	Saved volume 2.	Char(6)
DIVOL3	Saved volume 3.	Char(6)
DIVOL4	Saved volume 4.	Char(6)
DIVOL5	Saved volume 5.	Char(6)
DIVOL6	Saved volume 6.	Char(6)
DIVOL7	Saved volume 7.	Char(6)
DIVOL8	Saved volume 8.	Char(6)
DIVOL9	Saved volume 9.	Char(6)
DIVL10	Saved volume 10.	Char(6)
DIMORV	More volumes indicator: ▶ 0 = No ▶ 1 = Yes	Char(1)
DICcen	Creation century: ▶ 0 = Indicates 19xx ▶ 1 = Indicates 20xx	Char(1)
DICDAT	Creation date. The format is mmddyy.	Char(6)
DICTIM	Creation time.	Char(6)
DICUSR	Created by user.	Char(10)
DICSYS	System created on.	Char(8)
DICPRS	Compression status: ▶ X = Ineligible ▶ N = No ▶ F = Storage freed ▶ T = Temporary ▶ Y = Yes	Char(1)
DIUSDA	User-defined attribute.	Char(10)
DIODMN	Object domain.	Char(2)
DIOSTA	Object state: ▶ X'8000' = *SYSTEM ▶ X'0001' = *USER ▶ X'0000' = *INHERIT	Char(2)
DIAUDV	Object auditing value.	Char(10)
DIOFSZ	Offline size in bytes.	Hex(8)
DIOVRF	Object overflowed.	Char(1)
DIFILT	File type: S = Source	Char(1)

Field name	Description	Attribute
DINMEM	Number of members.	Binary(2)
DIOCTM	Object conversion time in milliseconds. This value depends on the values in data area QUSRSYS/QIZAFTCD when ANZOBJCVN OPTION(*COLLECT) was run.	Packed(12,0)
DIOSGN	Object sign state: <ul style="list-style-type: none"> ▶ X'00' = Not signed ▶ X'01' = Signed 	Char(1)
DIOPRF	Object profile data: <ul style="list-style-type: none"> ▶ X'00' = Profile data not available ▶ X'01' = Profile data available 	Char(1)
DIRESV	Reserved.	Char(1)

QAIZADIR: information for analyzed directories

Table B-2 lists the fields definitions of the QAIZADIR file in the QUSRSYS library. The QAIZADIR file is the target file for the collection of data analyzed in directories.

Table B-2 QAIZADIR file field definitions

Field name	Description	Attribute
QIZADIRIDX	Identifier of path name (applies only to directories).	Binary(4)
QIZADIRNAM1	The parent directory path. Only used when path length is below 1 KB.	VARGRAPHIC(1024)
QIZADIRNAM2	The parent directory path. Only used when path length is above 1 KB long. Can store paths up to 16 Mb long.	DBCLOB(16M)
QIZADRCCSID	The directory's CCSID.	Binary(4)
QIZADREGION	The directory's path region ID.	GRAPHIC(2)
QIZADLANGID	The directory's path language ID.	GRAPHIC(3)
QIZADIRLEN	Length of directory's path name in bytes.	Binary(4)
QIZADIRFID	The file ID of the directory.	GRAPHIC(35)
QIZADFID	The 4-byte file ID of the directory.	Binary(4)
QIZADIRFSID	The file system ID of the directory.	Binary(8)
QIZADIRGID	The generation ID.	Binary(8)

QAIZAOBJ: information for analyzed integrated file system objects

Table B-3 lists the field definitions of the QAIZAOBJ file in the QUSRSYS library. The QAIZAOBJ file is the target file for the collection of data analyzed in integrated file system objects.

Table B-3 QAIZAOBJ file field definitions

Field name	Description	Attribute
QIZADIRIDX	The parent directory index.	Binary(4)
QIZAOBJNAM	The name of the object.	VARGRAPHIC(1024)
QIZAOBJLEN	Length, in bytes, of the QIZAOBJNAM field.	Binary(4)
QIZACRTTIM	The time the object was created. The format is YYMMDDHHMMSS.	TIMESTAMP(UTC)
QIZAACCTIM	The time the object's data was last accessed. The format is YYMMDDHHMMSS.	TIMESTAMP(UTC)
QIZACHGTIMA	The time the object's attributes were last changed. The format is YYMMDDHHMMSS.	TIMESTAMP(UTC)
QIZACHGTIMD	The time the object's data was last changed. The format is YYMMDDHHMMSS.	TIMESTAMP(UTC)
QIZATRGREL	The creation target version of this object.	GRAPHIC(10)
QIZAOWN	Object owner.	GRAPHIC(10)
QIZAASP	The ASP in which the object is stored.	Binary(2)
QIZABLKSIZE	The block size in bytes.	Binary(4)
QIZANLNK	The number of links for the object.	Binary(4)
QIZAFLEID	The file ID of the object.	GRAPHIC(35)
QIZAFLEIDS	The 4-byte file ID of the object.	Binary(4)
QIZAGENID	The generation ID of the object.	Binary(8)
QIZAFSID	The file system ID.	Binary(8)
QIZADOM	Domain of the object.	GRAPHIC(10)
QIZASTA	State of the object.	GRAPHIC(10)
QIZAPRFDTA	Profiling data present for object: <ul style="list-style-type: none"> ▶ 0 = Not present ▶ 1 = Present 	GRAPHIC(1)
QIZAOPTLVL	The optimization level of the Java program. Valid values are: <ul style="list-style-type: none"> ▶ 0 = This object is not a Java program. ▶ nonzero = Optimization level value of the Java program. 	GRAPHIC(10)
QIZAUDATE	Usage information, last used date. The format is YYMMDDHHMMSS.	TIMESTAMP(UTC)
QIZAUDCOUNT	Usage information, days used count.	Binary(4)

Field name	Description	Attribute
QIZAPRMLNK	When an object has several names, this field is set only for the first name found. The way to know if an object already exists is by checking for its file system ID, file ID, and generation ID. These are unique when used together. ▶ 1 = First name found	Binary(2)
QIZASIG	Whether an object has an i5/OS digital signature. The following values are valid: ▶ 0 = Object does not have an i5/OS digital signature ▶ 1 = Object has an i5/OS digital signature.	Binary(2)
QIZASYSSIG	Whether the object was signed by a source that is trusted by the system. The following values are valid: ▶ 0 = None of the signatures came from a source that is trusted by the system. ▶ 1 = The object was signed by a source that is trusted by the system. If the object has multiple signatures, at least one of the signatures came from a source that is trusted by the system.	Binary(2)
QIZAMLSIG	Whether an object has more than one i5/OS digital signature. The following values are valid: ▶ 0 = The object has only one digital signature. ▶ 1 = The object has more than one digital signature. If the QIZASYSSIG field has the value 1, at least one of the signatures is from a source trusted by the system.	Binary(2)
QIZADIRTY2	The format of the directory object (*DIR). The following values are valid: ▶ 0 = The directory format is *TYPE1. ▶ 1 = The directory format is *TYPE2.	Binary(2)
QIZAFILTYP2	The format of the stream file (*STMF). The following values are valid: ▶ 0 = The stream file format is *TYPE1. ▶ 1 = The stream file format is *TYPE2.	Binary(2)
QIZACASE	Indicates the case sensitivity of the file system that contains this object. The following values are valid: ▶ 0 = File system is not case sensitive. ▶ 1 = File system is case sensitive.	Binary(2)
QIZACVNTIME	Estimated Java program conversion time for an object in milliseconds.	Packed(12,0)
QIZANAMFLG	Indicates whether the object name contains characters affected by the new Unicode casing rules in the target release. The following values are valid: ▶ 0 = Object name does not contain affected characters. ▶ 1 = Object name contains affected characters.	Binary(2)

Field name	Description	Attribute
QIZACVNFLG	Whether the Java program object requires a conversion. The following values are valid: <ul style="list-style-type: none"> ▶ 0 = Object does not require a conversion or is not a Java program. ▶ 1 = Object is a .jar, .zip or .class file with an attached program that requires conversion. 	Binary(2)

QAIZASPL: information for analyzed spooled files

Table B-4 lists the field definitions of the QAIZASPL file in the QUSRSYS library. The QAIZASPL file is the target file for the collection of analyzed spooled files.

Table B-4 QAIZASPL file field definitions

Field name	Description	Attribute
SPDTTM	Date (YYMMDD) and time (HHMMSS). The date and time of the collected data.	Char(12)
SPDCEN	Century digit where: <ul style="list-style-type: none"> ▶ 0 = Indicates 19xx ▶ 1 = Indicates 20xx 	Char(1)
SPSNAM	System name	Char(8)
SPVRM	Version-release-modification. Specifies the release level in VxRyMz format, where: Vx = The version number Ry = The release number Mz = The modification level	Char(6)
SPASP	ASP number. Specifies the ASP number: <ul style="list-style-type: none"> ▶ 1 = The system ASP ▶ 2 through 32 = A basic ASP ▶ 33 to 255 = An independent ASP 	Packed(5,0)
SPASPN	ASP device name. Specifies the name of the ASP. A value of blanks specifies the system ASP or a basic ASP.	Char(10)
SPSPLN	Spooled files. Specifies the number of spooled files in the QSPLxxxx libraries that need to be converted in the ASP.	Packed(11,0)
SPTMCV	Time to convert the spooled files. Specifies the total time in seconds required to convert all the spooled files in the ASP.	Packed(11,0)



IBM software skip-ship products requiring conversion

In this appendix we provide information about IBM skip-ship licensed program products (LPPs) that do not have a new version for 6.1 and must be converted after your system has been upgraded to 6.1. The LPPs listed in Table C-1 on page 74 are the skip-ship products that are supported on 6.1, but do not have a new version for 6.1 and must be converted.

Important: We recommend that before migrating to 6.1 you remove all of the unsupported LPPs for that release. In some cases, if you wait until upgrading to 6.1, you might not be able to completely remove these products easily.

Tip: The Work with Licensed Program (Go LICPGM) menu will not display all program products installed on a system. Use the Display Software Resource (DSPSFWRSC) command to list all of the installed software resources on your system.

Program conversion for LPPs must be done on the product library. Before the product is used for the first time, run the conversion on the library.

If you are installing one of these skip-ship products on 6.1, we recommend that you convert the objects in the product library during the installation. You can do this by using the Force Object Conversion (FRCOBJCVN) parameter on the RSTLICPGM command. For more information see 4.3, “Conversion of program objects during restore” on page 57.

If the skip-ship product was installed before the upgrade, the objects in the product library are not automatically converted as part of the upgrade process. In this case, we recommend that you do conversion by using the STROBJCVN command. There are factors to consider when planning for conversion of these products. For more information about object conversion, refer to Chapter 4, “Program conversion methods” on page 45.

If you receive message CPIB0DC “Digital signature lost for object” during the conversion process, the digital signature has been removed. Digital signatures are normally used only before a product runs, and therefore, do not affect product function.

Domino information: Domino product libraries contain many programs and service programs, some of which are quite large. The conversion process for Domino might take a significant amount of time. For more information specific to Domino, see the Lotus Domino for i5/OS Support Web site at the following address:

<http://www.ibm.com/servers/eserver/series/domino/support/>

Table C-1 Skip-ship products supported on 6.1

Product ID	Version	Product library	Product name
5722BZ100	V5R1M0	QIBS	IBM Business Solutions
5722IA100	V5R2M0	QIA1BASE	IBM Integration Assistant for iSeries
	V5R2M0	QIA1OPT13	Business Integration Connect – Express 4.2
	V5R2M0	QIA1OPT16	WebSphere® Partner Gateway – Express
5722IP100	V5R3M0	QIPS	IBM Infoprint Server for iSeries
5722XP100	V5R3M0	QIWR/IFS	System i Access for Wireless
5723J2500	V7M0	QDOMDOC	Lotus Domino Document Manger
5724H7200	V6R0M0	QMQM QMQMSAMP	WebSphere MQSeries® for iSeries MQ, V6
5724J2300	V7R5M1	QSAMETIME	Lotus Sametime® Instant Messaging
5724J2400	V7R0M0	QQPLACE	Lotus QuickPlace®
5724S31	V8.0.0.2	QQPLACE	Lotus Quickr™
5733FXD00	V4R5M0	QDOMFXD	Domino Fax
5733ID100	V4R5M0	QAFPDNR	Infoprint Designer
5733LD700	V7R0M3	QDOMINO703	Lotus Domino for iSeries V7.0.3
5733LD800	V8R0M1	QDOMINO801	Lotus Domino for iSeries V8
5733LEI	V6R5M4	QDOMINO703 or QDOMNO801	Lotus Enterprise Integrator®
5733W6000	V6R0M0	QWAS60	WebSphere Application Server for OS/400 V6
5733W6100	V6R1M0	QWAS61	WebSphere Application Server V6.1 for i5/OS
5733XT200	V1R1M0	QXMLLIB	XML Toolkit for IBM System i5®
5798FAX00	V5R2M0	QFAX	Facsimile Support
5799PTL0	V5R4M0	QAPTL	iSeries Tools for Developers

Related publications

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

Other publications

These publications are also relevant as further information sources:

- ▶ *Installing, Upgrading, or Deleting i5/OS and Related Software*, SC41-5120-10
<http://www.ibm.com/support/docview.wss?uid=pub1sc41512010>
- ▶ *System i ILE Concepts Version 6 Release 1*, SC41-5606
<http://publib.boulder.ibm.com/infocenter/iseriess/v6r1m0/topic/books/sc415606.pdf>
- ▶ Reclaim Spooled File Storage Experience Report
<http://publib.boulder.ibm.com/infocenter/iseriess/v5r4/topic/experience/spoolreclaim.pdf>

Online resources

These Web sites are also relevant as further information sources:

- ▶ System i i5/OS Memo to Users Version 6 Release 1
<http://publib.boulder.ibm.com/infocenter/iseriess/v6r1m0/topic/rzaq9/rzaq9.pdf>
- ▶ System i Upgrade planning
<http://www-304.ibm.com/systems/support/i/planning/upgrade/index.html>
- ▶ Informational APAR II14306 on Analyze Object Conversion
http://www-912.ibm.com/n_dir/nas4apar.nsf/c79815e083182fec862564c00079d117/3af47a966c4df94586257306003c6868?OpenDocument&Highlight=2,ii14306

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



IBM i Program Conversion: Getting Ready for 6.1 and Beyond



Prepare an earlier release for program conversions to IBM i 6.1 or a later release

IBM delivered IBM i 6.1 in March 2008. With 6.1, IBM exploits the capabilities of the Machine Interface (MI) architecture to significantly improve programs. Programs can benefit from better performance, a range of new operating system and processor capabilities, and even stronger system integrity. To enable these improvements, all MI programs created for previous releases must be converted to run on 6.1 or a later release. MI programs include integrated language environment (ILE) and original program model (OPM) programs.

Use the Analyze Object Conversion (ANZOBJCVN) tool

To convert a program, its creation data, which is a subset of observability, must be available. MI programs retain creation data by default, so that most programs can be converted, going all the way back to programs that were originally created for System/38. Even if an option was chosen to remove creation data from external access, Licensed Internal Code (LIC) can still access the creation data if the target releases of the program and its constituent modules are V5R1 or later. Thus a program lacks sufficient creation data for conversion only if the program or at least one of its modules was created for IBM OS/400 V4R5 or an earlier release and creation data was explicitly removed.

Convert programs to run on 6.1 or a later release

You can run the Analyze Object Conversion (ANZOBJCVN) command on V5R4 or V5R3 to help plan for your upgrade. The ANZOBJCVN command identifies programs that will not convert to run on 6.1 or a later release, if any, and estimates the times that are required for the program conversions. It also provides information about two other types of conversions: integrated file system names and spooled files. You can access the ANZOBJCVN command by loading and applying the appropriate PTFs that are listed in this paper.

This IBM Redpaper publication is intended for system administrators and field technicians to help them understand and prepare for upgrading or migrating to V6R1. It explains what the program conversion is and why programs are being created or re-created on V6R1. It then provides detailed steps for using the ANZOBJCVN tool on V5R3 or V5R4 to prepare for V6R1 object conversions. Finally it discusses the program conversion methods for converting programs to run on V6R1.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks