APPLICATION RESTART CONTROL
User Guide

Supporting

Version 3.9 of APPLICATION RESTART CONTROL for IMS™
Version 3.9 of APPLICATION RESTART CONTROL for DB2®
Version 3.9 of APPLICATION RESTART CONTROL for VSAM

March 2013
Contacting BMC Software

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

**United States and Canada**

<table>
<thead>
<tr>
<th>Address</th>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC SOFTWARE INC</td>
<td>1 713 918 8800 or</td>
<td>1 713 918 8000</td>
</tr>
<tr>
<td>2101 CITYWEST BLVD</td>
<td>1 800 841 2031</td>
<td></td>
</tr>
<tr>
<td>HOUSTON TX 77042-2827</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Outside United States and Canada**

<table>
<thead>
<tr>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>+01 713 918 8800</td>
<td>+01 713 918 8000</td>
</tr>
</tbody>
</table>


BMC, BMC Software, and the BMC Software logo are the exclusive properties of BMC Software, Inc., are registered with the U.S. Patent and Trademark Office, and may be registered or pending registration in other countries. All other BMC trademarks, service marks, and logos may be registered or pending registration in the U.S. or in other countries. All other trademarks or registered trademarks are the property of their respective owners.

CICS, DB2, IBM, IMS, MVS, OS/390, RACF, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

The information included in this documentation is the proprietary and confidential information of BMC Software, Inc., its affiliates, or licensors. Your use of this information is subject to the terms and conditions of the applicable End User License agreement for the product and to the proprietary and restricted rights notices included in the product documentation.

**Restricted rights legend**

U.S. Government Restricted Rights to Computer Software. UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure of any data and computer software by the U.S. Government is subject to restrictions, as applicable, set forth in FAR Section 52.227-14, DFARS 252.227-7013, DFARS 252.227-7014, DFARS 252.227-7015, and DFARS 252.227-7025, as amended from time to time. Contractor/Manufacturer is BMC SOFTWARE INC, 2101 CITYWEST BLVD, HOUSTON TX 77042-2827, USA. Any contract notices should be sent to this address.
Customer support

You can obtain technical support by using the BMC Support Central website or by contacting Customer Support by telephone or e-mail. To expedite your inquiry, see “Before contacting BMC.”

Support website

You can obtain technical support from BMC 24 hours a day, 7 days a week at http://www.bmc.com/support. From this website, you can

- read overviews about support services and programs that BMC offers
- find the most current information about BMC products
- search a database for issues similar to yours and possible solutions
- order or download product documentation
- download products and maintenance
- report an issue or ask a question
- subscribe to receive proactive e-mail alerts when new product notices are released
- find worldwide BMC support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

Support by telephone or e-mail

In the United States and Canada, if you need technical support and do not have access to the web, call 1 800 537 1813 or send an e-mail message to customer_support@bmc.com. (In the subject line, enter SupID:<yourSupportContractID>, such as SupID:12345). Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC

Have the following information available so that Customer Support can begin working on your issue immediately:

- product information
  - product name
  - product version (release number)
  - license number and password (trial or permanent)

- operating system and environment information
  - machine type
  - operating system type, version, and service pack or other maintenance level such as PUT or PTF
  - system hardware configuration
  - serial numbers
  - related software (database, application, and communication) including type, version, and service pack or maintenance level

- sequence of events leading to the issue

- commands and options that you used

- messages received (and the time and date that you received them)
  - product error messages
  - messages from the operating system, such as file system full
  - messages from related software
License key and password information

If you have questions about your license key or password, use one of the following methods to get assistance:

- Send an e-mail message to customer_support@bmc.com.
## Contents

**About this guide**

<table>
<thead>
<tr>
<th>Related publications</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventions</td>
<td>18</td>
</tr>
<tr>
<td>Notes, warnings, and tips</td>
<td>19</td>
</tr>
<tr>
<td>Special conventions</td>
<td>19</td>
</tr>
<tr>
<td>Typographic conventions</td>
<td>19</td>
</tr>
<tr>
<td>Syntax statements</td>
<td>20</td>
</tr>
</tbody>
</table>

### Chapter 1 Introduction

<table>
<thead>
<tr>
<th>Overview</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR/CTL products</td>
<td>22</td>
</tr>
<tr>
<td>AR/CTL functions</td>
<td>23</td>
</tr>
<tr>
<td>BATCH CONTROL FACILITY component of AR/CTL</td>
<td>24</td>
</tr>
<tr>
<td>AR/CTL services</td>
<td>25</td>
</tr>
<tr>
<td>Checkpoint/restart services</td>
<td>26</td>
</tr>
<tr>
<td>Restart processing</td>
<td>27</td>
</tr>
<tr>
<td>Checkpoint processing</td>
<td>28</td>
</tr>
<tr>
<td>Automated checkpoint data set processing</td>
<td>30</td>
</tr>
<tr>
<td>Automated checkpoint/restart determination</td>
<td>30</td>
</tr>
<tr>
<td>Checkpoint control by shift</td>
<td>30</td>
</tr>
<tr>
<td>Application reattach options</td>
<td>31</td>
</tr>
<tr>
<td>IMS-compatible mode</td>
<td>31</td>
</tr>
<tr>
<td>QUICKSTART-to-AR/CTL bridge</td>
<td>32</td>
</tr>
<tr>
<td>Data services</td>
<td>32</td>
</tr>
<tr>
<td>Sequential file interception</td>
<td>33</td>
</tr>
<tr>
<td>Local VSAM access</td>
<td>33</td>
</tr>
<tr>
<td>Remote VSAM access (file sharing)</td>
<td>34</td>
</tr>
<tr>
<td>GSAM replacement</td>
<td>34</td>
</tr>
<tr>
<td>ASAM callable interface</td>
<td>34</td>
</tr>
<tr>
<td>DB2 cursor repositioning</td>
<td>35</td>
</tr>
<tr>
<td>Program exception handling</td>
<td>35</td>
</tr>
<tr>
<td>SQL return code handling</td>
<td>36</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operational services</th>
<th>36</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suspend-and-resume interfaces</td>
<td>37</td>
</tr>
<tr>
<td>Automatic registration of non-IMS programs</td>
<td>37</td>
</tr>
<tr>
<td>AR/CTL for DB2 batch attachment facility</td>
<td>38</td>
</tr>
<tr>
<td>Early termination support</td>
<td>38</td>
</tr>
<tr>
<td>On-demand snap dumps</td>
<td>38</td>
</tr>
<tr>
<td>Enhanced call tracing</td>
<td>39</td>
</tr>
</tbody>
</table>
Chapter 2  Getting started 63

Overview 64
Getting started with AR/CTL implementation  65
    Making implementation decisions  65
    Setting up AR/CTL records  69
    Setting up AR/CTL with application program changes  70
    Setting up AR/CTL with JCL changes  71
    Performing other setup tasks  72
    Executing with AR/CTL  74
    Handling problems  74
Getting started with AR/CTL setup  75
    Creating an environment registration record  75
    Creating a program registration record  76
    Creating an automatic registration record  77
    Creating a program exclusion record  78
    Creating a processing options record  78
    Setting up cursor repositioning options  79
    Setting up reattach options  80
    Setting up program exception class options  80
    Setting up SQL return code options  81
    Setting up dynamic allocation options  82
    Setting the current shift identifier  83
    Setting up checkpoint pacing  84

AR/CTL components 39
AR/CTL ISPF interface 41
Batch utilities 42
Operator communications interface commands 43
BMC Software subsystems 43
Execution modules 44
Registration data sets and REGISET 44
REGISET records 45
History data set 46
Data set options and program options 46
Checkpoint/restart data sets 47
Trace data sets 47
VSAM dynamic backout log data set 48
VSAM file control table data set 48
Implementation of AR/CTL 48
Considerations 49
System resources 49
Required BMC Software products 50
DB2 considerations 50
IMS considerations 53
IMS-compatible mode considerations 55
CICS/VSAM considerations 56
Considerations for using AR/CTL services 57
Considerations for other software products 59
Operational considerations 60
Getting started with AR/CTL operations .................................................. 85
   Using the operator communications interface ................................ 85
   Using statistical reports ................................................................... 87
   Working with active records ............................................................ 87
Excluding or disabling AR/CTL ............................................................. 88
   Exclusion methods ........................................................................... 88
   Exclusion considerations ................................................................. 89
Testing the application program with AR/CTL ...................................... 91
Using AR/CTL DD statements ............................................................. 92
Using the IMS CHKPID parameter ..................................................... 96
Setting up DB2 connections ................................................................. 98
   Specifying DB2 connection information .......................................... 98
   Specifying DB2 connection parameters ....................................... 99
Using the AR/CTL Status Check utility ................................................. 100

**Chapter 3  Using AR/CTL checkpoint/restart services** 103

Overview ......................................................................................... 103
Methods for requesting checkpoint/restart services ................................ 104
Checkpoint/restart processing in an application program ..................... 105
Checkpoint/restart request considerations ........................................... 107
Application program changes between abend and restart ..................... 108
Program code to execute at restart ................................................... 109
Forced checkpoints .......................................................................... 110
Using DB2 commit statements to trigger checkpoints ......................... 110

**Chapter 4  Using AR/CTL data services** 113

Overview ......................................................................................... 114
Methods for using data services .......................................................... 115
General data services considerations .................................................. 116
   Using checkpoint/restart services with data services .................. 116
   Output sequential files .................................................................. 116
   Preallocating data sets ................................................................. 117
   Using GDGs ................................................................................. 118
   Using ASAM data set allocation parameters ................................ 118
Using sequential file interception services ........................................... 119
   Application program calls ............................................................. 119
   Processing option values ............................................................. 120
   Considerations ............................................................................. 120
Using local VSAM access services ....................................................... 121
   Application program calls ............................................................. 121
   Processing option values ............................................................. 122
   Considerations ............................................................................. 123
Using remote VSAM access services ................................................... 123
   Application program calls ............................................................. 123
   Processing option values ............................................................. 125
   VSAM file control table data set ............................................... 125
   AR/CTL enhanced processing ....................................................... 126
   CICS environment and connection information ......................... 126
Chapter 5 Using AR/CTL operational services

Overview .................................................. 138
Using AR/CTL suspend-and-resume interfaces ............................... 138
  Suspend-and-resume overview ........................................ 139
  Interface with BMC Software products for IMS .......................... 140
  Interface with the REORG PLUS for DB2 Online Feature ............... 143
Using early termination support ............................................. 145
  Uses for early termination support ................................... 146
  Backout with early termination ....................................... 146
  Processing options for early termination ............................... 147
  OCI commands for early termination ................................ 147
  Early termination support with an IMS application program ........... 148
Using on-demand snap dumps .............................................. 149
Using enhanced call tracing ................................................ 149
  Overview of enhanced call tracing .................................... 150
  Allocating the ARCDLTRC data set .................................... 151
  Starting and stopping enhanced call tracing ............................ 152
  Using trace records .................................................. 154
Formatting trace data ....................................................... 155
  JCL requirements .................................................. 155
  SYSIN keywords .................................................. 157
Using formatted trace reports ............................................... 160
  General information ................................................ 161
  IMS Trace Header Information report ................................ 161
  DB2 Trace Header Information report ................................ 162
  IMS Trace Detail Information report ................................ 163
  DB2 Trace Detail Information report ................................ 166
  DB2 Trace Trailer Information report ................................ 167
  IMS Trace Trailer Information report ................................ 167
  Trace Summary report ................................................ 168
  Dump Format reports ................................................ 168
### Chapter 7  Changing PL/I application programs 217

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>217</td>
</tr>
<tr>
<td>Supported versions of PL/I</td>
<td>218</td>
</tr>
<tr>
<td>Types of AR/CTL calls</td>
<td>218</td>
</tr>
<tr>
<td>Sample PL/I programs</td>
<td>218</td>
</tr>
<tr>
<td>Coding considerations</td>
<td>219</td>
</tr>
<tr>
<td>EXEC parameters</td>
<td>219</td>
</tr>
<tr>
<td>Compile and link issues</td>
<td>220</td>
</tr>
<tr>
<td>Run-time considerations</td>
<td>221</td>
</tr>
<tr>
<td>Using AR/CTL-format checkpoint/restart calls</td>
<td>221</td>
</tr>
<tr>
<td>Managing virtual storage for subprograms</td>
<td>222</td>
</tr>
<tr>
<td>Implementing the ARCSPVS API</td>
<td>223</td>
</tr>
<tr>
<td>Identifying virtual storage areas to be managed</td>
<td>224</td>
</tr>
<tr>
<td>Creating the $ARCSPVS parameter block</td>
<td>225</td>
</tr>
<tr>
<td>Using the Define function</td>
<td>227</td>
</tr>
<tr>
<td>Using the Refresh function</td>
<td>229</td>
</tr>
<tr>
<td>Using the Delete function</td>
<td>230</td>
</tr>
<tr>
<td>Using AR/CTL common calls</td>
<td>231</td>
</tr>
<tr>
<td>Using IMS-compatible calls</td>
<td>232</td>
</tr>
<tr>
<td>Using ASAM callable interface calls</td>
<td>234</td>
</tr>
</tbody>
</table>

### Chapter 8  Changing Assembler language application programs 237

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>237</td>
</tr>
<tr>
<td>Types of AR/CTL calls</td>
<td>238</td>
</tr>
<tr>
<td>Sample Assembler language programs</td>
<td>238</td>
</tr>
<tr>
<td>EXEC parameters</td>
<td>238</td>
</tr>
<tr>
<td>Assemble and link issues</td>
<td>239</td>
</tr>
<tr>
<td>Using AR/CTL-format checkpoint/restart calls</td>
<td>239</td>
</tr>
<tr>
<td>Managing virtual storage for subprograms</td>
<td>241</td>
</tr>
<tr>
<td>Implementing the ARCSPVS API</td>
<td>241</td>
</tr>
<tr>
<td>Identifying virtual storage areas to be managed</td>
<td>242</td>
</tr>
<tr>
<td>Creating the $ARCSPVS parameter block</td>
<td>243</td>
</tr>
<tr>
<td>Using the Define function</td>
<td>244</td>
</tr>
<tr>
<td>Using the Refresh function</td>
<td>246</td>
</tr>
<tr>
<td>Using the Delete function</td>
<td>247</td>
</tr>
<tr>
<td>Using AR/CTL common calls</td>
<td>248</td>
</tr>
<tr>
<td>Using IMS-compatible calls</td>
<td>249</td>
</tr>
<tr>
<td>Using ASAM callable interface calls</td>
<td>251</td>
</tr>
</tbody>
</table>

### Appendix A  Solving problems 253

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>254</td>
</tr>
<tr>
<td>Handling application program situations</td>
<td>255</td>
</tr>
<tr>
<td>Restarting a non-unique job step</td>
<td>256</td>
</tr>
<tr>
<td>Cold-starting a job step after an abend</td>
<td>256</td>
</tr>
<tr>
<td>Restarting at other than the selected checkpoint</td>
<td>257</td>
</tr>
<tr>
<td>Handling checkpoint/restart data set problems</td>
<td>258</td>
</tr>
<tr>
<td>Correcting a checkpoint data set out-of-space condition</td>
<td>259</td>
</tr>
</tbody>
</table>
Figures

AR/CTL components ................................................................. 40
Application Enhancement Series primary menu ............................. 42
APPLICATION RESTART CONTROL primary menu ........................ 42
Calling sequence for COBOL program ARCACTIV .......................... 101
Calling sequence for PL/1 program ARCACTIV ............................. 101
Calling sequence for Assembler program ARCACTIV ...................... 101
ASAM Copy utility JCL .............................................................. 134
Execution Summary Report ...................................................... 136
Enhanced call tracing structures ............................................. 150
Trace Format utility JCL ........................................................... 155
IMS Trace Header Information report ......................................... 161
DB2 Trace Header Information report ........................................ 162
IMS Trace Detail Information report (I/O PCB variation) (part 1 of 2) 163
IMS Trace Detail Information report (non-I/O PCB variation) ........... 164
IMS Trace Detail Information report (AIB variation) ....................... 164
DB2 Trace Detail Information report ........................................... 166
DB2 Trace Trailer Information report ......................................... 167
IMS Trace Trailer Information report .......................................... 167
Trace Summary report ............................................................ 168
Dump Format report ............................................................... 168
Virtual storage areas in an application program .......................... 185
$ARCSPVS parameter block ................................................... 186
Define function of the ARCSPVS API with SET usage .................. 188
Define function of the ARCSPVS API with ARCSPAD usage .......... 188
Refresh function of the ARCSPVS API ...................................... 189
Delete function of the ARCSPVS API ........................................ 190
AR/CTL common call example COBOL program .......................... 194
Virtual storage areas in an application program .......................... 224
$ARCSPVS parameter block ................................................... 225
Define function of the ARCSPVS API with direct addressing .......... 227
Refresh function of the ARCSPVS API ...................................... 229
Delete function of the ARCSPVS API ........................................ 230
AR/CTL common call example PL/1 program ............................. 231
Virtual storage areas in an application program .......................... 242
$ARCSPVS parameter block ................................................... 243
Define function of the ARCSPVS API ....................................... 245
Refresh function of the ARCSPVS API ...................................... 246
Delete function of the ARCSPVS API ........................................ 247
AR/CTL common call example Assembler language program ........ 248
Program ARCCCTRL ............................................................... 275
## Tables

AR/CTL tasks ................................................................. 64  
CHKPID parameter keywords ........................................... 97  
Checkpoint/restart services summary .................................. 104  
Data services summary .................................................... 115  
Trace record types .......................................................... 154  
AR/CTL-format checkpoint/restart call summary .................... 177  
$ARCSPVS parameter block fields ....................................... 187  
$ARCSPVS parameter block fields for the Define function ....... 189  
$ARCSPVS parameter block fields for the Refresh function ...... 190  
$ARCSPVS parameter block fields for the Delete function ...... 190  
IMS-compatible checkpoint/restart call summary .................. 195  
Checkpoint PCB mask fields .............................................. 201  
GSAM replacement ASAM call summary .............................. 203  
ASAM PCB mask fields ..................................................... 210  
Record search argument format ......................................... 211  
$ARCSPVS parameter block fields ....................................... 226  
$ARCSPVS parameter block fields for the Define function ....... 228  
$ARCSPVS parameter block fields for the Refresh function ...... 229  
$ARCSPVS parameter block fields for the Delete function ...... 230  
$ARCSPVS parameter block fields ....................................... 243  
$ARCSPVS parameter block fields for the Define function ....... 245  
$ARCSPVS parameter block fields for the Refresh function ...... 247  
$ARCSPVS parameter block fields for the Delete function ...... 248  
Non-standard situations .................................................... 254  
Utilities for creating diagnostic tape .................................. 266  
Sample programs ............................................................ 272  
Sample JCL ................................................................. 273  
Sample macros and definitions .......................................... 273  
Sample user exit routines .................................................. 274
About this guide

This guide provides overview, background, and orientation information about the BMC Software APPLICATION RESTART CONTROL (AR/CTL) products. The information is for database administrators (DBAs), system programmers, application programmers, operators, and job schedulers. This guide assumes that you are familiar with the subsystems your company uses to manage data (IBM® IMS™, DB2®, or VSAM subsystems), with IBM MVS™ JCL, and with your company’s conventions for the setup and use of application programs.

This guide works with the other AR/CTL manuals. Read this guide first before you start using AR/CTL.

Like most BMC documentation, this book is available in printed and online formats. To request printed books or to view online books and notices (such as release notes and technical bulletins), see the support website at http://www.bmc.com/support.

NOTE
Online books are formatted as PDF or HTML files. To view, print, or copy PDF books, use the free Adobe Reader from Adobe Systems. If your product installation does not install the reader, you can obtain the reader at http://www.adobe.com.

The software also offers online Help. To access Help, press F1 within any product or click the Help button in graphical user interfaces (GUIs).

Related publications

From the BMC Support Central website (http://www.bmc.com/support), you can use either of the following methods to access related publications that support your product or solution:

- Link to the BMC Documentation Center (https://webapps.bmc.com/infocenter/index.jsp) to browse documentation sets

- View BMC Quick Course Demos (short overviews of selected product concepts, tasks, or features), which are included in the BMC Documentation Center
Read individual product documents (books and notices) within the “A – Z Supported Product List”

You can order hardcopy documentation from your BMC sales representative or from the support site. You can also subscribe to proactive alerts to receive e-mail messages when notices are issued.

## Conventions

The guide uses several conventions to present information. This section describes these conventions.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS</td>
<td>The term IMS refers to all supported versions and releases of IBM IMS®. The specific product name, version, and release numbers are noted only when this information is significant.</td>
</tr>
<tr>
<td>DB2</td>
<td>The term DB2 refers to all supported versions and releases of IBM DB2®. The specific product name, version, and release numbers are noted only when this information is significant.</td>
</tr>
<tr>
<td>AR/CTL products</td>
<td>The collective term AR/CTL products refers to AR/CTL for IMS, AR/CTL for DB2, AR/CTL for VSAM, and the BATCH CONTROL FACILITY (BCF) component of AR/CTL.</td>
</tr>
<tr>
<td>Checkpoint/restart</td>
<td>Unless stated otherwise, the term checkpoint in this guide refers to application checkpoints rather than system checkpoints. The term restart refers to application restarts, not system restarts.</td>
</tr>
<tr>
<td>Keywords</td>
<td>In keyword descriptions, the characters shown in uppercase are required; you can omit the lowercase characters. When this guide discusses a keyword in text or shows it in an example, it uses the complete keyword in uppercase.</td>
</tr>
<tr>
<td>Libraries</td>
<td>This guide refers to various members of sample and install libraries. As distributed by BMC Software, these libraries have the following names:</td>
</tr>
<tr>
<td></td>
<td>■ HLQ.ARCSAMP</td>
</tr>
<tr>
<td></td>
<td>■ HLQ.ARCCUST</td>
</tr>
<tr>
<td></td>
<td>The variable HLQ is a high-level qualifier that should be easily identified by all installation users in your facility. Because the person who installed AR/CTL products at your site could have chosen any name, the guide refers to these libraries generically as sample libraries and install libraries.</td>
</tr>
</tbody>
</table>
Notes, warnings, and tips

This book uses notes and warnings as follows:

**NOTE**

Notes contain important information for you to consider or point out information that can help make the product easier to use.

**WARNING**

Warnings alert you to problem situations. For example, a warning could advise you of a potential loss of data if you do not carefully follow the instructions.

Special conventions

This book uses the following special conventions:

- All syntax, operating system terms, and literal examples are presented in this typeface.

- Variable text in path names, system messages, or syntax is displayed in *italic* text:

  `testsyst/instancelfilename`

- The symbol `=>` connects items in a menu sequence. For example, `Actions => Create Test` instructs you to choose the `Create Test` command from the `Actions` menu.

Typographic conventions

The guide uses the following typographic conventions to identify product elements:

<table>
<thead>
<tr>
<th>Description</th>
<th>Style</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entries you type</td>
<td>ALL CAPS, BOLDFACE</td>
<td>Type ADD at the cursor.</td>
</tr>
<tr>
<td>Specific key names</td>
<td>Initial Caps, Boldface</td>
<td>Press Enter.</td>
</tr>
<tr>
<td>Commands, keywords, parameters, nonspecific key names</td>
<td>SMALL CAPS</td>
<td>Use the TYPE keyword to select control block type.</td>
</tr>
</tbody>
</table>
Syntax statements

The following example shows a sample syntax statement:

```
COMMAND KEYWORD1 [KEYWORD2 | KEYWORD3] KEYWORD4={YES | NO} fileName...
```

The following table explains conventions for syntax statements and provides examples:

<table>
<thead>
<tr>
<th>Item</th>
<th>Example</th>
</tr>
</thead>
</table>
| Items in italic type represent variables that you must replace with a name or value. If a variable is represented by two or more words, initial capitals distinguish the second and subsequent words. | alias
|                                                                      | databaseDirectory
|                                                                      | serverHostName        |
| Brackets indicate a group of optional items. Do not type the brackets when you enter the option. A comma means that you can choose one or more of the listed options. You must use a comma to separate the options if you choose more than one option. | [tableName, columnName, field]
|                                                                      | [-full, -incremental, -level] (UNIX) |
| Braces indicate that at least one of the enclosed items is required. Do not type the braces when you enter the item. | {DBDName | tableName} |
|                                                                      | UNLOAD device={disk | tape, fileName | deviceName} |
|                                                                      | {a | c} (UNIX)          |
| A vertical bar means that you can choose only one of the listed items. In the example, you would choose either commit or cancel. | {commit | cancel} |
|                                                                      | {commit | cancel} (UNIX)  |
| An ellipsis indicates that you can repeat the previous item or items as many times as necessary. | columnName ... |

The Next Page field contains the number of the page to display next.
Chapter 1

Introduction

This chapter introduces the APPLICATION RESTART CONTROL (AR/CTL) products from BMC Software. This chapter contains the following information:

Overview .......................................................... 22
   AR/CTL products .................................................. 23
   AR/CTL functions .................................................. 24
   BATCH CONTROL FACILITY component of AR/CTL .......... 25
   AR/CTL services .................................................. 26
Checkpoint/restart services ...................................... 26
   Restart processing ............................................... 27
   Checkpoint processing .......................................... 28
   Automated checkpoint data set processing ...................... 30
   Automated checkpoint/restart determination .................. 30
   Checkpoint control by shift .................................... 30
   Application reattach options ................................... 31
   IMS-compatible mode .......................................... 31
   QUICKSTART-to-AR/CTL bridge ................................. 32
Data services ..................................................... 32
   Sequential file interception .................................... 33
   Local VSAM access ............................................. 33
   Remote VSAM access (file sharing) ............................ 34
   GSAM replacement ............................................. 34
   ASAM callable interface ....................................... 34
   DB2 cursor repositioning ...................................... 35
   Program exception handling ................................... 35
   SQL return code handling ...................................... 36
Operational services ............................................. 36
   Suspend-and-resume interfaces ................................. 37
   Automatic registration of non-IMS programs ................. 37
   AR/CTL for DB2 batch attachment facility ..................... 38
   Early termination support ..................................... 38
   On-demand snap dumps ....................................... 38
   Enhanced call tracing .......................................... 39
AR/CTL components ............................................. 39
   AR/CTL ISPF interface ......................................... 41
   Batch utilities ................................................ 42
BMC Software offers the following AR/CTL products to meet the needs of application programs that execute in the MVS system environment:

- APPLICATION RESTART CONTROL for IMS
- APPLICATION RESTART CONTROL for DB2
- APPLICATION RESTART CONTROL for VSAM

The AR/CTL products work separately or together to provide services for application programs that use the following types of data structures:

- sequential files, using queued sequential access method (QSAM)
- DB2 tables
- IMS full-function, Fast Path, and/or GSAM databases
- virtual storage access method (VSAM) data sets
- any combinations of these data structures
AR/CTL products

Each AR/CTL product, alone or with other AR/CTL products, supports certain types of application programs.

**APPLICATION RESTART CONTROL for IMS**

AR/CTL for IMS works independently in application programs that use IMS databases only, sequential files only, or both. AR/CTL for IMS works with AR/CTL for VSAM in programs that use IMS databases and VSAM data sets. These programs can also use sequential files.

AR/CTL for IMS works with AR/CTL for DB2 in programs that use DB2 tables, IMS databases, and DB2-oriented services of AR/CTL (explained later in this chapter) in an IMS environment. These programs can also use sequential files. However, AR/CTL for IMS can work alone with programs that use DB2 tables and IMS databases but do not use DB2-oriented services of AR/CTL.

**APPLICATION RESTART CONTROL for DB2**

AR/CTL for DB2 works independently in application programs that use DB2 tables only, sequential files only, or both.

AR/CTL for DB2 works with AR/CTL for VSAM in programs that use DB2 tables and VSAM data sets. These programs can also use sequential files.

AR/CTL for DB2 works with AR/CTL for IMS in programs that use DB2 tables, IMS databases, and DB2-oriented services of AR/CTL in an IMS environment. These programs can also use sequential files.

**APPLICATION RESTART CONTROL for VSAM**

AR/CTL for VSAM works independently in application programs that use VSAM data sets only, sequential files only, or both.

AR/CTL for VSAM works with AR/CTL for DB2 and AR/CTL for IMS in any program that uses VSAM data sets. These programs can also use sequential files.
AR/CTL functions

AR/CTL products use the following functions:

- the AR/CTL Installation System
- communication with BMC Software subsystems
- AR/CTL application supervisor
- AR/CTL enhanced processing
- internal and external security
- panels for managing the AR/CTL ISPF environment
- utilities for managing the registration data sets

AR/CTL Installation System

The AR/CTL Installation System (AESIS) configures the AR/CTL products, the BMC Software subsystems, and the registration data sets. For more information, see the APPLICATION RESTART CONTROL Configuration Guide.

Communication with BMC Software subsystems

AR/CTL functions communicate with the BMC Primary Subsystem (BMCP) and the BMC Consolidated Subsystem (BCSS). These subsystems perform many APF-authorized tasks for AR/CTL products, allow implementation of AR/CTL products without JCL changes, and manage the registration data sets (collectively called the REGISET) that contain information essential to product operations. For more information, see the APPLICATION RESTART CONTROL Administrator Guide.

AR/CTL application supervisor

The AR/CTL application supervisor handles communication and coordination among AR/CTL products, the application program, and IMS (if applicable).

AR/CTL enhanced processing

AR/CTL enhanced processing provides enqueue/dequeue services for AR/CTL for VSAM. AR/CTL enhanced processing executes at the MVS system level.
Internal and external security

AR/CTL product components often require other security in addition to the security provided by the system security manager. You can use AR/CTL external or internal security to prevent unauthorized access to AR/CTL product functions and components. Access can be defined for individual users and groups of users. AR/CTL external security uses techniques compatible with the IBM Resource Access Control Facility (RACF®). AR/CTL internal security uses an AR/CTL-defined data set and load module. For more information, see the APPLICATION RESTART CONTROL Administrator Guide.

Panels for managing the AR/CTL ISPF environment

AR/CTL products share a set of panels that you can use to manage the AR/CTL ISPF environment. They include panels for showing and hiding IMS, DB2, and VSAM information on AR/CTL panels; panels for displaying information about your system environment; and panels for setting default values for printing and submitting jobs. For more information, see the APPLICATION RESTART CONTROL Administrator Guide.

Utilities for managing the REGISET

AR/CTL products share the REGISET Copy utility, an ISPF-based utility for copying AR/CTL product records to a different REGISET (with or without a record ID change) or to the same REGISET (with a record ID change). This utility is intended for use as a migration tool to help you move from a test to a production environment. For more information, see the APPLICATION RESTART CONTROL Administrator Guide.

BATCH CONTROL FACILITY component of AR/CTL

AR/CTL for IMS can work closely with the BATCH CONTROL FACILITY (BCF) component of AR/CTL. The AR/CTL BCF component automates and controls the IMS batch DL/I backout process and enhances the IMS batch DL/I logging environment. Although some options of AR/CTL require the BCF component to be active, AR/CTL and the BCF component are independent; you can obtain the major benefits of one without using the other. When used together, AR/CTL and the BCF component provide significant benefits for batch DL/I programs. For more information about the AR/CTL BCF component, see the APPLICATION RESTART CONTROL General Information: BATCH CONTROL FACILITY manual.
The AR/CTL products provide checkpoint/restart services, data services, and operational services for application programs. Depending on the application program type and the specific AR/CTL services you want to use, the application program might require no program changes, no JCL changes, and no relinking to use AR/CTL:

- Checkpoint/restart services provide (or enhance) an environment that allows an application program job step to be restarted from an application checkpoint after a failure.

- Data services focus on reestablishing the application program’s position within data structures at restart. Data services also provide availability, integrity, and performance enhancements for supported data structures.

- Operational services focus on improving your control over the application program execution-time environment; they include suspend-and-resume interfaces for other BMC Software products, early termination support, on-demand snap dumps, and enhanced call tracing.

### Checkpoint/restart services

Checkpoint/restart services provide (or enhance) an environment that allows an application program job step to be restarted after a failure. AR/CTL checkpoint/restart services offer the following features:

- restart processing
- checkpoint processing
- automated management of checkpoint/restart data sets
- automated determination of checkpoint/restart requirements
- checkpoint pacing by shift
- application reattach options
- IMS-compatible mode
Restart processing

At the beginning of an application program job step, the program issues a request for restart services through a program call or through an AR/CTL processing option. In response to this request, AR/CTL determines whether the last execution of the job step terminated normally or abnormally. If the last execution terminated normally, AR/CTL sets up the extended restart environment for a normal start. If the last execution terminated abnormally, AR/CTL sets up the environment for restart and performs the following processing:

- program restart at the last checkpoint

  Restart occurs from the last checkpoint that completed before the failure. Without restart, all changes that the job step made to databases and data sets must be backed out (or you must recover the databases and data sets); then the job step must start over from the beginning. With restart, only the changes that the job step made since the last checkpoint must be backed out; then the job step resumes processing at the last checkpoint.

- automatic repositioning of data structures that use AR/CTL data services

  During restart, AR/CTL can reestablish the program’s position within supported data structures. Repositioning allows the program to continue processing the data set at the point where it was reading or writing data at the last checkpoint. IMS reestablishes position within IMS databases. AR/CTL reestablishes position within the data sets using AR/CTL data services. AR/CTL can reestablish position within the DB2 cursor, or the program can reestablish the position itself by saving the necessary information during checkpoint processing.

- restoration of working storage areas

  During restart, AR/CTL can restore the program’s working storage areas in main memory. These working storage areas were saved in the checkpoint record written during checkpoint processing. Restoring the working storage areas allows the program to continue processing at the last checkpoint.

  Through the Subprogram Virtual Storage (ARCSPVS) application program interface (API), AR/CTL can also restore saved areas of virtual storage for subprograms that are executing under the main program. If an application subprogram already uses the subprogram support that is provided with the QUICKSTART for MVS product, no program changes are required for the use of AR/CTL support for subprograms.
AR/CTL offers several methods for an application program to request restart services, depending on the program type:

- The Automatic Restart option is available in all AR/CTL products. The option is available for many COBOL application programs, but it has some restrictions on usage. The option is set externally from the program; no program changes are necessary to use this option. Other options allow you to control the restoration of working storage at restart.

- The AR/CTL common call is available in all AR/CTL products for use with all supported types of programs. The program issues the AR/CTL common call with the XRST function code as the first program call.

- The IMS-format extended checkpoint/restart (XRST) call is available in AR/CTL for IMS. The call is for use with any IMS program or IMS-compatible program. If an existing program issues the XRST call, no changes are necessary.

- The AR/CTL restart (ARCXRST) call is available in AR/CTL for DB2 and AR/CTL for VSAM. The call is for use with any non-IMS program. If the program issues the ARCXRST call, it must be the first program call.

**Checkpoint processing**

An application program can issue a request for checkpoint services through a program call, or AR/CTL can perform checkpoint processing automatically at externally defined points during processing. Usually, the checkpoint request is issued at the end of each unit of work (UOW).

In response to a checkpoint request, AR/CTL saves the application program’s working storage areas (including virtual storage areas in subprograms), saves the information needed to reestablish the program’s position within the data sets that use AR/CTL data services, and signals all database management systems (DBMSs), such as DB2 and IMS, to perform commit processing.

Commit processing makes the database changes permanent and releases the locks that the DBMS has placed on the resources used by the application program, making those resources available for other processing. Timely release of resource locks is important to system performance.
AR/CTL offers several methods for an application program to request checkpoint services, depending on the program type. Most of these methods eliminate or minimize the changes required to implement checkpoint/restart services in an application program. The following methods are available:

- The Automatic Checkpoints option is available in all AR/CTL products. The option is available for many COBOL application programs, but it has some restrictions on usage. AR/CTL can issue an automatic checkpoint based on activity against a specified trigger, which can be a sequential file, a VSAM data set, a DB2 cursor if you have AR/CTL for DB2, or an IMS program communication block (PCB) if you have AR/CTL for IMS. The option is set externally from the program; most programs require no changes to use the option.

- The AR/CTL common call is available in all AR/CTL products for use with all supported types of programs. This call allows programmers to use a single call format regardless of the DBMS in use. The program issues the AR/CTL common call with the CHKP function code.

- The AR/CTL checkpoint (ARCECHK or ARCCHKP) call is available in AR/CTL for DB2 and AR/CTL for VSAM and can be used in any non-IMS program.

- Programs that use AR/CTL for DB2 can issue DB2 commit requests, and AR/CTL regards each commit request as a trigger for a checkpoint. This method is available in AR/CTL for DB2 only, and the program cannot not use IMS. The program can use the Automatic Restart option.

- Programs that use IMS (or IMS-compatible calls and structures) can issue the IMS extended checkpoint (CHKP) call. If an existing program already issues the CHKP call, no change is required. This method is available in AR/CTL for IMS only.

**NOTE**

An application program can use only one method; mixing methods is not valid.

Regardless of the method an application program uses to request checkpoint services, AR/CTL expands the scope of the request to include processing for all relevant DBMSs involved in the application program execution.
Automated checkpoint data set processing

AR/CTL employs efficient, automated methods for managing checkpoint data sets. For each application program execution, AR/CTL dynamically allocates a checkpoint data set to contain the two most recent checkpoints for an application program. As the program executes and a new checkpoint occurs, AR/CTL writes the new checkpoint to the checkpoint data set, reusing space occupied by the oldest of the two checkpoints.

If program execution completes normally, AR/CTL automatically deletes the checkpoint data set because the checkpoint records it contains are no longer needed for restart. If the program abnormally terminates or a system failure occurs, AR/CTL retains the checkpoint data set, which becomes the restart data set for the next execution. When the program is restarted, AR/CTL accesses the restart data set to obtain the information necessary for restart.

Automated checkpoint/restart determination

After an application program job step is submitted for execution, AR/CTL performs these tasks:

- determines automatically whether the job step requires a normal start or a restart
- determines which checkpoint is valid for restart
- provides the checkpoint information to the application program in restart situations
- if the AR/CTL BCF component is active during backout of a DLI batch job step, verifies that the checkpoint is the same one that the BCF component used for backout

Checkpoint control by shift

AR/CTL can control the frequency of the checkpoints taken by an application program in two ways:

- AR/CTL can monitor the threshold (maximum) interval allowed without a checkpoint. If no checkpoint is issued within that interval, AR/CTL can issue a warning or an abend for the job step.
AR/CTL can control the pacing (minimum) interval between checkpoints and bypass (pace) checkpoints that do not meet the defined pacing requirements. Checkpoint pacing allows the program to issue requests for checkpoints at logical points during processing without concern for performance issues. The program can also issue an explicit call to force a checkpoint without regard for the pacing requirements.

You can define multiple checkpoint pacing classes. In each class, you can set up separate pacing options that depend on the current shift (such as prime, second, and third), on whether the program is executing with batch or online characteristics, or on other criteria you define. You can use a pacing user exit routine to override or accept the pacing recommendation.

**Application reattach options**

To reduce the need for operator intervention in certain abend conditions (such as deadlock and rollback), AR/CTL can reattach an application program instead of terminating when these abends occur. To MVS, the job step is still executing; however, to the DBMS, the application program has abended. The DBMS backs out the uncommitted changes made by the application program. AR/CTL can reposition the supported data structures when AR/CTL reattaches the application program.

Application reattach options require that the environment support automatic backout.

**IMS-compatible mode**

You might be using IMS in a DLI batch application program solely to obtain support for checkpoint/restart and sequential file repositioning (with the generalized sequential access method, or GSAM). To reduce or eliminate usage of IMS resources, AR/CTL for IMS can replace IMS for this type of application program. No application program changes are needed. You can continue to use existing database descriptions (DBDs) and program specification blocks (PSBs).

When AR/CTL replaces IMS, the application program executes in IMS-compatible mode. The program can issue no IMS calls except for checkpoint, extended restart, and GSAM calls, and it must conform to the IMS rules for using these calls. AR/CTL constructs an IMS-like environment and performs checkpoint/restart and GSAM processing as though IMS were present, but no IMS modules are invoked and no IMS control blocks are used.

To use IMS-compatible mode, you must have AR/CTL for IMS.
QUICKSTART-to-AR/CTL bridge

To allow most programs that contain calls to the BMC Software QUICKSTART for MVS product to work with AR/CTL without program changes, AR/CTL provides the QUICKSTART for MVS-to-AR/CTL bridge.

AR/CTL supports all QUICKSTART calls, including checkpoint calls in subprograms. AR/CTL provides the Subprogram Virtual Storage (ARCSPVS) application program interface (API) for managing defined areas of virtual storage for any application program, including a subprogram, in a restartable environment. If an application subprogram already uses the subprogram support that is provided with the QUICKSTART for MVS product, no program changes are required for the use of AR/CTL support for subprograms.

For support of rollback checkpoint calls, you can use AR/CTL application reattach options. A rollback-type call forces a user abend, which causes changes to be backed out. AR/CTL treats a restart as a result of a rollback as it does any other restart. If you set up application reattach options and ensure that a restart from the beginning of the UOW is valid, a rollback-type call restarts application processing automatically at the last checkpoint.

Conversion of an existing QUICKSTART program to work with AR/CTL is a simple process that involves no application program changes. For more information, see the APPLICATION RESTART CONTROL Configuration Guide.

Data services

Data services focus on reestablishing an application program’s position within non-DBMS data sets at restart. Data services can also provide availability, integrity, and performance enhancements for supported data sets. AR/CTL provides the following data services, depending on the type of application program and the AR/CTL products you have:

- sequential file interception (QSAM)
- local VSAM access
- remote VSAM access (also called file sharing)
- IMS generalized sequential access method (GSAM) replacement
- application sequential access method (ASAM) callable interface
- DB2 cursor repositioning
- program exception handling
- SQL return code handling

In a non-IMS environment, an application program must use AR/CTL checkpoint/restart services to use repositioning.
Sequential file interception

Sequential file interception is provided for data sets that an application program accesses through standard MVS QSAM requests. AR/CTL dynamically intercepts these requests so that it can reposition the data set at restart time. A user exit and output staging (where AR/CTL holds output data in memory and writes it only at checkpoint time) are also available.

If an existing program contains standard MVS QSAM requests, no program changes are necessary to implement sequential file interception services. Sequential file interception is available with all AR/CTL products.

Local VSAM access

Local VSAM access is provided for VSAM data sets that are accessed exclusively by a batch VSAM application program. The program issues native VSAM requests. AR/CTL for VSAM dynamically intercepts these requests to provide repositioning support and, as options, VSAM logging and dynamic backout to the last completed checkpoint. These functions were formerly provided by the RECOVERY PLUS for CICS®/VSAM (RPCV) product from BMC Software.

If you use VSAM logging and dynamic backout during application program execution, AR/CTL writes log records to a VSAM dynamic backout log. Each log record contains the image of the record from the updated VSAM data set as the record existed before the update. If a failure occurs, AR/CTL uses the log records to back out (or reverse) the changes to restore the VSAM data set to its condition at the last completed checkpoint. The data set is then ready for job step restart.

In most failure situations, backout completes before the job step terminates. In situations where AR/CTL does not receive control for termination processing (such as if a system failure occurs or the job is cancelled), backout is performed at restart.

AR/CTL uses a write-ahead log (WAL) to ensure that dynamic backout can be performed in all abend situations. For better performance of application programs that perform mainly sequential updates, AR/CTL can defer writing of the WAL buffers. If WAL writes are deferred, AR/CTL can perform dynamic backout for all user abends and most system abends; however, if a system failure occurs or the job is cancelled, AR/CTL cannot perform dynamic backout. You must recover the data sets from a backup and rerun the job from the beginning.

If an existing program contains native VSAM requests, no program changes are necessary to implement local VSAM access. To use local VSAM access, you must have AR/CTL for VSAM.
Remote VSAM access (file sharing)

Remote VSAM access (also known as VSAM file sharing) allows VSAM data sets to be shared and updated between a batch VSAM application program and one or more IBM CICS® regions. The batch program and the CICS regions can execute on a single MVS image or in different MVS images within a SYSPLEX environment. While a file is being shared, the CICS region owns the data sets; it manages all access to the data sets, logs changes made by those transactions, and backs out uncommitted changes in case of failure. AR/CTL dynamically intercepts the native VSAM requests issued by a batch VSAM application program, transforms the requests to online CICS commands, and ships the commands to the CICS region for service.

If an existing program contains native VSAM requests, no program changes are necessary to implement remote VSAM access. To use remote VSAM access, you must have AR/CTL for VSAM.

GSAM replacement

GSAM replacement is provided for IMS and IMS-compatible application programs. The program issues standard IMS GSAM calls. In response to these calls, AR/CTL replaces the IMS GSAM services and provides services that IMS does not provide, including output staging, a user exit, and record-level repositioning.

If an existing program contains GSAM calls, no program changes are necessary to implement GSAM replacement. If a program uses IMS for checkpoint/restart and GSAM services only, you can remove IMS from the job step execution without program changes. To use GSAM replacement, you must have AR/CTL for IMS.

ASAM callable interface

The ASAM callable interface is provided mainly for DB2 application programs to use if AR/CTL for VSAM is not available. A program can use this interface to obtain limited support for repositioning of VSAM entry-sequenced data sets (ESDSs), input VSAM key-sequenced data sets (KSDSs), and standard MVS sequential files. Output staging, a user exit, and record-level repositioning are also available.

The program must be changed to issue AR/CTL ASAM calls. To use the ASAM callable interface, you must have AR/CTL for DB2 or AR/CTL for VSAM.
DB2 cursor repositioning

A major problem encountered in the DB2 checkpoint/restart environment is cursor repositioning at restart time. AR/CTL can perform repositioning automatically for the DB2 cursors used by an application program. After an abend, cursor position (the point where the application program was processing) is lost. If you do not use AR/CTL automatic cursor repositioning, at restart the application program must reestablish the cursor position at the last completed checkpoint. AR/CTL for DB2 offers the following methods for performing automatic DB2 cursor repositioning; these methods are implemented externally to the application program and require no program changes:

**COLUMN**

In the cursor repositioning record, you define one or more columns from the cursor. The columns specified must create a unique key for the cursor row. As each fetch cursor request is performed, AR/CTL retains the data from these columns for checkpoints and uses the data at restart to locate the correct row during the first fetch cursor.

**COUNT**

AR/CTL maintains a count of fetch cursor operations and, at restart, issues fetches until it reaches the correct row within the cursor. This method is useful if no rows are added to or deleted from the cursor before the cursor position at the last checkpoint.

To use automatic DB2 cursor repositioning, you must have AR/CTL for DB2. For more information, see the *APPLICATION RESTART CONTROL Reference Manual*.

---

**NOTE**

DB2 cursor repositioning is not available when the cursor is a scrollable rowset cursor.

---

Program exception handling

AR/CTL can intercept and attempt to handle application program exception conditions that usually result in a system 0Cx-type abend. This program exception handling might allow application processing to run to completion, allowing you to correct the problem after execution completes successfully.

AR/CTL can handle a program exception in any of the following ways:

- Report the exception condition. Ignore the condition, and continue with the next program instruction.
- Report the exception condition. Zero the result field of the abending instruction, and continue with the next program instruction.

- Report the condition. Restart the unit-of-work (UOW), which means abend the application program job step, invoke back out of changes to the last completed checkpoint, and then restart the application program at that checkpoint. (For more information about backout, see the APPLICATION RESTART CONTROL Reference Manual.) Skip the input record that the application program was processing when the exception occurred, and continue processing with the next record.

- Report the condition. Restart the unit-of-work. Skip the input record that the application program was processing when the exception occurred, write the skipped record to a reject file, and continue processing with the next record. For more information, see the APPLICATION RESTART CONTROL Reference Manual.

**SQL return code handling**

AR/CTL can intercept a defined SQL return code received during application program processing and issue a defined user abend code and reason code. This abend code and reason code can be defined as a combination that is eligible for application reattach processing.

SQL return code interception requires a license for AR/CTL for DB2. For more information, see the APPLICATION RESTART CONTROL Reference Manual.

**Operational services**

Operational services focus on improving your control over an application program execution-time environment. The following operational services are available:

- suspend-and-resume interfaces for other BMC Software products
- automatic registration of non-IMS programs
- the AR/CTL for DB2 batch attachment facility
- early termination support
- on-demand snap dumps
- enhanced call tracing

Some operational services are available through the command-oriented operator communications interface (OCI). Others are available through AR/CTL processing options. Some are available through the OCI and processing options.
Suspend-and-resume interfaces

AR/CTL provides suspend-and-resume interfaces for many BMC Software products. Suspend-and-resume processing allows these products to obtain a point of consistency required for reorganization or recovery. (In recovery products, the process of obtaining this point of consistency is known as Get and Hold Recovery Point.)

Using a combination of early termination and application reattach support, AR/CTL suspends batch application processing when needed so that the other BMC Software product can obtain a point of consistency. After this point is obtained, AR/CTL restarts the application program. Suspend-and-resume processing is automatic and transparent to the application program.

Batch application programs that are not under the control of AR/CTL cannot participate in suspend-and-resume processing.

A license for AR/CTL for DB2 is required if you want to use the suspend-and-resume interface with BMC Software products for DB2. A license for AR/CTL for IMS is required if you want to use the suspend-and-resume interfaces with BMC Software products for IMS. For more information about the interfaces, see “Using AR/CTL suspend-and-resume interfaces” on page 138.

Automatic registration of non-IMS programs

For an AR/CTL product to participate in the execution of a non-IMS application program, an applicable non-IMS program registration record must be present in the REGISET. You can create a non-IMS program registration record manually, or the AR/CTL application supervisor might be able to create these records automatically from the criteria that you specify in an automatic registration record.

For DB2 application job steps, automatic registration records identify the ddname that, if present in the job step, causes the AR/CTL application supervisor to register the job step for AR/CTL product services. The AR/CTL application supervisor can also extract DB2 connection parameters from the data set that is associated with this ddname. The connection parameters can be indicated in the nodes of the data set name or can be contained in the first record in the data set. The program can be registered automatically for full non-IMS AR/CTL services or for the AR/CTL for DB2 batch attachment facility only.

For non-DB2, non-IMS programs, automatic registration records identify the ddname that, if present in the job step, causes the AR/CTL application supervisor to register the job step for AR/CTL product services.
AR/CTL for DB2 batch attachment facility

AR/CTL for DB2 provides a batch attachment facility that an application program can use for connection to a DB2 subsystem. You can register a program to use all applicable AR/CTL services or to use the batch attachment facility only. If you register the program for the batch attachment facility only, AR/CTL for DB2 can provide a call attachment facility (CAF) connection or a Recoverable Resource Manager Services attachment facility (RRSAF) connection to the DB2 subsystem. (AR/CTL provides no RRSAF support other than the connection to DB2.) You can also specify an abend completion code or error return code to be used if a problem occurs during the connection to DB2.

Early termination support

Early termination support allows you to terminate an application job after the next checkpoint, immediately after the next call, at a specified time of day, after a specified amount of time, after a specified number of checkpoints, before a specified number of checkpoints, and before the final checkpoint.

Termination after a checkpoint requires no backout because the checkpoint signifies that the changes are committed. Immediate termination requires backout of uncommitted changes. Early termination support always leaves restart information intact and ready for restarting the job at the last completed checkpoint.

Early termination support can be useful in these situations:

- stopping a job step that is executing online in preparation for restarting it in batch, or stopping a batch execution for restart in the online environment
- testing restart logic without changing the program
- quiescing a job in preparation for scheduled system maintenance (the job can be restarted and completed after normal operations are resumed)

On-demand snap dumps

You can use the OCI to obtain a standard MVS snap dump of application program storage. You can use the snap dump to analyze application program problems. The snap dump does not cause program termination.
Enhanced call tracing

Enhanced call tracing can help you develop, test, debug, and audit application programs. Enhanced call tracing provides information about the IMS DL/I calls, IMS Fast Path calls, DB2 SQL statements, and AR/CTL ARCCHKP, ARCECHK, and ARCXRST calls issued by an application program. Unlike many existing trace facilities, enhanced call tracing allows you to begin and end tracing while the application program is executing. It also places the trace data in a separate data set for easy access and does not significantly degrade overall system performance.

DL/I and Fast Path trace records contain the results of the call: the entire PCB, including the key feedback, and part of the resulting I/O area. SQL trace records contain the SQL statement type, the precompiler statement number, and the name of the program. AR/CTL provides a batch utility for formatting the trace records.

Enhanced call tracing does not support program requests that AR/CTL intercepts through dynamic call interception.

AR/CTL components

AR/CTL consists of the following components (Figure 1 on page 40):

- AR/CTL Interactive System Productivity Facility (ISPF) interface
- batch utilities
- OCI commands
- BMC Software subsystems
- AR/CTL execution modules
- REGISET and REGISET records
- history data set
- data set options and program options
- checkpoint/restart data set
- trace data sets
- VSAM dynamic backout log data set (not shown)
- VSAM file control table data set (not shown)

AR/CTL and the AR/CTL BCF component (if active) execute under the AR/CTL application supervisor, which manages coordination and communication among the products. The application program also executes under the AR/CTL application supervisor.
AR/CTL ISPF interface

AR/CTL provides an ISPF interface for accomplishing most user tasks. The ISPF interface consists of a set of ISPF panels and their supporting dialogs that use common user access (CUA) guidelines. You can use the ISPF interface to perform most product tasks, including these:

- register environments to work with AR/CTL products
- register programs for AR/CTL product participation
- exclude programs from AR/CTL product participation
- view and modify AR/CTL active job information
- identify the current shift for AR/CTL checkpoint pacing
- set AR/CTL checkpoint pacing options
- view AR/CTL statistical reports
- set AR/CTL processing options
- provide AR/CTL dynamic allocation skeleton information
- set AR/CTL data set options and program options
- set AR/CTL remote VSAM access options
- set AR/CTL DB2 cursor and long cursor repositioning options
- set AR/CTL program exception handling options
- set AR/CTL SQL return code handling options
- work with the AR/CTL environment

Figure 2 on page 42 shows the Application Enhancement Series primary menu. Figure 3 on page 42 shows the APPLICATION RESTART CONTROL primary menu. For more information about the AR/CTL ISPF interface, see the APPLICATION RESTART CONTROL Administrator Guide. For more information about the AR/CTL ISPF interface, see the APPLICATION RESTART CONTROL Reference Manual.
Batch utilities

AR/CTL provides batch utilities for accomplishing selected product tasks:

- automatically prepare for restart after a system failure
- set the current shift identifier
- format the enhanced call tracing data set
- copy an ASAM data set without reblocking it
- list AR/CTL active job step records

Figure 2   Application Enhancement Series primary menu

AESPRIM            Application Enhancement Series V.X.X.X
Command ===> _________________________________________________________________

Select an option. Then press Enter.

Application Enhancement Series (AES)
  1. AES records
  2. BATCH CONTROL FACILITY (BCF)
  3. APPLICATION RESTART CONTROL (AR/CTL)

AES Common Options
  11. Display, print, jobcard, allocation, and profile options
  12. Messages
  13. Security
  14. Product authorization
  15. REGIST record copy utility
  16. Exit

Figure 3   APPLICATION RESTART CONTROL primary menu

APPLICATION RESTART CONTROL
Command ===> _________________________________________________________________

Type or verify the ID of the BMC Consolidated Subsystem (BCSS) to use.  
Blank the field to request a list of active AES/BCSS subsystems. 
BMC Consolidated Subsystem ID . . . ____
Current BCSS . . . . . . . . . . : 

Select an option. Then press Enter.

  1. Active jobsteps                  9. Dynamic allocation options
  3. Checkpoint pacing options       11. Remote VSAM options
  4. Reports                        12. Program exception class options
  5. Processing options              13. SQL return code options
  6. Cursor repositioning options    14. AES common utilities
  7. Long cursor repositioning opts  15. Exit
  8. Reattach options
Operator communications interface commands

The OCI accepts commands that you can issue from the master console to invoke and control early termination support, enhanced call tracing, and on-demand snap dumps. These commands are a fast, efficient method for using the operational services of AR/CTL.

BMC Software subsystems

AR/CTL uses BMC Software standard subsystems for resource management. A major advantage of using subsystem architecture is that it reduces virtual storage requirements in user address spaces. AR/CTL uses two BMC Software subsystems—the BMC Primary Subsystem (BMCP) and the BMC Consolidated Subsystem (BCSS). For more information about the subsystems, see the APPLICATION RESTART CONTROL Administrator Guide.

BMCP

The BMCP establishes supervisory services for the BCSS and many BMC Software products. It allows interception of open, close, attach, and link requests in the system. (AR/CTL enhanced processing is used for interception of enqueue and dequeue requests for remote VSAM access in AR/CTL for VSAM.) Because BMC Software products share the BMCP, you need only one copy of the BMCP on an MVS system. All associated BMC Software products continue to operate normally even if the BMCP terminates; however, BMC Software recommends that you leave the BMCP running at all times.

During configuration of AR/CTL or other products that use the BMCP, the BMCP is set up to start at each IPL.

BCSS

The BCSS manages I/O to the registration data sets (collectively called the REGISET), manages APF-authorized functions, and performs processing for intercepted open, close, attach, and link requests.

The BCSS must be active on the MVS system where you want to execute application programs that use AR/CTL products and where you want to access records in the REGISET through the ISPF interface. During configuration of AR/CTL or other products that use the BCSS, the BCSS is set up to start at each IPL.
More than one BCSS can be active on an MVS system. One must be defined as the public BCSS; it is typically used for production work. Each of the others is defined as a private BCSS; a private BCSS is typically used for testing new releases of AR/CTL. A default BCSS can be designated for all AR/CTL products on each MVS. You use the BCSS identifier (BCSID) to identify the subsystem (and REGISET) you want to use.

Three AR/CTL components are available with the BCSS: the IMS component, the non-IMS component, and the VSAM component. The non-IMS component must be enabled to allow AR/CTL for DB2 and AR/CTL for VSAM to operate. The IMS component must be enabled to allow AR/CTL for IMS (and the AR/CTL BCF component) to operate. The VSAM component must be installed to allow AR/CTL to perform VSAM logging and dynamic backout.

You can use the AR/CTL Status Check utility (program AESUVBCS) to ensure that the required AR/CTL products, components, and functions are available for application program execution.

## Execution modules

AR/CTL execution modules perform the major functions of AR/CTL. These modules must be available during application program execution through the environment registration record in the REGISET. AR/CTL does not require APF authorization.

## Registration data sets and REGISET

A registration data set is a VSAM KSDS that contains the information necessary for central control over various phases of application program processing with AR/CTL and the AR/CTL BCF component (if active). During configuration of AR/CTL or other BMC Software products that use the BCSS, you define a set of registration data sets, called the REGISET. The BCSS manages the REGISET. Each BCSS uses only one REGISET, but a REGISET can be shared by more than one BCSS (if the BCSSs are running on different MVS images).

You can define two types of registration data sets—primary and duplex—and you can define multiple data sets of each type. Each registration data set contains the same records as the other copies contain, and the BCSS keeps the records and their contents in sync. The REGISET usually is allocated as two primary registration data sets, but it can consist of a maximum of 11 primary and 11 duplex registration data sets.

- BCSS uses a primary registration data set when providing AR/CTL products with responses to REGISET queries. You must have at least one primary registration data set; you can have as many as 11. If you have more than one and the active primary registration data set fails, the next primary registration data set is
available and normal processing continues. You can recover the failing data set at a more convenient time, and recovery of the failing data set does not require restoration. If you encounter a failure in a primary registration data set and no other primary registration data set is available, the BCSS stops.

- BCSS uses a duplex registration data set to maintain an active backup copy of the active primary registration data set. Use of a duplex registration data set is optional; you can define as many as 11. Duplex registration data sets are used only for recovering primary registration data sets.

You can restore a primary registration data set by using another primary registration data set in the REGISET, a duplex registration data set in the REGISET, or a backup copy of the primary registration data set.

If you use the AR/CTL BCF component, AR/CTL and the BCF component must share the REGISET. AR/CTL can share the REGISET with other BMC Software products that use the BCSS (such as DATA ACCELERATOR and the DATA PACKER products).

For information about how to allocate and manage the REGISET, see the APPLICATION RESTART CONTROL Administrator Guide.

REGISET records

Most of the information that AR/CTL needs for controlling and managing application program processes is stored in records in the REGISET. You create, update, and delete many types of these records through the ISPF interface. AR/CTL creates and updates some types of these records, and you can also view, change, and delete them through the ISPF interface.

Each record is identified by qualifiers in the record key. These qualifiers relate to the run-time environment and can include items such as the MVS ID, the job name, the step name, and the program name. To reduce the number of records to manage, AR/CTL allows you to use wildcard characters, with specific characters, in the records you create.

For information about the types of REGISET records used by AR/CTL, the qualifiers used in the record keys, and the use of wildcard characters in these qualifiers, see the APPLICATION RESTART CONTROL Reference Manual.
History data set

AR/CTL uses the history data set to store history information (statistical reports) about application program job steps. This VSAM KSDS can be created and dynamically allocated during program execution.

For information about the history data set, see the APPLICATION RESTART CONTROL Reference Manual.

Data set options and program options

For non-IMS application programs, AR/CTL uses a data set option member (also known as a file characteristics block, or FCB) to determine which data services to provide for a data set. You can create FCBs for individual data sets, or you can create a default FCB to use if no specific FCB is available. AR/CTL can generate an FCB internally during execution if you do not provide a specific or default FCB.

For non-IMS application programs, AR/CTL can use a program option member (also known as an application specification block, or ASB) to identify the FCBs used by the application program and to determine certain characteristics of the application program. The program can use a default ASB (called AUTO$ASB) to manage all files that the program uses.

Data set and program options are external to the application program, data sets, and JCL; no changes to existing structures are necessary. You can use the AR/CTL ISPF interface to set up the options. The option members are stored in one or more libraries that must be available during application program execution. AR/CTL can allocate the option member libraries dynamically.

If the application program uses IMS, it continues to use IMS database descriptions (DBDs) and program specification blocks (PSBs). You can set up default or specific data set and program option members to obtain individual AR/CTL data services. An IMS program can also use the default ASB (AUTO$ASB) through a processing option.

For information about data set options and program options, see the APPLICATION RESTART CONTROL Reference Manual.
Checkpoint/restart data sets

AR/CTL uses a checkpoint/restart data set to store the information needed to restart an application program in case of failure. During AR/CTL initialization of each normal-start job step, AR/CTL dynamically allocates a new checkpoint data set. During checkpoint processing, AR/CTL writes a checkpoint record to the checkpoint data set; the record contains the information necessary to reestablish data set position and restore working storage areas. During restart after a job step abend or a system failure, AR/CTL uses the checkpoint data set from the failed execution as the restart data set. From the restart data set, AR/CTL reads the latest valid checkpoint record to obtain the information necessary to restart the job step.

AR/CTL manages the checkpoint/restart data set as a VSAM linear data set. The data set needs to be only large enough to hold two checkpoint records plus a control record (an allocation size of one to five cylinders is typical). During checkpoint processing, AR/CTL overwrites the oldest of the two checkpoint records with the new checkpoint record. In this manner, one checkpoint is always valid for restart.

AR/CTL automatically allocates the checkpoint/restart data set from dynamic allocation information you provide through the AR/CTL ISPF interface. The checkpoint/restart data set can use space on direct access storage device (DASD) volumes managed by the Storage Management Subsystem (SMS). AR/CTL automatically deletes the checkpoint/restart data set when the checkpoint records are no longer needed for restart purposes.

For application programs that use local VSAM access services and VSAM logging and dynamic backout, AR/CTL uses a write-ahead log (WAL) to ensure data integrity. The WAL is part of the checkpoint/restart data set.

Trace data sets

AR/CTL can produce extensive trace information during the execution of an application program. Enhanced call trace data is helpful for application program debugging. ASAM and VSAM trace data is for BMC Software Product Support to use in problem diagnosis.
VSAM dynamic backout log data set

For application programs that use local VSAM access services, AR/CTL can use a VSAM dynamic backout log data set to perform VSAM logging and dynamic backout. To provide access to the log data set during application program execution, AR/CTL can allocate the data set dynamically from a dynamic allocation record in the REGISET or a DD statement can be added to the program execution JCL. If the program was previously set up to use RPCV for VSAM logging and dynamic backout, no change is necessary.

VSAM file control table data set

For application programs that use remote VSAM access services, AR/CTL uses the VSAM file control table data set to track information about the CICS regions and VSAM files that are participating in file sharing. All CICS regions and batch application programs in the file sharing environment use the same VSAM file control table data set. The data set is allocated and formatted during configuration of AR/CTL for VSAM. To provide access to the data set from a CICS region, a DD statement is added to the CICS startup JCL. To provide access from a batch application program, AR/CTL can allocate the data set dynamically from a dynamic allocation record in the REGISET or a DD statement can be added to the program execution JCL.

Implementation of AR/CTL

AR/CTL offers streamlined methods for obtaining AR/CTL services in many application programs. These streamlined methods require no application program changes, no recompiling or relinking of application programs, and no JCL changes. For many application programs, implementation of AR/CTL is as easy as this:

1. Configure AR/CTL and the supporting AR/CTL, BMCP, and BCSS components in your environment. The AR/CTL Installation System tailors JCL to perform almost all configuration tasks from the information you provide through an ISPF dialog.

2. Create the required records in the REGISET. Global records of most required record types are created during AR/CTL configuration. You must manually create only one type of record—the program registration record. Creating REGISET records is easy to do with the ISPF interface, and you can use wildcard characters to reduce the number of records created.
To obtain checkpoint/restart services, most COBOL application programs can use the Automatic Restart and Automatic Checkpoints processing options of AR/CTL. You can set these options externally to application programs through the processing options record in the REGISET.

To obtain most data services, no action is necessary. To obtain cursor repositioning services, you create cursor repositioning records in the REGISET.

3 Execute application programs with AR/CTL. If a failure occurs during job step execution, correct the problem that caused the failure and resubmit the job step.

Considerations

This section describes considerations for using AR/CTL in your environment. For a discussion of the system resources required for AR/CTL, see the Installation System User Guide.

NOTE

For complete information about AR/CTL support for operating systems, hardware, and database management systems, you can use the Product Availability and Compatibility (PAC) utility on the BMC Software Customer Support Web site at http://www.bmc.com/support_home.

System resources

AR/CTL requires the following system resources.

Hardware

AR/CTL operates on any processor that supports the IBM z/OS® operating system. BMC Software licenses AR/CTL to run on specific CPUs.

Operating system

AR/CTL is supported for use with all IBM-supported versions and releases of the z/OS operating system.

At application program execution, AR/CTL determines the version of MVS you are using and selects the proper routines to execute for that version.
Required BMC Software products

The AR/CTL products you need depend on the application program type:

- A program that uses IMS or IMS-compatible structures and calls requires AR/CTL for IMS. If you want to use the application reattach options with a batch DLI program, the AR/CTL BCF component is required.

- A program that uses native VSAM requires AR/CTL for VSAM for easiest implementation and unrestricted support for VSAM data sets.

- A program that needs to share a VSAM file with a CICS region requires AR/CTL for VSAM.

- A program that uses local VSAM access services, including VSAM logging and dynamic backout, requires AR/CTL for VSAM.

- A program that uses DB2 but does not use IMS requires AR/CTL for DB2.

- A program that uses DB2 with IMS requires AR/CTL for IMS. AR/CTL for DB2 is optional; however, the AR/CTL functions that are specific to DB2 are available only with AR/CTL for DB2.

- A program that uses QSAM files can use AR/CTL for DB2, AR/CTL for VSAM, or AR/CTL for IMS. If the program uses AR/CTL for IMS, it must execute program DFSRRC00.

NOTE
Any application programs that start with DFS or DSN are automatically excluded for AR/CTL. This exclusion includes IMS and possibly DB2 IVPs.

DB2 considerations

The following shows how AR/CTL supports DB2:

DB2 requirements

If you are using AR/CTL for DB2, your site must have a licensed copy of DB2. AR/CTL is supported for use with all IBM-supported versions and releases of DB2.
AR/CTL product license

If a CPU authorization password to enable AR/CTL for DB2 participation is not installed, DB2 calls are ignored and these DB2-oriented services of AR/CTL are unavailable:

- automatic DB2 cursor repositioning
- checkpoint pacing by SQL call counts
- checkpoint threshold monitoring by SQL call counts
- automatic checkpoints based on a trigger DB2 cursor
- enhanced call tracing of SQL calls
- attach to the DB2 subsystem on behalf of the application program
- SQL return code handling
- suspend-and-resume interfaces with BMC Software products for DB2

Cursor WITH HOLD option

AR/CTL does not change the cursor to use the WITH HOLD option. If the program uses AR/CTL checkpoint services and the cursor does not use the WITH HOLD option, the program must tolerate cursor position being lost when a commit is performed.

Connection types

If you use the package method of binding a DB2 application program, you must enable the correct connection types for the DB2 attach method you want to use. By default, all connection types are enabled in the BIND PACKAGE. If you do not use this default and you use the CAF attach method, enable DB2CALL. If you do not use this default and you use the DSN attach method, enable DLIBATCH. AR/CTL might force the use of DSN in some cases; therefore, you might want to enable both DB2CALL and DLIBATCH or use the default ENABLE(*). If the correct connection type is not enabled, the application program receives an SQL -807 return code.

AR/CTL DB2 plan

The AR/CTL DB2 plan, default name ARCTLP$3, must be bound (and execute authority must be granted) on each DB2 subsystem where application programs use automatic checkpoints by a trigger cursor or automatic cursor repositioning. Use the same plan name for all DB2 subsystems. The default plan name can be changed. For more information, see the APPLICATION RESTART CONTROL Configuration Guide.
Products that perform batch attach

AR/CTL replaces any other method, including program IKJEFT01, that you might be using to perform a batch attach of the application program. For DB2 batch application programs using AR/CTL, any other product performing attachments to DB2 on behalf of the DB2 application program must be removed from the STEPLIB, JOBLIB, and LNKLST concatenation. An application program can be registered to use the AR/CTL batch attachment facility without using other AR/CTL services.

Statically-linked DB2 language interface module

You must recompile and/or relink an existing non-IMS DB2 program if the program load module includes (statically links) a DB2 language interface module (DSNALI, DSNELI, or DSNHLI). This static linking might have been performed through the COBOL NODYNAM compile option, with a link-edit INCLUDE statement, or in a PL/I environment. The DB2 language interface module must be replaced with the AR/CTL language interface module (ARCLI000). If the program calls any other program or subprogram that has included a DB2 language interface module, you must also recompile and/or relink the called program or subprogram. If the DB2 language interface module remains statically linked and is executed with AR/CTL, the result is an SQL -927 return code.

An IMS/DB2 program does not need to be relinked even if the DB2 language interface module is statically linked.

CAF programs

AR/CTL handles the connection to DB2 on behalf of the application program, replacing the DB2 call attach facility (CAF). However, AR/CTL supports existing CAF programs without program changes. If the DB2 language interface module (DSNALI or DSNELI) is loaded dynamically (whether the CAF connection to DB2 is performed implicitly or explicitly), no action is necessary to use the CAF program with AR/CTL. If the program uses an implicit connection and the DB2 language interface module has been included in the load module (static linking), the program must be relinked to replace the DB2 language interface module with the AR/CTL language interface module (ARCLI000); no recompile is required. If the program uses an explicit connection and the DB2 language interface module has been statically linked, the program must be recompiled with the DYNAM compile option and relinked to replace the DB2 language interface module with the AR/CTL language interface module (ARCLI000).
Execution without AR/CTL

If you want a non-IMS DB2 application program to execute with AR/CTL at times and without AR/CTL at other times, you must provide your own DSNHLLI language interface module, as explained in the DB2 documentation. Your language interface module must determine whether AR/CTL should participate and, if not, your module should perform the connection to (and termination from) DB2 on behalf of the application program. You must link this module with the application program.

Data sharing groups

AR/CTL supports the data sharing group attachment name as the DB2 subsystem name in the subsystem member (SSM) if the application program uses the CAF value for the DB2Attach Type option. AR/CTL does not provide this support if the application program uses the DSN value for the DB2Attach Type option. With this attach type, AR/CTL simulates an IMS batch attach, which does not support the group attachment name.

IMS considerations

The following shows how AR/CTL supports IMS:

IMS requirements

If you are using AR/CTL for IMS (in a true IMS environment), your site must have a licensed copy of the IMS Database Manager. AR/CTL is supported for use with all IBM-supported versions and releases of IMS.

**NOTE**

When you migrate from one release of IMS to a new release, IMS requires all application programs to complete normally under the old release before executing under the new release. IMS cannot use a log data set created under a previous release during restart of a program on a new release.

AR/CTL product license

If a CPU authorization password to enable AR/CTL for IMS participation is not installed, AR/CTL does not participate in any IMS region. No IMS-oriented services of AR/CTL are available, including the suspend-and-resume interface with BMC Software products for IMS.
PSB PROCOPTs

AR/CTL supports all PSB processing options (PROCOPTs) and DL/I databases.

Region types

IMS application programs use the same EXEC statement with AR/CTL as they use without AR/CTL. AR/CTL supports the following IMS region types:

DLI
   IMS mode and IMS-compatible mode

DBB
   IMS mode and IMS-compatible mode

BMP
   IMS mode (AR/CTL supports BMPs running in an IMS DB/CTL environment.)

AR/CTL supports the EXEC DLI environment. AR/CTL also supports the use of IMS-defined keywords that suppress checkpoint messages written to the operator console.

Automatic checkpoints

When the Automatic Checkpoints processing option is set to Y for an IMS program, AR/CTL saves the entire working storage area of the application program at each checkpoint performed. For programs (especially in a DLI batch environment) that have large working storage areas, consider the effect of writing the checkpoint records to the IMS logs and take appropriate action to prevent problems.

Fast Path databases

When an IMS application program accesses a Fast Path database, a final checkpoint must be performed before the dependent region terminates. Other checkpoints might be required in response to a status code from DL/I. Certain AR/CTL checkpoint/restart services can affect required checkpoints:

checkpoint pacing
   Checkpoint pacing can cause a required checkpoint to be bypassed. Do not turn on checkpoint pacing unless you use the Checkpoint Pacing exit (force required checkpoints to be processed). Or you can change the calls in the application program to use the AR/CTL common call with the FORCE option.
**automatic checkpoint/restart**
When AR/CTL performs automatic checkpoint/restart for an application program, AR/CTL cannot determine when the application program is ready to terminate. Do not use automatic checkpoint/restart in an application program that accesses a Fast Path database.

**Abend and restart on different IMS systems**

AR/CTL supports application programs that abend on one IMS system and are restarted on another IMS system. The IMSID qualifier in the program registration record must contain wildcards.

**Multiple MVS tasks**

AR/CTL for IMS services must be invoked under the same task as the IMS region controller (DFSRRC00). Neither AR/CTL nor IMS can perform error recovery under a sibling or daughter task.

**IMS-compatible mode considerations**

The following shows how AR/CTL supports IMS-compatible mode environments, where AR/CTL replaces IMS.

**AR/CTL product license**

You must have a CPU authorization password to enable AR/CTL for IMS.

**IMS requirements**

If you are using AR/CTL for IMS in IMS-compatible mode, no IMS license is required. In IMS-compatible mode, AR/CTL uses no IMS code.

**IMS calls**

The application program cannot issue any other types of IMS calls except for checkpoint (CHKP), extended restart (XRST), and GSAM-type calls. The program cannot access any full-function IMS databases.
Generated control blocks

The application program can continue to use existing DBDs and PSBs that are generated by IMS utilities. AR/CTL does not provide utilities to generate these control blocks. If you no longer have the IMS utilities for control block generation and you need to change or create a control block for an application program, you must replace all IMS-format control blocks with AR/CTL-format control blocks.

IMS region types

The application program JCL must be set up to execute a DLI batch region. Batch message processing (BMP) regions are not supported for IMS-compatible mode.

Automatic restart and automatic checkpoints options

For IMS-compatible mode programs, AR/CTL does not support the Automatic Restart option or the Automatic Checkpoints option.

CICS/VSAM considerations

The following shows how AR/CTL for VSAM supports the CICS/VSAM environment.

CICS

If you want to use remote VSAM access services in a batch VSAM application program, your site must have a licensed copy of the IBM CICS product. AR/CTL is supported for use with all IBM-supported releases of CICS and CICS Transaction Server.
AR/CTL product license

If a CPU authorization password to enable AR/CTL for VSAM participation is not installed, no native VSAM requests are intercepted and these VSAM-oriented services of AR/CTL are unavailable:

- local VSAM access services
- remote VSAM access services (file sharing)
- checkpoint pacing by VSAM call counts
- checkpoint threshold monitoring by VSAM call counts
- automatic checkpoints based on a trigger VSAM file
- repositioning of VSAM data sets at restart without application program changes

VSAM batch local shared resources

AR/CTL supports VSAM batch LSR for direct access VSAM data sets. AR/CTL does not support VSAM batch LSR for sequential access VSAM data sets.

Considerations for using AR/CTL services

Consider the following issues when you use AR/CTL services.

Temporary data sets

If an application program uses a temporary data set, make sure that it is not needed for restart. Because the temporary data set will not be present at restart, an application program cannot be restarted if it requires the temporary data set at restart. If the application program requires the data set at restart, change the job step to allocate a permanent data set. If the program does not require the data set at restart, you can turn off repositioning for the temporary data set in the data set options member.

Concatenated input

Change in the status of concatenated input is not supported for restart when the number of concatenations is less at restart than in the original execution.

Input sequential PDS and PDS/E members

AR/CTL supports repositioning of members of a sequential partitioned data set (PDS) or sequential extended PDS (PDS/E) for input processing.
Considerations for using AR/CTL services

NOTE
You cannot mix PDSs and PDS/Es.

Rollback processing

In an IMS application program, IMS-format rollback calls are valid only if the program uses no data services (sequential interception, local and remote VSAM, and GSAM replacement). AR/CTL passes ROLB and ROLS calls on to IMS for processing; for other IMS rollback calls, AR/CTL issues a user 778 abend for the program; AR/CTL can reattach the program and continue processing from the last checkpoint, or you can restart the program yourself. If a program issues an IMS-format rollback call in an environment where IMS is not present, results are unpredictable.

If an application program is using the DSN attach method, SQL rollbacks are not supported. The application program can issue an SQL rollback if the program is using the CAF attach method; however, only the DB2 data will be rolled back in response to the SQL rollback. AR/CTL does not reposition sequential, VSAM, or ASAM files in response to an SQL rollback, and staged output is not affected by the SQL rollback.

Truncated blocks

The nature of checkpoint/restart support is that all output data held in storage is forced to be written at a checkpoint whether the block is full or not full. These truncated blocks might cause a significant increase in the total size of output data sets. You can use checkpoint pacing to prevent an excessive number of truncated blocks from occurring.

Performance

You might notice a difference in the performance of a VSAM-only application program after you implement checkpoint/restart support, file sharing support, or both. To provide this support, AR/CTL for VSAM must perform a substantial amount of work on behalf of the application program. This work includes logging changes, locking records accessed through file sharing, and tracking program progress during execution. This work is equivalent to the work that a database management system, such as IMS or DB2, performs for an application program.
Stream-oriented I/O

The repositioning performed by AR/CTL on supported data sets is based on data set records or blocks. If the application program reads or writes the data one character at a time (with a stream-oriented I/O technique), AR/CTL cannot provide repositioning for the data set.

Considerations for other software products

Consider these issues when you use AR/CTL with other software products.

Job scheduling and job restart packages

AR/CTL works transparently with job scheduling packages. The focus of these packages is on cleanup of the system environment after an abend and resubmission of the job at the proper step. The focus of AR/CTL is on recreating the application environment and repositioning sequential files for application restart.

CASE tools

AR/CTL works with most MVS-based computer assisted software engineering (CASE) tools. If the application program cannot use the Automatic Restart and Automatic Checkpoints options, the CASE tool must allow you to insert the AR/CTL calls at the proper places in the application program and to predict where the data will be defined in working storage so that this working storage can be identified and saved during the checkpoint processing. If you need information about implementing AR/CTL in an application program that you are developing with a CASE tool, contact BMC Software Product Support.

DASD compression products

AR/CTL is compatible with the BMC Software DATA PACKER products. If you have any concerns about compatibility between AR/CTL and a data compression product, contact BMC Software Product Support.

Diagnostic aid products

At abnormal termination of an application program job step, AR/CTL must close all data sets that are under AR/CTL control. Data sets that might appear to be open to the application program will not appear in a list of open data sets, such as list data sets produced by diagnostic aid products.
Non-IBM software

During the execution of some non-IBM software packages, AR/CTL might initialize when AR/CTL participation is not wanted. If this situation occurs, the easiest solution is to add a program exclusion record for the application program.

Products for application testing

AR/CTL is disabled if the execution environment includes participation of a product that is designed to enhance the application program testing environment. These products might interfere with AR/CTL termination functions. AR/CTL is meant for use in application programs that execute correctly without AR/CTL.

BMC Software IMS database utilities products

AR/CTL does not provide checkpoint/restart services or data services during execution of BMC Software database utilities products. Most of these products use enhanced methods for I/O processing and record management; AR/CTL supports standard methods only. AR/CTL supports the CONCURRENT REORG for IMS product for suspension of BMP activity.

Operational considerations

Consider these points when you use AR/CTL in your operations environment.

Shared REGISET

You can restart a job step on a different MVS system than the system where the job step executed previously; however, both systems must use the same REGISET. You can use a different REGISET (and subsystem) on different MVS systems; using a different REGISET ensures that active job steps executing in one system are not confused with active job steps executing in another system. If two or more MVS systems are sharing a REGISET, ensure that one or more execution qualifiers distinguish job steps executing in a system from those executing in all other systems that share the REGISET.

Uncataloged data sets in STEPLIB

For an application program execution that uses AR/CTL, uncataloged or temporary data sets that are included in the STEPLIB concatenation are not supported. AR/CTL products are excluded from execution when the STEPLIB contains an uncataloged or temporary data set.
ABEND, STEP processing

If you are using AR/CTL for VSAM and an application program is updating any VSAM files sequentially (deferred buffer writes), data integrity is compromised when the application task is canceled or abended with the STEP parameter of the ABEND macro in an Assembler language routine.

The STEP parameter causes all tasks in the job step to terminate abnormally. AR/CTL, as one of the tasks that abends, might not receive control to complete termination tasks and, therefore, can leave the active job step member in ACTIVE status. Before the job step can be restarted, you might need to reset the active job step member to ABENDED status.

Stop region and OS cancel commands

To avoid potential checkpoint-in-doubt situations, use the AR/CTL OCI STOP command rather than the IMS stop region command or the OS cancel command. AR/CTL maintains checkpoint control information to track the progress of the checkpoint process. At the beginning of the checkpoint process, AR/CTL updates this information to indicate that the checkpoint process has started. When the checkpoint process is complete, AR/CTL updates this information to indicate that the process is complete and to identify which checkpoint to use for restart. If an event (such as a power outage, an IMS stop region command, or an OS cancel) prevents AR/CTL from updating the checkpoint control information, a checkpoint-in-doubt situation can result, and you must manually determine which checkpoint to use for restart. If you use the OCI STOP command, AR/CTL is able to update the checkpoint control information and avoid the in-doubt situation.
# Getting started

This chapter tells you how to start using APPLICATION RESTART CONTROL (AR/CTL). It introduces the tasks you must perform to use AR/CTL and summarizes the steps of those tasks. This chapter contains the following information:

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>64</td>
</tr>
<tr>
<td>Getting started with AR/CTL implementation</td>
<td>65</td>
</tr>
<tr>
<td>Making implementation decisions</td>
<td>65</td>
</tr>
<tr>
<td>Setting up AR/CTL records</td>
<td>69</td>
</tr>
<tr>
<td>Setting up AR/CTL with application program changes</td>
<td>70</td>
</tr>
<tr>
<td>Setting up AR/CTL with JCL changes</td>
<td>71</td>
</tr>
<tr>
<td>Performing other setup tasks</td>
<td>72</td>
</tr>
<tr>
<td>Executing with AR/CTL</td>
<td>74</td>
</tr>
<tr>
<td>Handling problems</td>
<td>74</td>
</tr>
<tr>
<td>Getting started with AR/CTL setup</td>
<td>75</td>
</tr>
<tr>
<td>Creating an environment registration record</td>
<td>75</td>
</tr>
<tr>
<td>Creating a program registration record</td>
<td>76</td>
</tr>
<tr>
<td>Creating a program exclusion record</td>
<td>78</td>
</tr>
<tr>
<td>Creating a processing options record</td>
<td>78</td>
</tr>
<tr>
<td>Setting up cursor repositioning options</td>
<td>79</td>
</tr>
<tr>
<td>Setting up reattach options</td>
<td>80</td>
</tr>
<tr>
<td>Setting up program exception class options</td>
<td>80</td>
</tr>
<tr>
<td>Setting up SQL return code options</td>
<td>81</td>
</tr>
<tr>
<td>Setting up dynamic allocation options</td>
<td>82</td>
</tr>
<tr>
<td>Setting the current shift identifier</td>
<td>83</td>
</tr>
<tr>
<td>Setting up checkpoint pacing</td>
<td>84</td>
</tr>
<tr>
<td>Getting started with AR/CTL operations</td>
<td>85</td>
</tr>
<tr>
<td>Using the operator communications interface</td>
<td>85</td>
</tr>
<tr>
<td>Using statistical reports</td>
<td>87</td>
</tr>
<tr>
<td>Working with active records</td>
<td>87</td>
</tr>
<tr>
<td>Excluding or disabling AR/CTL</td>
<td>88</td>
</tr>
<tr>
<td>Exclusion methods</td>
<td>88</td>
</tr>
<tr>
<td>Exclusion considerations</td>
<td>89</td>
</tr>
<tr>
<td>Testing the application program with AR/CTL</td>
<td>91</td>
</tr>
<tr>
<td>Using AR/CTL DD statements</td>
<td>92</td>
</tr>
<tr>
<td>Using the IMS CHKPID parameter</td>
<td>96</td>
</tr>
<tr>
<td>Setting up DB2 connections</td>
<td>98</td>
</tr>
</tbody>
</table>
Overview

Table 1 provides an overview of the tasks for using AR/CTL. Most of these tasks are discussed briefly in this chapter.

Table 1  AR/CTL tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Comments</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>install AR/CTL, and process CPU ID password</td>
<td>Use the Installation System (the Installation System was formerly known as the OS/390® and z/OS Installer) to install AR/CT.</td>
<td>required</td>
<td>Installation System User Guide</td>
</tr>
<tr>
<td></td>
<td>Use the Installation System to process a CPU ID password that authorizes the product to run on your processor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>configure AR/CTL</td>
<td>Use the AR/CTL Installation System to configure AR/CTL.</td>
<td>required</td>
<td>APPLICATION RESTART CONTROL Configuration Guide</td>
</tr>
<tr>
<td>implement AR/CTL in an application program and environment</td>
<td>You must perform certain tasks to have AR/CTL work with the application program and environment. The tasks to perform depend on: whether the program uses DB2, IMS, or VSAM; which services of AR/CTL you use; and which implementation methods you use.</td>
<td>required</td>
<td>page 65</td>
</tr>
<tr>
<td>set up AR/CTL records</td>
<td>To control and customize AR/CTL processing, you can create and change AR/CTL records in the REGISET. Some records are required and are created during AR/CTL configuration.</td>
<td>required</td>
<td>page 75</td>
</tr>
<tr>
<td>operate with AR/CTL</td>
<td>You can work with the REGISET records that AR/CTL uses to track the status of job steps. You can also use the operational services of AR/CTL during operations.</td>
<td>as needed</td>
<td>page 85</td>
</tr>
<tr>
<td>exclude AR/CTL participation</td>
<td>AR/CTL participation might be undesirable in some cases. You can use any of several methods to prevent AR/CTL participation.</td>
<td>as needed</td>
<td>page 88</td>
</tr>
<tr>
<td>test with AR/CTL</td>
<td>You should test AR/CTL with an application program before the program executes in a production environment with AR/CTL.</td>
<td>recommended before use in production</td>
<td>page 91</td>
</tr>
</tbody>
</table>
Getting started with AR/CTL implementation

The tasks you perform to implement AR/CTL in an application program depend on the application program type (whether it uses IMS, DB2, and/or VSAM) and the AR/CTL services you want to use. The implementation process is organized into these procedures:

- making implementation decisions
- setting up AR/CTL records
- setting up AR/CTL with application program changes
- setting up AR/CTL with JCL changes
- performing other setup tasks
- executing with AR/CTL
- handling problems

**NOTE**

Before you use AR/CTL with an application program in a production environment, you should test the application program with AR/CTL. For more information, see “Testing the application program with AR/CTL” on page 91.

Making implementation decisions

These tasks give you direction for the other tasks in the implementation process:

1. Determine which AR/CTL services you want to implement for an application program.

   For more information, see Chapter 1, “Introduction.”

2. If you want to use checkpoint/restart services, determine which method to use for implementing restart processing; the available methods depend on the type of program. A program can use only one method.

   For more information, see Chapter 3, “Using AR/CTL checkpoint/restart services.”

- An IMS, DB2, VSAM, or QSAM program that is compiled with COBOL might be able to use the Automatic Restart processing options without program changes to implement a restart call. A processing options record that contains suitable option values is required.

- A DB2, VSAM, or QSAM program can issue the AR/CTL restart (AR CXRST) call, or the AR/CTL common call. Program changes are required.
Making implementation decisions

- A program that uses IMS can issue the IMS extended restart (XRST) call or the AR/CTL common call. If the program already issues the IMS XRST call, no change to the call is necessary.

- For subprograms, AR/CTL can restore saved virtual storage areas through the Subprogram Virtual Storage (ARCSPVS) application program interface (API).

You might need to provide some code, such as priming application program variables, to execute when the program is restarted. If the application program already contains restart logic, no change to that logic is necessary.

3 If you want to use checkpoint/restart services, determine which method to use for implementing checkpoint processing; the available methods depend on the type of program. A program can use only one method.

For more information, see Chapter 3, “Using AR/CTL checkpoint/restart services.”

- An IMS, DB2, VSAM, or QSAM program that is compiled with COBOL might be able to use the Automatic Checkpoints options without program changes to implement checkpoint calls. A processing options record that contains suitable option values is required.

- A DB2, DB2/VSAM, or DB2/QSAM program can issue DB2 commit calls. DB2 commit calls trigger AR/CTL checkpoint requests. If the program already contains DB2 commit calls, no changes to those calls are necessary.

- A DB2, DB2/VSAM, VSAM-only, or sequential-file-only program can issue AR/CTL checkpoint (ARCECHK or ARCHKP) calls, or AR/CTL common calls. Program changes are required.

- A program that uses IMS can issue IMS checkpoint (CHKP) calls or AR/CTL common calls. If the program already issues IMS CHKP calls, no changes to those calls are necessary.

- For subprograms, AR/CTL can save virtual storage areas through the ARCSPVS API.

4 If you want to use data services, determine which method to use for implementing the service; the available methods depend on the data service.

For more information, see Chapter 3, “Using AR/CTL checkpoint/restart services.”

- To use sequential file interception, the program issues standard MVS QSAM requests. If the program already issues these requests, no changes to those requests are necessary.
To use local VSAM access services, the program issues standard VSAM requests, which AR/CTL intercepts. If the program already issues these requests, no changes to those requests are necessary.

To use remote VSAM access services, the program issues standard VSAM requests, which AR/CTL intercepts. If the program already issues these requests, no changes to those requests are necessary.

To use ASAM services through the ASAM replacement for GSAM, a program that uses IMS or IMS-compatible calls and structures can issue GSAM-format calls. If the program already issues these calls, no changes to those calls are necessary.

To use ASAM services through the ASAM callable interface, a program that does not use IMS can issue ARCASAM calls.

To use DB2 cursor repositioning services, program exception handling, or SQL return code handling, you create or modify records in the REGISET, as described in “Setting up AR/CTL records” on page 69.

**5** Determine whether the application program must be recompiled (or reassembled). This task is required for an existing application program only if the program was changed.

**6** Determine whether the application program must be relinked. This task is required if any of the following conditions is true:

- Any program changes were made (or the program is new).
- The program uses COBOL, uses DB2, does not use IMS, and is compiled with the NODYNAM option. (An IMS/DB2 program does not need to be relinked.)
- The program uses PL/I, uses DB2, and does not use IMS. (An IMS/DB2 program does not need to be relinked.)

For more information, see Chapter 6, “Changing COBOL application programs,” Chapter 7, “Changing PL/I application programs,” or Chapter 8, “Changing Assembler language application programs.”

**7** Determine whether EXEC statement changes to the JCL are necessary:

- If the application program uses DB2 and is attached to run under any other program or product (except AR/CTL), the JCL must be changed to invoke the program directly.

For more information, see “Setting up AR/CTL with JCL changes” on page 71.
Making implementation decisions

- If the JCL is set up to execute program ARCCTRL (as required for AR/CTL Version 1), the EXEC statement needs no changes. Module ARCCTRL is no longer shipped with AR/CTL Version 2; the BCSS handles execution of program ARCCTRL automatically. (If program ARCCTRL is not found during execution, an error message indicates that the BCSS is not active or the ATTACH function is not enabled.) You can change the EXEC statement to specify the application program name.

8 Determine whether you want to use the program registration method or the AES$ssid DD statement method to provide access to the BCSS:

- To use the program registration method, you create a program registration record or an automatic registration record (which causes a program registration record to be created). During step initialization, the BCSS checks for the presence of an applicable record in the REGISET. If the record is present, AR/CTL products are invoked to participate in the execution. If the record is not present, the program executes without AR/CTL product participation. **BMC Software recommends the program registration method.**

- To use the AES$ssid DD statement method, you insert the AES$ssid DD DUMMY statement in the step JCL; ssid is the four-character subsystem ID of the specific BCSS to use during execution. The AES$ssid DD statement method might be useful in some situations but is not recommended for general use.

For more information, see “Using AR/CTL DD statements” on page 92.

9 Optional. Determine whether you want to use the AR/CTL Status Check utility.

The AR/CTL Status Check utility can ensure that AR/CTL products, AR/CTL components of the BCSS, and functions of the AR/CTL components are available for application program execution. You can implement the utility by changing the JCL to execute the application program or by changing the application program itself.

The AR/CTL Status Check utility can also test whether AR/CTL is active in the application program address space. If AR/CTL is not active in the address space, the utility can return a specified return code or can issue an abnormal termination with a specified abend completion code and reason code.

For more information, see “Using the AR/CTL Status Check utility” on page 100 and see the *APPLICATION RESTART CONTROL Administrator Guide*.

10 For an application program that uses DB2, determine whether you must set up DB2 connections and, if so, which method to use. If the DB2 program also uses IMS or AR/CTL Version 1, these connections have already been set up. If the DB2 application program does not use IMS or AR/CTL Version 1, you can use either of the following methods (as discussed in “Setting up DB2 connections” on page 98):
You can set the connection information in a non-IMS program registration record. This method is the easiest unless you already use one of the other methods.

You can create a subsystem member (SSM) that provides DB2 connection information at job step execution.

Setting up AR/CTL records

To create or modify the AR/CTL records that are required for AR/CTL, perform these steps as applicable:

1. Register the environment for AR/CTL participation. Create an environment registration record that points to the AR/CTL load module libraries you want to use.

   For more information, see “Creating an environment registration record” on page 75.

2. Create program registration records as necessary, depending on the implementation decisions you have made:

   - For a DB2 or VSAM program, you can create a non-IMS program registration record or an automatic registration record.
     
     For more information, see “Creating a program registration record” on page 76 or “Creating an automatic registration record” on page 77.

   - For an IMS program, create an IMS program registration record.

     For more information, see “Creating a program registration record” on page 76.

3. Check the values of the various AR/CTL processing options that will be used for the application program.

   For more information, see “Creating a processing options record” on page 78.

4. If applicable, check the values of the DB2 cursor repositioning options. If necessary, adjust the values or create a record to contain the values appropriate for the application program.

   For more information, see “Setting up cursor repositioning options” on page 79.

5. If applicable, check the values of the application reattach options. If necessary, adjust the values or create a record to contain the values appropriate for the application program.
Setting up AR/CTL with application program changes

Depending on the methods you choose for implementing AR/CTL services and participation, perform the following steps as applicable.

For more information, see Chapter 6, “Changing COBOL application programs,” Chapter 7, “Changing PL/I application programs,” or Chapter 8, “Changing Assembler language application programs.”

1 If you are using one or more implementation methods that require program changes, change the application program.

2 If you are using the AR/CTL Status Check utility (program AESUVBCS) and you want to call it from the application program, change the application program.

For more information, see the APPLICATION RESTART CONTROL Administrator Guide.

For more information, see “Setting up reattach options” on page 80.

6 If applicable, check the values of the program exception class options. If necessary, adjust the values or create a record to contain the values appropriate for the application program.

For more information, see “Setting up program exception class options” on page 80.

7 If applicable, check the values of the SQL return code class options. If necessary, adjust the values or create a record to contain the values appropriate for the application program.

For more information, see “Setting up SQL return code options” on page 81.

8 AR/CTL requires dynamic allocation records for some AR/CTL data sets, including the checkpoint data set. In most cases, the global records created during configuration of AR/CTL are suitable; however, you might want to check that the values are appropriate for the needs of a particular application program.

For more information, see “Setting up dynamic allocation options” on page 82.

9 If you want to use AR/CTL checkpoint pacing, you can check and modify the current shift identifier and checkpoint pacing options.

For more information, see “Setting the current shift identifier” on page 83 and “Setting up checkpoint pacing” on page 84.
3 If you are using the AR/CTL Status Check utility (program ARCACTIV), change the application program.

For more information, see “Using the AR/CTL Status Check utility” on page 100.

4 If you made program changes, recompile the program.

5 If you made program changes or if the program uses the DB2 language interface module in a statically linked non-IMS environment (such as with the COBOL NODYNAM option or in a PL/I environment), relink the program. Any DB2 language interface modules (DSNALI, DSNELI, or DSNHLI) must be replaced with the AR/CTL language interface module (ARCLI000). The AR/CTL load library must be ahead of the DB2 library in the SYSLIB concatenation.

If the COBOL program uses an explicit connection to the DB2 call attach facility (CAF), does not use IMS, and the DB2 language interface module is statically linked, you must recompile the program with the DYNAM option.

6 If the program uses DB2 and you made program changes, perform normal DB2 bind processing for the application program; no special application bind processing is required.

---

Setting up AR/CTL with JCL changes

Depending on the type of application program and on the methods you choose to implement AR/CTL participation, perform these steps as applicable:

1 If the program uses DB2 and is attached to run under any other program or product (except AR/CTL), change the JCL to invoke the program directly.

Many batch DB2 programs that use the TSO attach method execute program IKJEFT01 and provide the DB2 attach and execute commands with the SYSTSIN DD statement. The following example shows this method of executing a DB2 program without AR/CTL:

```
//S1      EXEC PGM=IKJEFT01
//SYSTSIN DD   *
   DSN SYSTEM (DB2P)
   RUN PROGRAM (MYPROG)
END
/*
// ADDITIONAL DD CARDS......
```
Performing other setup tasks

To use AR/CTL for a DB2 program that is set up to use TSO attach (or any product that performs the attach on behalf of the application program), you must modify the EXEC statement in the JCL to execute the DB2 program in place of IKJEFT01. You can provide the DB2 attach information in the program execution record or with the ARCSSMLB DD statement, as explained in “Setting up DB2 connections” on page 98.

DB2 programs that used AR/CTL Version 1 execute program ARCCTRL. You do not need to change this program name.

2 If you want to use the AESssid DD statement, insert the AESssid DD DUMMY statement into the step JCL, where ssid is the four-character subsystem ID of the BCSS to use during execution.

For more information, see “Using AR/CTL DD statements” on page 92.

NOTE

Even if you code the AESssid DD statement, you must still provide a program registration record that applies to the job step (unless the program name is DFSRRC00 or ARCCTRL).

3 If you are using the AR/CTL Status Check utility (program AESUVBCS) to ensure that AR/CTL products, components, and functions are available, and if you want to execute this utility as a stand-alone program, insert the utility job step ahead of the application program execution job step.

For more information, see the APPLICATION RESTART CONTROL Administrator Guide.

Performing other setup tasks

Depending on the AR/CTL services and options you want to use, perform these tasks as applicable:

1 If want to use data services, you can set up default and specific data set option members (also known as file characteristics blocks or FCBs). You can force use of a particular set of data set options members by setting up a program options member (also known as an application specification block or ASB), or you can use the AUTO$ASB method to use default data set options.

For more information, see the APPLICATION RESTART CONTROL Reference Manual.
If you use the AUTO$ASB method and you want to bypass use of AR/CTL data services for one file but use them for other files, you can create a specific FCB for the file. The FCB member name is the ddname of the file. To provide access to the FCB during application program execution, you can add the ARC DD statement to the job step or create a dynamic allocation record for the job step (the record specifies the data set name of the library that contains the FCB). During execution, AR/CTL first searches for a specific FCB for each file. If a specific FCB exists, AR/CTL uses the options in the specific FCB for that file only. If a specific FCB does not exist for the file, AR/CTL automatically generates one for the file.

2 If the application program uses remote VSAM access (file sharing), you can verify that the necessary AR/CTL enhanced processing function is installed in the MVS system in which you want to use remote VSAM access. Also, you can verify that the CICS environment and multiregion operations (MRO) connection information is set up for remote VSAM access services. Typically, these tasks are performed during AR/CTL customization.

For more information, see the APPLICATION RESTART CONTROL Configuration Guide.

3 If the program is a batch DLI application program and you want to use the application reattach options of AR/CTL, set up the BATCH CONTROL FACILITY (BCF) component of AR/CTL to perform dynamic backout of application program changes.

For more information, see the APPLICATION RESTART CONTROL Reference Manual: BATCH CONTROL FACILITY.

4 If you want to include AR/CTL DD statements in your application program JCL, change the JCL.

For more information, see “Using AR/CTL DD statements” on page 92.

5 If the application program uses DB2 but not IMS and you do not want to use a non-IMS program registration record to provide DB2 connection information, set up the DB2 connection information in the subsystem member.

For more information, see “Setting up DB2 connections” on page 98.
Executing with AR/CTL

After you have set up the required AR/CTL records, made program and JCL changes (if applicable), and performed other setup tasks as necessary, you are ready to execute an application program with AR/CTL.

1. Submit the application program JCL for execution as usual.

2. During application program execution, you can use the operator communications interface (OCI) to start and stop enhanced call tracing, terminate the execution early, and get snap dumps of application storage.

   For more information, see “Using the operator communications interface” on page 85.

3. You can view statistical information stored in the history data set.

   For more information, see “Using statistical reports” on page 87.

4. If the execution does not complete successfully, identify the cause and solution of the problem and take action to correct the problem. Then the usual action is to resubmit the job step that failed. You can view and modify information about the execution in the active records for the execution.

   For more information, see “Working with active records” on page 87.

Handling problems

Standard operations for AR/CTL are highly automated; however, certain non-standard situations can require manual intervention. These situations include abnormal termination of non-unique job steps, damage or destruction of essential AR/CTL components, and checkpoint in-doubt conditions. Chapter A, “Solving problems” describes how to handle common non-standard situations you might encounter.

A BMC Software product support representative for AR/CTL is always on call to answer questions and solve problems. You can contact the product support representative by calling the support number listed on the back of the title page. You can help the product support representative deliver a faster response to your question or problem if you gather certain information before you call. “Contacting BMC Software Product Support” on page 265 describes the types of helpful information about your environment, application program, and AR/CTL maintenance level.
Getting started with AR/CTL setup

To set up the AR/CTL environment to work with your application programs, create and update records in the REGISET. Use the ISPF interface for these tasks:

- create environment registration records
- create program registration records
- create program exclusion records
- set the current shift identifier
- set checkpoint pacing options
- set processing options at several levels
- set cursor repositioning options
- set reattach class options
- set program exception class options
- set SQL return code class options
- set dynamic allocation information at several levels

Creating an environment registration record

To provide access to AR/CTL execution modules and services, you must create one or more environment registration records in the REGISET. Different records are used for non-IMS environments and IMS environments. A non-IMS environment is any environment where IMS is not active or where application programs that execute in the environment do not use IMS-type calls and structures. An IMS environment is any environment where IMS is active or where IMS-compatible application programs use IMS-type calls and structures but IMS is not active.

To create an environment registration record, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Administrator Guide.

1. Access the Application Enhancement Series primary menu, and select option 1 (AES records).

2. On the Application Enhancement Series Records panel, select option 2 (Non-IMS Environment registration) or option 6 (IMS Environment registration).

3. On the Limit List of Records panel, type or verify asterisks in all fields, and press Enter.

4. On the List Records panel, enter the ADD command.
Creating a program registration record

5 On the Add Record panel, type the qualifiers to use in the record key, type the requested information (if applicable), and press Enter to validate the information.

6 Enter the END command to save the record and exit from the panel.

Creating a program registration record

To set up AR/CTL to participate in application program execution without JCL changes, you can create one or more program registration records in the REGISET. Different records are used for non-IMS programs and IMS programs. For non-IMS programs, you might be able to create one or more automatic registration records instead of non-IMS program registration records, as described in “Creating an automatic registration record” on page 77.

To create a program registration record, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Administrator Guide.

1 Access the Application Enhancement Series primary menu, and select option 1 (AES records).

2 On the Application Enhancement Series Records panel, select option 3 (Non-IMS Program manual registration) or option 7 (IMS Program registration).

3 On the Limit List of Records panel, type or verify asterisks in all fields, and press Enter.

4 On the List Records panel, enter the ADD command.

5 On the Add Record panel, type the qualifiers to use in the record key, type the requested information (if applicable), and press Enter to validate the information.

NOTE

To allow a job step that abends on one IMS system to be restartable on a different IMS system, set the IMSID to wildcard characters in the program registration record. Although the IMSID is not in the key of the restart control record, the IMSID is in the key of the program registration record that identifies the application program for AR/CTL participation.

6 Enter the END command to save the record and exit from the panel.
Creating an automatic registration record

To set up AR/CTL to participate in non-IMS application program execution without JCL changes, you might be able to create one or more automatic registration records instead of non-IMS program registration records. The AR/CTL application supervisor can use an automatic registration record to create non-IMS program registration records automatically.

To create an automatic registration record, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Administrator Guide.

1. Access the Application Enhancement Series primary menu, and select option 1 (AES records).

2. On the Application Enhancement Series Records panel, select option 5 (Non-IMS Program automatic registration).

3. On the Limit List of Records panel, type or verify asterisks in all fields, and press Enter.

4. On the List Records panel, enter the ADD command.

5. On the Add Record panel, type the qualifiers to use in the record key, and press Enter.

6. On the Add Record panel, type the qualifiers to use in the record key, and press Enter.

7. On the QUICKSTART DSNAME Criteria (1 of 8) panel and following panels, enter the criteria to use for determining whether a program is eligible for automatic registration and the parameters to store in an automatically created non-IMS program registration record.

   A program can be registered to use the AR/CTL for DB2 batch attachment facility only.

8. Enter the END command to save the record and exit from the panel.
Creating a program exclusion record

To prevent AR/CTL products from participating in application program execution, you can create one or more program exclusion records in the REGISET. Different records are used for non-IMS programs and IMS programs.

To create a program exclusion record, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Administrator Guide.

1. Access the Application Enhancement Series primary menu, and select option 1 (AES options).

2. On the Application Enhancement Series Records panel, select option 4 (Non-IMS Program exclusion) or option 8 (IMS Program exclusion).

3. On the Limit List of Records panel, type or verify asterisks in all fields, and press Enter.

4. On the List Records panel, enter the ADD command.

5. On the Add Record panel, type the qualifiers to use in the record key, and press Enter to validate the information.

6. Enter the END command to save the record and exit from the panel.

Creating a processing options record

AR/CTL processing options allow you to customize the use of AR/CTL features and functions in your environment. AR/CTL merges the applicable processing options records from the REGISET at execution. Options set in a more specific record override options set in a less specific record.

To create a new processing options record, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Reference Manual.

1. Access the APPLICATION RESTART CONTROL primary menu, and select option 5 (Processing options).

2. On the Limit List of Records panel, type or verify asterisks in all fields, and press Enter.

3. On the List Records panel, enter the ADD command.
4 On the Add Record panel, type the qualifiers to use in the record key, and press Enter.

5 For each option field on the Processing Options panel, you can type one of the values listed to the right of the option. You can type ? to display the help panel with more information about the option; on the help panel, you can type the option value you want to use.

6 Press Enter to validate the information. To continue to the next page, enter the DOWN command. Enter the END command to save the record and exit from the panel.

Setting up cursor repositioning options

AR/CTL can perform cursor repositioning for a DB2 program—reestablish the program’s position within the cursor automatically at restart of the application program execution.

NOTE

DB2 cursor repositioning is not available when the cursor is a scrollable rowset cursor.

To invoke cursor repositioning during application program execution, change (or verify) the values in the most specific processing option record that applies to the job step. Set the Automatic DB2 Cursor Repositioning option to Y, and set the Action for No Repositioning Record option to the action to take if a cursor does not have a matching repositioning record. Then create a cursor repositioning record, which contains cursor repositioning options for a particular DB2 cursor, for each DB2 cursor to be repositioned.

To create a cursor repositioning record, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Reference Manual.

1 Access the APPLICATION RESTART CONTROL primary menu, and select option 6 (Cursor repositioning options) or option 7 (Long cursor repositioning opts).

2 On the Limit List of Records panel, type or verify asterisks in all fields, and press Enter.

3 On the List Records panel, enter the ADD command.

4 On the Add Record panel, enter the qualifiers for the record you want to add.
Setting up reattach options

AR/CTL can perform application reattach for a program—automatically restart an application program after an abend without ending the current job step—depending on the reattach conditions you have defined.

To invoke application reattach during application program execution, change (or verify) the values in the most specific processing option record that applies to the job step. Set the Reattach Application Program option to Y, and set the Reattach Class Name option to the name of the reattach class record you will create. Then create a reattach class record that contains the reattach options you want to use for the reattach process.

To create a reattach class record, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Reference Manual.

1. Access the APPLICATION RESTART CONTROL primary menu, and select option 8 (Reattach options).

2. On the List Records panel, enter the ADD command.

3. On the Add Record panel, enter the name of the reattach class you want to define.

4. For each option field on the Reattach Options panel, type the values you want to use. Field-level help with ? is not available on these panels.

5. Press Enter to validate the information. Enter the END command to save the record and exit from the panel.

Setting up program exception class options

AR/CTL can intercept and attempt to handle application program exception conditions that usually result in a system 0Cx-type abend. This program exception handling might allow application processing to run to completion, allowing you to correct the problem after execution completes successfully.
To set up program exception handling, create or update a processing option record that applies to the application program execution. In this processing option record, set the Intercept Program Exceptions option to Y. Set the Program Exception Class Name option to the name of the program exception class record that defines (or will define) the program exception class options you want to use. Then create a new program exception class record (or select an existing record) that contains the program exception class options you want to use. Make sure that the class name of the program exception class record is the same as the value of the Program Exception Class Name processing option.

To create a program exception class record, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Reference Manual.

1. Access the APPLICATION RESTART CONTROL primary menu, and select option 12 (Program exception class options).

2. On the List Records panel, enter the Add command.

3. On the Add Record panel, enter the name of the program exception class you want to define.

4. For each option field on the Program Exception Class Options panels, type the values you want to use. Field-level help with ? is not available on these panels.

5. Press Enter to validate the information. Enter the END command to save the record and exit from the panel.

### Setting up SQL return code options

AR/CTL can intercept a defined SQL return code received during application program processing and issue a defined user abend code and reason code. This abend code and reason code can be defined as a combination that is eligible for application reattach processing.

To set up SQL return code handling, create or update a processing option record that applies to the application program execution. In this processing option record, set the Intercept SQL Return Codes option to Y. Set the SQL Return Code Class Name option to the name of the SQL return code class record that defines (or will define) the SQL return code options you want to use. Then create a new SQL return code class record (or select an existing record) that contains the SQL return code class options you want to use. Make sure that the class name of the SQL return code class record is the same as the value of the SQL Return Code Class Name option.
To create an SQL return code class record, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Reference Manual.

1 Access the APPLICATION RESTART CONTROL primary menu, and select option 13 (SQL return code options).

2 On the List Records panel, enter the ADD command.

3 On the Add Record panel, enter the name of the SQL return code class you want to define.

4 For each option field on the SQL Return Code Class Options panel, type the values you want to use. Field-level help with ? is not available on these panels.

5 Press Enter to validate the information. Enter the END command to save the record and exit from the panel.

### Setting up dynamic allocation options

During application program execution, AR/CTL must dynamically allocate the checkpoint data set from a dynamic allocation record (ddname ARCCHKP) in the REGISET. AR/CTL can dynamically allocate the trace data set from a dynamic allocation record named ARCDLTRC. If the program uses local VSAM access services, AR/CTL can dynamically allocate the VSAM dynamic backout log. AR/CTL can also dynamically allocate the data set/program options libraries (ddname ARC), the AR/CTL external subsystem definition library (ddname ARCSSMLB), and the AR/CTL history data set (ddname ARCHIST) if you provide dynamic allocation records for them.

At execution, AR/CTL uses the dynamic allocation information in the most specific record that applies to the job step; the dynamic allocation information is not merged.

To create a new dynamic allocation record for ddname ARCCHKP, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Reference Manual.

1 Access the APPLICATION RESTART CONTROL primary menu, and select option 9 (Dynamic allocation options).

2 On the Select Dynamic Allocation Record DDname panel, select option 1 (ARCCHKP).

3 On the Limit List of Records panel, type or verify asterisks in all fields, and press Enter.
4 On the List Records panel, enter the **ADD** command.

5 On the Add Record panel, type the qualifiers to use in the record key, and press **Enter**.

6 For each option field on the Skeleton Options panel, you can type one of the values listed to the right of the option. Or you can type ? to display the help panel with more information about the option; on the help panel, you can type the option value you want to use.

7 Press **Enter** to validate the information. To continue to the next page, enter the **DOWN** command. Enter the **END** command to save the record and exit from the panel.

---

**Setting the current shift identifier**

AR/CTL allows you to use different values for the checkpoint pacing criteria at different times of the day. During the prime shift, for example, you might want to allow checkpoints for batch jobs to occur more frequently than on other shifts to reduce resource contention with online processing. During the third shift, you might want to allow fewer checkpoints to occur because performance of batch jobs might be more critical than concern for online response.

To determine the current shift, AR/CTL uses a current shift identifier record in the REGISET. You can create a new current shift identifier record with the ISPF interface, and you can set the current shift identifier to a different shift with the ISPF interface or through a batch job.

**ISPF interface**

To change a current shift identifier record or add a new record with the AR/CTL ISPF interface, perform the following steps; all steps are required. For details about this task, see the *APPLICATION RESTART CONTROL Reference Manual*.

1 Access the APPLICATION RESTART CONTROL primary menu, and select option 2 (Current shift identifier).

2 On the Limit List of Records panel, type or verify asterisks in all fields, and press **Enter**.

3 On the List Records panel, enter the **ADD** command (if you want to create a new record), or select an existing record.

4 If you entered the **ADD** command, type the qualifiers to use in the record key on the Add Record panel, and press **Enter**.
5 On the Current Shift Display/Update panel, type the shift number you want to use.

6 Press Enter to validate the information. Enter the END command to save the record and exit from the panel.

**Batch job**

To change the shift with a batch job, use the sample JCL in member #ARCURSH of the sample library. Use the shift parameter of the PARM parameter on the EXEC statement to control the shift number. You can set up this batch job to execute on a scheduled basis.

**Setting up checkpoint pacing**

AR/CTL can pace checkpoints for an application program—allow checkpoint processing to occur or return control to the application program without performing checkpoint processing, depending on whether the checkpoint meets the checkpoint conditions you have defined. Checkpoint pacing is defined and performed externally to the application program. AR/CTL makes the pacing decision dynamically, based on real-time factors such as the characteristics of the program execution, the number of calls, and the time between checkpoints. Checkpoint pacing criteria can change dynamically while the program is executing.

Checkpoint pacing allows the application program to issue checkpoints each time the program reaches a logical point during processing, without concern for operational issues such as checkpoint overhead.

To invoke checkpoint pacing during application program execution, change (or verify) the values in the most specific processing option record that applies to the job step. Set the Use Checkpoint Pacing option to Y and set the Pacing Class Name option to the name of the pacing class record you will create. Then create a checkpoint pacing record, which contains pacing options to define the pacing conditions to use.

To create a checkpoint pacing record, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Reference Manual.

1 Access the APPLICATION RESTART CONTROL primary menu, and select option 3 (Checkpoint pacing options).

2 On the List Records panel, enter the ADD command.

3 On the Add Record panel, enter the name of the pacing class you want to define.
For each option field on the Pacing Class Options panel, type the values you want to use. Field-level help with ? is not available on these panels.

Press Enter to validate the information. To continue to the next page, enter the DOWN command. Enter the END command to save the record and exit from the panel.

**Getting started with AR/CTL operations**

This section summarizes how to access and work with active records, checkpoint pacing, reports, and the functions of the OCI.

**Using the operator communications interface**

You can use the OCI to invoke these AR/CTL actions during processing:

- invoke early termination support
- request a snap dump of application program storage
- begin and end enhanced call tracing

**Early termination support**

Early termination support through the OCI allows you to terminate an application job after the next checkpoint or terminate the job immediately after the next call. Termination after a checkpoint requires no backout because the checkpoint signifies that the changes are committed. Immediate termination requires backout of uncommitted changes. In both cases, all restart information is left intact and ready for restarting the job at the last completed checkpoint.

For more information, see “Using early termination support” on page 145.

**On-demand snap dump**

You can use the OCI to obtain a standard MVS snap dump of application program storage. You can use the snap dump to analyze application program problems. The snap dump does not cause program termination.

For more information, see “Using on-demand snap dumps” on page 149.
Enhanced call tracing

AR/CTL provides enhanced call tracing to help you develop, test, debug, and audit application programs. Enhanced call tracing provides essential information about the IMS DL/I, IMS Fast Path, and SQL calls issued by an application program. You can begin enhanced call tracing with the Enhanced Call Trace option or OCI commands. You can end enhanced call tracing only with OCI commands. AR/CTL writes tracing information to the trace data set (ddname ARCDLTRC). You can allocate this data set in the job step JCL, or AR/CTL can allocate it dynamically if you set up the ARCDLTRC dynamic allocation record. You can use the Trace Format utility to create reports from the trace data.

For more information, see Chapter 5, “Using AR/CTL operational services.”

Using OCI functions

The OCI consists of a set of MVS MODIFY commands issued from the MVS system console or on the console log. These commands are valid:

SET TRACE ON {DLI | SQL | ALL} [START n] [FOR n]
Begin tracing. You can include the keywords shown in braces to further control tracing: limit tracing to a specific subsystem, start tracing after the application program issues the specified number of calls, and continue tracing for the specified number of calls.

DISPLAY TRACE
Display the current status of the enhanced call tracing process.

SET TRACE OFF
End tracing.

STOP
Abnormally terminate the application program when the current or next program checkpoint completes.

STOP IMMEDIATE
Abnormally terminate the application program when the current program call completes or the next program call is issued.

SNAP
Produce an MVS standard snap dump of application program storage after the next AR/CTL call.
Using statistical reports

When AR/CTL is active during an application program job step, it automatically produces reports about many activities that occurred during the execution. AR/CTL writes the reports to the ARCSTATS data set. AR/CTL also writes the set of reports to the history data set if the value of the Number of History Sets processing option is greater than zero.

To view reports stored in the history data set through the AR/CTL ISPF interface, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Reference Manual.

1. Access the APPLICATION RESTART CONTROL primary menu, and select option 4 (Reports).
2. On the Report Options panel, specify how you want to view the reports.
3. On the List Records panel, select the set of reports you want to view by entering S in the choice entry field to the left of the report set. AR/CTL displays the report in ISPF Browse mode.
4. Enter the END command to return to the List Records panel.

Working with active records

AR/CTL creates a restart control record in the REGISET to track the status of an application program job step that has begun execution but has not completed successfully (because it is still executing or has failed). If the application program uses native VSAM data sets, AR/CTL also creates a VSAM recovery record to track the status of the application program’s VSAM processing. You can perform several actions on active records.

To work with active records, perform the following steps; all steps are required. For details about this task, see the APPLICATION RESTART CONTROL Reference Manual.

1. Access the APPLICATION RESTART CONTROL primary menu, and select option 1 (Active jobsteps).
2. On the Select Active Record Type panel, select option 1 (Restart control records).
3. On the Limit List of Records panel, type or verify asterisks in all fields, and press Enter.
4. On the List Records panel, type one of the following action codes next to the restart control record that you want to select:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Delete this restart control record.</td>
</tr>
<tr>
<td>P</td>
<td>Print this restart control record.</td>
</tr>
<tr>
<td>S</td>
<td>Display this restart control record.</td>
</tr>
<tr>
<td>M</td>
<td>Modify the status of this restart control record.</td>
</tr>
<tr>
<td>C</td>
<td>Display and change information about the restart data set named in the restart control record.</td>
</tr>
<tr>
<td>K</td>
<td>Display the restart data set data.</td>
</tr>
<tr>
<td>O</td>
<td>Set manual restart information.</td>
</tr>
<tr>
<td>J</td>
<td>Submit the job step named in the restart control record for restart with the submit restart exit.</td>
</tr>
</tbody>
</table>

5. Press **Enter**. Depending on the action code, AR/CTL displays a view-only panel or a panel that allows you to update the record.

6. On a panel that allows you to update the record, press **Enter** to validate the information. To continue to the next page of a panel, enter the **DOWN** command. Enter the **END** command to save the updates to a record and exit from the panel.

**Excluding or disabling AR/CTL**

This section describes the methods that you can use to prevent AR/CTL from participating in application program execution. You can use these methods to phase in use of AR/CTL after installation, to exclude AR/CTL from executing in job steps that will never need restart, or to disable AR/CTL in emergency situations.

**Exclusion methods**

AR/CTL offers the following exclusion methods. Not all methods are available in all situations.

**NOTE**

The Force AR/CTL Execution option (set at the AR/CTL level) is used only if AR/CTL encounters an initialization failure. It has no effect on intentional methods for deactivating AR/CTL.
Program exclusion record
You can create a program exclusion record for the specific program name. This record excludes all AR/CTL products and the AR/CTL BCF component from participation in program execution.

APPLICATION RESTART CONTROL Active option
You can set the APPLICATION RESTART CONTROL Active processing option to N at the appropriate level.

Bypass Execution DDname option
You can identify a ddname with the Bypass Execution DDname option and include this ddname in the job step JCL. The presence of this ddname deactivates AR/CTL for the job step.

Security check for AR/CTL participation
You can use AR/CTL external security to prevent AR/CTL participation for applications executing with a particular ARCID. If access is allowed or if a profile name is not found, AR/CTL participates if all other participation criteria allow participation. If access is denied, AR/CTL does not participate.

Exclusion considerations
If the program does not use any AR/CTL services, if you have not changed the program or the execution JCL to implement AR/CTL, and if you have never linked the program with AR/CTL, you can use the program exclusion record method to disable AR/CTL. BMC Software recommends this method for permanent exclusion.

If the program uses AR/CTL services, if you have changed the program or the execution JCL to implement AR/CTL, or if you have linked the program with AR/CTL, the considerations discussed in this section might apply.

--- WARNING ---
If you temporarily deactivate AR/CTL for a job step that has been using AR/CTL services, you must make sure that the required services are provided in some other way (if necessary) and must make sure that AR/CTL processing will be performed correctly when you reactivate AR/CTL for the application. It is usually safe to reactivate AR/CTL after the job step has completed successfully.

Exclusion for IMS programs
In most IMS application program executions that do not use DB2, AR/CTL operates transparently. You can disable AR/CTL with any of the exclusion methods. No JCL changes are required. If the program contains no AR/CTL-format calls, no program changes are required. If the program contains any AR/CTL-format calls, the program must be changed to remove the AR/CTL calls, then must be recompiled and relinked.
Exclusion considerations

Exclusion for DB2 programs

In an application program execution that uses DB2, AR/CTL might be required to perform services such as connection to the DB2 subsystem and processing of AR/CTL calls. If so, AR/CTL cannot be disabled or excluded until you change the program, the JCL, or both:

- If AR/CTL has been performing the connection to the DB2 subsystem on behalf of the program, you must provide for connection without AR/CTL before you can exclude AR/CTL.

- If the program uses AR/CTL calls (restart, checkpoint, and/or ASAM), you must change the program, recompile it, and relink it before you can exclude AR/CTL.

- If the program uses the AR/CTL Automatic Restart options, the Automatic Checkpoint options or DB2 commits, and QSAM sequential file requests or VSAM requests, no program changes are required before you disable AR/CTL. However, the program might need to be relinked.

- If the application program uses COBOL, if it uses the NODYNAM compiler option, and if it was linked with the AR/CTL DSNHLI module, you must relink the program with your own DSNHLI module before you can disable AR/CTL with any method. (This condition applies for any language when the AR/CTL DSNHLI is included in the load module.)

- If the application program uses COBOL and the DYNAM compiler option, you cannot use the Bypass DD Statement processing option or the APPLICATION RESTART CONTROL Active processing option to disable AR/CTL because the AR/CTL version of the DSNHLI module will always be loaded instead of your own DSNHLI module. You can use the non-IMS program exclusion record to disable AR/CTL because your own DSNHLI module will be loaded instead of the AR/CTL DSNHLI module. (This condition applies for any language that dynamically loads DSNHLI.)

Exclusion for non-IMS, non-DB2 programs

If an application program does not use IMS or DB2 and it uses AR/CTL calls (restart, checkpoint, and/or ASAM), you must change the program, recompile it, and relink it before you can exclude AR/CTL. If the program does not contain AR/CTL-format calls, no program changes, no recompiles, and no relinks are required before you exclude AR/CTL.
Exclusion for programs that use the AES$ssid DD statement

The following shows the methods for implementing and excluding or deactivating AR/CTL interact with the AES$ssid DD statement:

- If an AES$ssid DD statement is present, if a program registration record is present, and if a program exclusion record is present, the exclusion record takes effect: AR/CTL is deactivated.

- If an AES$ssid DD statement is present, if a program registration record is not present, and if a program exclusion record is present, the exclusion record does not take effect: AR/CTL is activated.

- If an AES$ssid DD statement is present and if a bypass DD statement is defined in the processing options and is present, AR/CTL is deactivated.

Testing the application program with AR/CTL

Whether you implement AR/CTL in an existing application program or include AR/CTL in the development of a new program, you should test the program as thoroughly as possible before using it for production work. The following items highlight some important areas to test; however, the list is not necessarily complete:

- Ensure correct repositioning on the databases and data sets after a checkpoint and after restart.

- Ensure that the user areas are saved in the checkpoint record with valid addresses, that all expected saved data areas within working storage are restored at restart, and that the restored data areas are complete.

- Ensure that all of the calls are issued in the expected order, according to application program design.

- BMC Software recommends that the application program issue a checkpoint at every logical unit of work boundary and that you use checkpoint pacing to control the frequency of checkpoints allowed to complete. If you do not use checkpoint pacing, you should check and adjust the frequency of checkpoints.

- Test a variety of abend conditions, such as abends before the first checkpoint call, after the first checkpoint call, and after the second checkpoint call. AR/CTL provides early termination options you can use during testing without changing the program logic.
After your testing is complete, the REGISET Record Copy utility, available through the ISPF interface, can help you transfer records quickly from a test environment to a production environment. The utility can copy a record from a REGISET to the same REGISET with a change to the record ID. It can copy a record to different REGISET with or without a change to the record ID.

For more information, see the APPLICATION RESTART CONTROL Administrator Guide.

Using AR/CTL DD statements

In addition to any DD statements that the application program already uses, the following DD statements are valid in the job step JCL of application programs that use AR/CTL:

AES$ssid DD

Optional if a program registration record applies to the execution; required if you want to use AR/CTL, if no program registration record applies, and if the program name is ARCCTRL or DFSRRC00; ignored if no program registration record applies and the program name is not ARCCTRL or DFSRRC00. Code the ddname as AES$ssid, where ssid is the four-character subsystem ID of the specific BCSS to use during execution. Code the statement as DUMMY. You might want to use the AES$ssid DD statement in the following situations:

- If more than one REGISET contains a program registration record that applies to the program, you can use the AES$ssid DD statement to ensure that the execution uses a specific BCSS.

- For a non-IMS program that used AR/CTL Version 1 (the JCL was changed to execute program ARCCTRL), you can use the AES$ssid DD statement to implement AR/CTL Version 2 without creating a program registration record.

- For IMS programs, you can use the AES$ssid DD statement instead of creating a program registration record.

If the program name is not ARCCTRL or DFSRRC00 and no program registration record applies to the execution, the AES$ssid DD statement is ignored; the execution continues without AR/CTL product participation. If the BCSS specified in the ddname of the AES$ssid DD statement is not active, execution continues without AR/CTL product participation. If more than one AES$ssid DD DUMMY statement is present, execution continues without AR/CTL product participation.
BMC Software recommends the use of program registration records instead of the AES$ssid DD statement.

For more information, see the APPLICATION RESTART CONTROL Administrator Guide.

ARC DD
Required if the application program uses specific program option members (ASBs), specific or default data set option members (also known as file characteristics blocks or FCBs), or both; can be allocated dynamically. Describes the library (or libraries) that contain these members. If the program uses the default ASB (AUTO$ASB) and AR/CTL internal default values for FBCs, this data set is not used.

Data set and program option members are load modules. The library that contains these members must have the following DCB parameters, which are typical of a load module library: DSORG=P0 and RECFM=U.

AR/CTL can allocate this data set dynamically if you provide a dynamic allocation record to define it. If the JCL contains the ARC DD statement and the REGISET contains an ARC dynamic allocation record, the DD statement overrides the dynamic allocation record. For information about how to create dynamic allocation records, see the APPLICATION RESTART CONTROL Reference Manual.

ARV DD
Optional. Use this statement to specify the data set name of the library that contains the CICS APPLID or CICS connection table that you want to work with.

AR/CTL can allocate this data set dynamically if you provide a dynamic allocation record to define it. If the JCL contains the ARV DD statement and the REGISET contains an ARV dynamic allocation record, the DD statement overrides the dynamic allocation record. For information about how to create dynamic allocation records, see the APPLICATION RESTART CONTROL Reference Manual.

AR/CTL uses the following search order for locating the ARVRCON and ARVAPID modules. These steps occur independently for each module.

1. If the ARV DD statement is included in the JCL, AR/CTL searches the JCL for the data set name. If the data set name is not present, AR/CTL allocates the data set dynamically if a skeleton is present.

2. If the ARV DD statement is included in the JCL or is dynamically allocated, AR/CTL attempts to load the module from the library that is identified with ARV DD statement.
3. If no ARV DD statement is present and AR/CTL cannot allocate it dynamically from a skeleton, it is loaded from the STEPLIB, JOBLIB, or LINKLST concatenation.

ARCSSMLBD
Optional for non-IMS DB2 application programs (can be allocated dynamically); not applicable for application programs that do not use DB2. Describes the AR/CTL procedure library containing the subsystem member that defines the DB2 subsystem connection information, as referenced by the ssm value of the PARM parameter. This data set must have the following DCB parameters: RECFM=F or FB, LRECL=80, BLKSIZE=n (where n is a multiple of 80). The name of the subsystem member that AR/CTL uses is the ARCID concatenated with the value of the ssm parameter of the EXEC statement.

The information provided with the subsystem member can be provided through the program registration record instead of the ARCSSMLB DD statement.

AR/CTL can allocate this data set dynamically if you provide a dynamic allocation record to define it. If the JCL contains the ARCSSMLB DD statement and the REGISET contains an ARCSSMLB dynamic allocation record, the DD statement overrides the dynamic allocation record. For information about how to create dynamic allocation records, see the APPLICATION RESTART CONTROL Reference Manual.

ARXRST DD
Required only for restart of non-unique job steps. If the job step is identified as non-unique and it requires restart, you must provide the data set name of the checkpoint data set that contains the necessary restart information. The presence of the restart data set in the job step JCL is the only indication that the job step needs to be restarted.

For more information, see “Restarting a non-unique job step” on page 256.

ARCPRT DD
Optional; AR/CTL dynamically allocates the ARCPRT data set if you omit the DD statement. You can include the ARCPRT DD statement for use in reporting errors detected in the ARCSYSIN control statements. The statement defines a sequential data set. You can define it as SYSOUT= to send the output to the same destination as the job messages, or you can define it as a DASD data set. On DASD, use the following DCB parameters: DSORG=PS, RECFM=FBA, LRECL=133, BLKSIZE=n (where n is a multiple of 133).
ARCSTATS DD

Optional. In addition to writing statistics to the history data set, AR/CTL can write them to the output data set defined by the ARCSTATS DD statement. If you provide the ARCSTATS data set, you can define it as SYSOUT=* to send the output to the same destination as the job messages, or you can define it as a DASD data set. On DASD, use the following DCB parameters: DSORG=PS, RECFM=FBA, LRECL=133, BLKSIZE=n (where n is a multiple of 133). If you omit the ARCSTATS DD statement, AR/CTL writes the statistical reports to the history data set only. For information about the AR/CTL reports, see the APPLICATION RESTART CONTROL Reference Manual.

SYSUDUMP

Optional. See “Obtaining traces” on page 267.

ASMTRACE

Optional; use this statement only if requested by BMC Software Product Support. See “Obtaining traces” on page 267.

RMTTRACE

Optional; use this statement only if requested by BMC Software Product Support. See “Obtaining traces” on page 267.

VSMTRACE

Optional; use this statement only if requested by BMC Software Product Support. See “Obtaining traces” on page 267.

ARCDLTRC DD

Optional. AR/CTL writes enhanced call tracing records to this data set. You can include the ARCDLTRC DD statement in the job step JCL, or AR/CTL can allocate the trace data set dynamically.

For more information, see “Using enhanced call tracing” on page 149.

ARCSYSIN DD

Optional, but not recommended for use in production environments; however, it is useful for testing options. You can set AR/CTL processing options at the execution level with keywords in the ARCSYSIN data set. Typically, this data set is included in stream with the job step JCL, but it can be in a separate data set instead. If a separate data set, use the following DCB parameters: DSORG=PS, RECFM=F or RECFM=FB, LRECL=80, BLKSIZE=n (where n is a multiple of 80).

For more information, see the APPLICATION RESTART CONTROL Reference Manual.
Using the IMS CHKPID parameter

arvdblog DD
Required for local VSAM access services (can be allocated dynamically). Specify the data set name and disposition of the VSAM dynamic backout log. AR/CTL uses this log to provide dynamic backout support for batch VSAM application programs.

The VSAM Dynamic Backout Log DDname option defines the ddname to use for this DD statement. The default ddname (as distributed by BMC Software) is ARVDBLOG. If you use another ddname (except RCMDBLOG), you must change the value of this processing option to match the ddname you use. If you use RCMDBLOG, no change is necessary; AR/CTL supports the RCMDBLOG ddname for compatibility with previous levels of AR/CTL.

AR/CTL can allocate the VSAM dynamic backout log data set dynamically if you provide a dynamic allocation record to define it. If AR/CTL dynamically allocates this data set, the value of the VSAM Dynamic Backout Log DDname option must be ARVDBLOG or RCMDBLOG.

For more information, see the APPLICATION RESTART CONTROL Reference Manual.

arvfctab DD
Required for remote VSAM access services; can be allocated dynamically. To perform remote VSAM access services, AR/CTL must access the VSAM file control table data set. You can provide a DD statement in the JCL to execute the batch application program, or you can allow AR/CTL to allocate the data set dynamically. If you provide the DD statement in the JCL, you can use any ddname; however, this ddname must be specified as the value of the VSAM File Control Table DDname processing option. The default ddname is ARVFCTAB.

For more information, see “VSAM file control table data set” on page 125.

Using the IMS CHKPID parameter

In the JCL to execute an IMS application program, AR/CTL supports use of the CHKPID parameter (the eighth positional parameter of a DLI EXEC statement, or the ninth positional parameter of a BMP EXEC statement). AR/CTL supports all IMS-supported keywords and values in the CHKPID parameter. AR/CTL also supports AR/CTL-defined keywords, which allow you to cold-start the job step and, as an option, to suppress checkpoint messages. The AR/CTL-defined keywords are valid only when AR/CTL is present during execution.

Table 2 shows CHKPID parameter keywords that control message suppression, cold start, or both. The message suppression keywords do not affect the restart status of the job step.
The following shows how AR/CTL handles other keywords and values in the CHKPID parameter:

- If the CHKPID parameter contains an 8-character checkpoint ID and the identified checkpoint record is in the restart data set, AR/CTL performs the restart. If this record is not flagged as valid for restart, AR/CTL issues an abnormal termination for the job step.

- If the CHKPID parameter contains an 8-character checkpoint ID and the identified checkpoint record is not in the restart data set, AR/CTL issues an abnormal termination for the job step.

- If the CHKPID parameter contains a time stamp and the application program does not use any AR/CTL repositioning services, AR/CTL passes control to IMS and IMS performs the restart. The Reattach Application Program option is set to N.

- If the CHKPID parameter contains a time stamp and the application program uses any AR/CTL repositioning services, AR/CTL issues an abnormal termination for the job step.

- If the CHKPID parameter contains the LAST keyword and the application program uses any AR/CTL repositioning services, AR/CTL ignores the keyword and performs the restart.

- If the CHKPID parameter contains the LAST keyword, the application program does not use any AR/CTL repositioning services, and the program is executing in a DLI batch environment, AR/CTL ignores the keyword and performs the restart.
Setting up DB2 connections

This section describes how to set up DB2 connection information.

For all application programs that use DB2, you must provide access to DB2 connection information at job step execution:

- In a non-IMS environment, you can create a program registration record or automatic registration record in the REGISET (the recommended method) or create a subsystem member (SSM) and specify the member name as a parameter on the EXEC statement in the JCL (the method that is compatible with previous versions of AR/CTL). If a program registration record and an SSM both apply to the application program execution, AR/CTL uses the information in the program registration record.

- In an IMS environment, you use the same connections that you use for DB2 connections in an IMS program without AR/CTL.

When connecting to DB2, AR/CTL uses the job name as the connection name. By default, AR/CTL uses the application program name as the plan name. If the plan name is different from the program name, you can specify the plan name in the program registration record or use the resource translation table to translate the application program name to a different plan name.

You can register a program to use the AR/CTL for DB2 batch attachment facility without using other AR/CTL services.

Specifying DB2 connection information

The method that AR/CTL uses to obtain connection information depends on whether the application program does not use IMS, uses IMS, or is compatible with IMS but IMS is not present.
Non-IMS programs

During initialization of a non-IMS program, AR/CTL tries to obtain DB2 subsystem connection information first from an SSM stored in the AR/CTL procedure library (ddname ARCSSMLB). To determine the name of the member to use, AR/CTL appends the value of the SSM parameter (which can be specified on the EXEC statement) to the ARCID.

The SSM contains a control statement that defines the DB2 connection information. If you use SSMs, one SSM is required for each DB2 subsystem that AR/CTL communicates with. For a non-IMS program, the control statement consists of the DB2 keyword and its positional parameters. If you omit an optional parameter, retain the comma to maintain the position of the other parameters. The following example shows the format for use with a DB2 program:

```
DB2=(ssn,SYS1,esmt,rtt,reo,crc)
```

IMS and IMS-compatible programs

For a program that uses IMS or IMS-compatible calls and structures, AR/CTL obtains connection information in the same way that IMS obtains it. Use the same connection techniques required for IMS and DB2 without AR/CTL: the SSM or the DDITV02 input data set. In the IMS subsystem member, you can use positional parameters or keywords. In the DDITV02 data set, you can use positional parameters.

Specifying DB2 connection parameters

The following keyword values and positional parameters are valid for the DB2 subsystem type:

**ssn**
Required. The MVS subsystem name (one to four characters) of the DB2 subsystem that AR/CTL attaches. This name is determined by your site and must match the name identified to MVS during MVS system generation.

**SYS1**
Optional. AR/CTL maintains this field for compatibility with IMS.

**esmt**
Required. The table name (one to eight characters) of the DB2 external subsystem module table. This table is created by DB2 and defined in DB2. The usual module name is DSNMIN10. It specifies the DB2 modules that AR/CTL must load and the work areas that AR/CTL must create. AR/CTL maintains this field for compatibility with IMS.
Using the AR/CTL Status Check utility

1. **rtt**
   - Optional. The table name (one to eight characters) of the resource translation table. This table is created by DB2 with the DSNMAPN macro. AR/CTL uses the resource translation table to map an application program module to a DB2 application plan. AR/CTL maintains this field for compatibility with IMS.

   For more information about generation of the resource translation table, see the DB2 administration documentation.

2. **reo**
   - Optional. The region error option code (one character) that determines the action AR/CTL takes if the application program issues a DB2 request before connection to the DB2 subsystem is complete or if problems are encountered with the DB2 subsystem. The table shows values that are valid. The default value is **R**:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>AR/CTL sends an SQL return code to the application program, indicating that the request for DB2 services has failed. If AR/CTL cannot establish an initial connection to DB2, it does not send an SQL return code; the application program terminates with a U1776 abend.</td>
</tr>
<tr>
<td>Q</td>
<td>AR/CTL does not use this value when executing without IMS but maintains it for compatibility with IMS. AR/CTL translates the value to <strong>A</strong>.</td>
</tr>
<tr>
<td>A</td>
<td>AR/CTL issues a U1776 abend for the application program.</td>
</tr>
</tbody>
</table>

3. **crc**
   - Optional. The command recognition character. AR/CTL does not use this value but maintains it for compatibility with IMS. The value can be any single EBCDIC character.

---

### Using the AR/CTL Status Check utility

The AR/CTL Status Check utility consists of a module (program ARCACTIV or program AESUVBCS), which can be called from the application program address space, and an accompanying control block ($ARCACTU). The module tests whether AR/CTL is active in the address space. If AR/CTL is not active in the address space, the module can return a specified return code or can issue an abnormal termination with a specified abend completion code and reason code. The module also returns the subsystem ID of the BCSS that the application program is using; you can use this BCSS ID as input to the AR/CTL Status Check utility (program ARCACTIV or program AESUVBCS).
Member $ARCACTU of the AR/CTL sample library contains a sample $ARCACTU DSECT that you can include in your program. The control block contains fields that you can set for the error return code or for the error abend completion code and the error abend reason code. The control block also contains fields for the return code and the BCSS ID; your program can check these fields after program ARCACTIV or program AESUVBCS sets their values.

The COBOL, PL/1, and Assembler language statements in Figure 4 through Figure 6 on page 101 show the calling sequence for program ARCACTIV.

**Figure 4  Calling sequence for COBOL program ARCACTIV**

<table>
<thead>
<tr>
<th>WORKING-STORAGE SECTION.</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 ARCACTU-PARMS.</td>
</tr>
<tr>
<td>02 ACTU             PIC X(8) VALUE '$ARCACTU'.</td>
</tr>
<tr>
<td>02 ACTU-COMPLETION PIC X(4) VALUE '00000000'.</td>
</tr>
<tr>
<td>02 ACTU-REASON        PIC X(4) VALUE '00000000'.</td>
</tr>
<tr>
<td>02 ACTU-ERROR         PIC X(4) VALUE '00000000'.</td>
</tr>
<tr>
<td>02 ACTU-RESERVED-I    PIC X(112) VALUE LOW-VALUES.</td>
</tr>
<tr>
<td>02 ACTU-RETURN        PIC X(4) VALUE '00000000'.</td>
</tr>
<tr>
<td>02 ACTU-BCS-SSID       PIC X(4) VALUE SPACES.</td>
</tr>
<tr>
<td>02 ACTU-RESERVED-O    PIC X(116) VALUE LOW-VALUES.</td>
</tr>
</tbody>
</table>

PROCEDURE DIVISION.

MOVE X'00000010' TO ACTU-ERROR.
CALL 'ARCACTIV' USING ARCACTU-PARMS.

**Figure 5  Calling sequence for PL/1 program ARCACTIV**

<table>
<thead>
<tr>
<th>DCL 1 ARCACTU STATIC,</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 ACTU            CHAR(8) INIT('$ARCACTU'),</td>
</tr>
<tr>
<td>2 ACTU_COMPLETION  BIN FIXED(31) INIT(0),</td>
</tr>
<tr>
<td>2 ACTU_REASON      BIN FIXED(31) INIT(0),</td>
</tr>
<tr>
<td>2 ACTU_ERROR       BIN FIXED(31) INIT(16),</td>
</tr>
<tr>
<td>2 ACTU_RESERVED_I  CHAR(112),</td>
</tr>
<tr>
<td>2 ACTU_RETURN      BIN FIXED(31) INIT(0),</td>
</tr>
<tr>
<td>2 ACTU_BCS_SSID    CHAR(4) INIT(' '),</td>
</tr>
<tr>
<td>2 ACTU_RESERVED_O  CHAR(116);</td>
</tr>
</tbody>
</table>

ACTU_RESERVED_I = LOW(112);
ACTU_RESERVED_O = LOW(116);
CALL ARCACTIV(ARCACTU) ;

**Figure 6  Calling sequence for Assembler program ARCACTIV**

<table>
<thead>
<tr>
<th>LA R0,$ARCACTU * LOAD A($ARCACTU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA R1,$ARCACTU_LEN(0,0) * LOAD LENGTH $ARCACTU</td>
</tr>
<tr>
<td>LA R14,=CL8'$ARCACTU' * LOAD A(IDENTIFIER VALUE)</td>
</tr>
<tr>
<td>LA R15,=B(0,0) * LOAD LENGTH IDENTIFIER VALUE</td>
</tr>
<tr>
<td>MVCL R0,R14 * INITIALIZE $ARCACTU</td>
</tr>
</tbody>
</table>

*  

| LA R1,$ARCACTU * LOAD A($ARCACTU) |
| USING $ARCACTU,R1 * ADDRESSABILITY $ARCACTU |
| LA R0,16(0,0) * LOAD ERROR RETURN CODE |
| ST R0,ACTU_ERROR * SAVE ERROR RETURN CODE |
| DROP R1 * ADDRESSABILITY $ARCACTU |
**Figure 6** Calling sequence for Assembler program ARCACTIV

```
* LOAD EP=ARCACTIV             * LOAD ARCACTIV
LR  R15,R0                     * LOAD A(ARCACTIV)
LA  R0,$ARCACTU                * LOAD A($ARCACTU)
ST  R0,@$ARCACTU               * SAVE A(A($ARCACTU))
LA  R1,$ARCACTU                * LOAD A(A($ARCACTU))
BASR R14,R15                   * CALL ARCACTIV
LR  R2,R15                     * SAVE ARCACTIV RETURN CODE
* DELETE EP=ARCACTIV           * DELETE ARCACTIV
* LTR  R2,R2                   * AR/CTL ACTIVE?
BNZ  ARC_NOT_ACTIVE           * NO -- ERROR
.
.
.
$ARCACTU SECTION=$ARCACTU,PREFIX=ACTU
```
Using AR/CTL checkpoint/restart services

This chapter provides overview information about using the checkpoint/restart services of APPLICATION RESTART CONTROL (AR/CTL); most of the information applies regardless of the type of application program and environment. This chapter contains the following information:

Overview ................................................................. 103
Methods for requesting checkpoint/restart services .............................. 104
Checkpoint/restart processing in an application program ....................... 105
Checkpoint/restart request considerations ........................................ 107
Application program changes between abend and restart ..................... 108
Program code to execute at restart ......................................... 109
Forced checkpoints ....................................................... 110
Using DB2 commit statements to trigger checkpoints ........................... 110

Overview

Checkpoint/restart services provide (or enhance) an environment that allows an application program job step to be restarted after a failure. These services are available through AR/CTL processing options (in some cases) and through requests explicitly issued by the application program.

Most of the specific information about checkpoint/restart services is located in other parts of the AR/CTL documentation set. This chapter discusses general topics related to checkpoint/restart services and tells you where to locate specific information.
Methods for requesting checkpoint/restart services

To provide checkpoint/restart services, AR/CTL requires an application program to request these services. You can choose among several methods, depending on the AR/CTL product you have and the type of application program (see Table 3).

**NOTE**
You cannot mix different types of checkpoint triggers in a single application program. You cannot mix AR/CTL-format calls with IMS-format calls in the same program.

### Table 3  Checkpoint/restart services summary

<table>
<thead>
<tr>
<th>Checkpoint/restart service</th>
<th>AR/CTL product</th>
<th>Description</th>
<th>Format of program requests</th>
<th>More information</th>
</tr>
</thead>
<tbody>
<tr>
<td>automatic restart</td>
<td>available in all AR/CTL products for use with COBOL programs</td>
<td>Specify Y for the Automatic Restart option; you can also specify values for the Restore Working Storage and Restart Return Code options.</td>
<td>none</td>
<td>AR/CTL reference manual</td>
</tr>
<tr>
<td>automatic checkpoints</td>
<td>available in all AR/CTL products for use with COBOL programs</td>
<td>Specify Y for the Automatic Checkpoints option, specify the trigger to use, and specify the count of I/O or fetch requests against the triggers; automatic restart is required.</td>
<td>none</td>
<td>AR/CTL reference manual</td>
</tr>
<tr>
<td>AR/CTL-format restart and checkpoint calls</td>
<td>available in AR/CTL for DB2 and AR/CTL for VSAM</td>
<td>The program issues calls directly to AR/CTL for checkpoint/restart services.</td>
<td>ARCXRST call, ARCECHK call, ARCCHKP call</td>
<td>page 177</td>
</tr>
<tr>
<td>AR/CTL common call</td>
<td>available in all AR/CTL products</td>
<td>The program issues calls directly to AR/CTL for checkpoint/restart services.</td>
<td>langTARC call, where lang is CBL, ASM, or PLI</td>
<td>page 191</td>
</tr>
<tr>
<td>DB2 commits</td>
<td>available in AR/CTL for DB2</td>
<td>DB2 commit statements trigger AR/CTL checkpoint requests.</td>
<td>EXEC SQL COMMIT statement</td>
<td>page 110</td>
</tr>
<tr>
<td>IMS-format restart and checkpoint calls</td>
<td>available in AR/CTL for IMS</td>
<td>The program issues standard IMS extended checkpoint/restart calls.</td>
<td>langTDLI call, where lang is CBL, ASM, or PLI</td>
<td>page 195</td>
</tr>
</tbody>
</table>
Checkpoint/restart processing in an application program

This section describes a typical processing flow of an application program that uses explicit checkpoint/restart requests. Depending on the method that the program uses to obtain checkpoint/restart services, the program might need to identify function codes and storage areas for structures such as the restart ID area and the checkpoint application request block (ARB).

For information about how processing occurs for a program that uses Automatic Restart and Automatic Checkpoints options, see the APPLICATION RESTART CONTROL Reference Manual.

1. Receive control from AR/CTL or IMS.

2. Issue a request (XRST or ARCXRST call) for restart processing during initialization. The restart request establishes the checkpoint/restart environment and determines the starting status of the job step. It also identifies one to seven areas in the program’s working storage to save during checkpoint processing and restore during restart processing. AR/CTL performs initialization processing. If the program is restarted from a previous checkpoint, AR/CTL restores the user areas identified by the restart request.

**NOTE**
The program should always issue the restart request before the first checkpoint request, data set open request, DL/I call, or EXEC SQL call.

3. When AR/CTL returns control, check the contents of the restart ID field:

   - If the restart ID area is blank, perform normal start initialization processing.
   - If the restart ID area is not blank, perform restart initialization processing. This processing might include priming variables and reestablishing position within DB2 tables. IMS automatically reestablishes position in IMS databases. AR/CTL automatically reestablishes position in the sequential, VSAM, and ASAM data sets that use AR/CTL data services.

4. Issue the first request (CHKP or ARCCHPK call) for checkpoint processing. Checkpoint pacing does not apply to the first checkpoint; AR/CTL always allows the first checkpoint to complete.
5. Issue data set open requests.

6. Perform processing to the end of a logical unit of work (UOW).

7. If applicable, save information needed to reestablish DB2 table position. This action would be necessary if, for example, the application program does not use the DB2 CURSOR WITH HOLD option. This information must be in a user area that is saved during checkpoint processing.

8. Issue a checkpoint request. If you use checkpoint pacing (bypass), the program can issue a checkpoint request at the end of any logical UOW without concern for operational and performance issues. If checkpoint pacing requirements allow checkpoint processing to occur, AR/CTL notifies the DBMS to commit changes. In preparation for a possible restart, AR/CTL saves the application program’s working storage areas and sequential data set positioning information.

9. If applicable, check the status code:

   - If the program uses AR/CTL-format checkpoint calls and checkpoint pacing and the Pacing Status Code option (in the pacing class record) has a nonblank value, the program can check the status code field in the checkpoint ARB. If the program does not use checkpoint pacing, the status code is always blank.

   - If the program uses AR/CTL common calls and checkpoint pacing and the Pacing Status Code option (in the pacing class record) has a nonblank value, the program can check the status code field named in the common call. If the program does not use checkpoint pacing, the status code is always blank.

   - If the program uses IMS-format checkpoint calls, the program can check the status code field in the checkpoint program communication block (PCB).

   - If the program issues a DB2 commit statement, AR/CTL does not return a status code.

   **NOTE**
   If AR/CTL detects a problem, it does not return control to the application program. AR/CTL terminates the application program task.

10. Take action based on the status code, if applicable; for most application programs, no action is necessary:
If the status code field indicates that AR/CTL has paced (bypassed) the checkpoint request, the program might need to accommodate the paced checkpoint. For example, if the program does not use the DB2 CURSOR WITH HOLD option, the program must not attempt to reestablish DB2 table position or it must be able to tolerate a DB2 status code indicating that table position was already established.

If the status code field is blank, AR/CTL has not paced the checkpoint request. For example, if the program does not use the DB2 CURSOR WITH HOLD option, the program must reestablish database or table position.

11. Continue processing UOWs, and issue a checkpoint request at the end of each UOW.

12. Issue the final checkpoint request before termination, to ensure synchronization of multiple DBMSs. BMC Software recommends that you use the force option with the AR/CTL common call or the ARCECHK call to ensure that this checkpoint is not bypassed because of checkpoint pacing.

During termination, AR/CTL performs the following tasks:

- AR/CTL closes all ASAM data sets.
- AR/CTL closes all data sets that use local virtual storage access method (VSAM) access services of AR/CTL. If AR/CTL obtained exclusive use of the data set from CICS, it leaves the data set closed and disabled to CICS.
- AR/CTL ends the connection with any CICS regions that AR/CTL was using for remote VSAM access services.

**Checkpoint/restart request considerations**

Keep the following information in mind when you use checkpoint/restart requests:

- If an application program uses remote VSAM access services (VSAM file sharing), the frequency of checkpoints issued can affect performance of CICS systems that own shared VSAM data sets. When an application program updates a record in the CICS environment, CICS locks the record or control interval (CI) until the program commits the changes by issuing a checkpoint call. As the interval between program checkpoints grows longer, more locks are held and, potentially, performance is degraded.

- If the application program is using the DB2 option CURSOR WITH HOLD, it does not need to reposition the DB2 tables after a checkpoint request; however, DB2 does not release locks until the application program issues a CLOSE CURSOR or terminates.
Application program changes between abend and restart

For COBOL programs, only data in the working storage section or linkage section can be saved during checkpoint processing and restored during restart. The data areas in the FILE SECTION are not restored; if data from a record previously read or written is needed after restart, move the data to the working storage section.

The application program should save the same areas in the checkpoint requests as the areas restored in the restart request. Saving different areas is not recommended; an abend results if the lengths of the two areas are different. If the application program saves different areas, the program must ensure that it tells AR/CTL to restore these fields during restart by matching them on the restart request. If AR/CTL cannot complete the checkpoint, it issues an application program abend. Saving areas of different lengths in different checkpoint requests is not a valid coding practice.

Although AR/CTL does not have any restrictions on the size of working storage the program can save during checkpoint processing, be aware that larger storage areas are more costly (in performance and resources) to write than smaller storage areas. If the program has large storage areas to save, you might want to reduce the frequency of checkpoints through the use of checkpoint pacing.

To implement checkpoint/restart services in programs that have subprograms, you can use the Subprogram Virtual Storage (ARCSPVS) application program interface (API) to define the virtual storage areas that you want AR/CTL to save during checkpoint processing and to restore automatically during initialization processing or manually on demand. You can define these areas in addition to the user areas that you can define with checkpoint (CHKP) and restart (XRST) calls.

The program should issue a checkpoint call as the final call. If the program is using checkpoint pacing, the checkpoint call should use the force option to ensure that the checkpoint is not bypassed.

If the application program uses AR/CTL checkpoint/restart services, it must issue the restart request before opening a data set that uses AR/CTL data services. It must also issue the same type of open request (input, output, or I/O) at restart time as the last type of open request it issued before the abend occurred.

Application program changes between abend and restart

If you need to change an application program between a program failure and a restart, you must not change the way that the program maps the working storage areas or the lengths of the working storage areas that are saved in the checkpoint record.
AR/CTL uses starting and ending addresses, rather than field definitions, to restore user areas. If you change the mapping of these areas, the fields in the restored areas will not match the fields in the changed program and unpredictable results might occur. Actions that change the map of working storage include adding or deleting fields and changing the length of fields.

It is generally safe to recompile and relink the program between abend and restart.

You can add new fields at the end of working storage. If you use automatic checkpoints, these new fields are saved during checkpoint processing after restart.

If you use automatic DB2 cursor repositioning with the count method, you can change the declare cursor statement (but the cursor name must not change) and add or remove columns. If you use the column method, you cannot remove a column if that column is defined to AR/CTL in the cursor repositioning record. You can add other columns.

You can add new logic to the program, such as logic to bypass a bad input record that has caused the abend.

**Program code to execute at restart**

It might be necessary to write some code, such as priming application program variables, to execute when an application program is restarted. For example, a page counter might need to be incremented. If the last page of a report written before failure was 153, the next page after restart will be 154 and the application program might need to pre-increment the page counter. This type of code is specific to the application program and usually requires individual analysis of each program to ensure proper restartability.

When the application program issues the restart call, AR/CTL repositions the data sets that are using AR/CTL data services. However, if a COBOL program accesses a KSDS through an alternate index (AIX), the program must reestablish its position to the appropriate key in the AIX by using the key information in a data area saved during checkpoint processing.

If the application program is using DB2, the application program must reposition the DB2 tables by using the key information in a data area saved during checkpoint processing. If the application program already contains restart logic for IMS restart, no change is necessary.

The application program must issue an OPEN request after the restart (ARCXRST or XRST) call for any file that AR/CTL accesses through dynamic call interception (sequential file interception, local VSAM access, and remote VSAM access services).
Forced checkpoints

At certain points in an application program, the program might need to force a checkpoint (exempt it from being paced). These points include the final checkpoint issued by the application program. The following shows how a program can force a checkpoint:

- Non-IMS programs can issue the ARCECHK call with a code of FORCE in the ARB.
- All programs can issue the AR/CTL common call with the force option.
- IMS and IMS-compatible programs that use the IMS-compatible checkpoint (CHKP) call, which does not support a force option, can force a checkpoint through a user exit routine. For more information, see the APPLICATION RESTART CONTROL Reference Manual.

Forcing a checkpoint resets timers and counters that determine when to pace a checkpoint.

**NOTE**

AR/CTL automatically exempts from pacing the first checkpoint issued by the application program (or caused by automatic checkpoint processing).

Using DB2 commit statements to trigger checkpoints

This section describes how to use DB2 commit statements in an application program with AR/CTL. During program execution, DB2 commit statements trigger AR/CTL checkpoint requests. This service reduces the changes necessary for an existing program. When you use DB2 commit statements, keep the following issues in mind:

- For a DB2 commit statement to trigger checkpoint requests, an application program must use the ARCXRST call, the AR/CTL common call XRST function, or the Automatic Restart option. During restart request processing, AR/CTL performs the processing necessary to prepare for DB2 commit statements to trigger checkpoint requests. AR/CTL saves the user areas identified in the restart call or all of working storage if the program is using the Automatic Restart option.

- An application program cannot issue IMS-format calls with DB2 commit statements. IMS does not allow DB2 commit statement processing from within an IMS application program.
- An application program must use DB2 commit statements only, or one type of AR/CTL checkpoint request (whether from an explicit call or from the Automatic Checkpoints option) only, but not both.

- AR/CTL automatically generates a checkpoint ID when an application program issues a DB2 commit statement.

- An existing application program might issue DB2 commit statements for the sole purpose of releasing DB2 locks. You might need to review the program logic to determine whether the use of DB2 commit statements is appropriate for application restart.

- An application program retains control over issuing the DB2 commit statement.

- If an application program uses remote VSAM access services (VSAM file sharing), the frequency of checkpoints issued can affect performance of CICS systems that own shared VSAM data sets. When an application program updates a record in the CICS environment, CICS locks the record or control interval (CI) until the program commits the changes by issuing a checkpoint (CICS syncpoint). As the interval between program checkpoints (commits) grows longer, more locks are held, which can adversely affect other tasks or jobs that are processing against the same data.

- If an application program is using CURSOR WITH HOLD, the program does not need to reposition the DB2 tables after a CHKP call. The main reason for issuing checkpoint calls/commits when using CURSOR WITH HOLD is to release DB2 locks. These locks will be released during checkpoint/commit processing, CLOSE CURSOR, or program termination.
Using AR/CTL data services

This chapter provides information about using the data services of APPLICATION RESTART CONTROL (AR/CTL); most of the information applies regardless of the type of application program and environment. This chapter contains the following information:

Overview .................................................................................................................. 114
General data services considerations ................................................................. 116
  Using checkpoint/restart services with data services ........................................ 116
  Output sequential files ....................................................................................... 116
  Preallocating data sets .................................................................................... 117
  Using GDGs ....................................................................................................... 118
  Using ASAM data set allocation parameters .................................................... 118
Using sequential file interception services .......................................................... 119
  Application program calls .............................................................................. 119
  Processing option values ............................................................................... 120
  Considerations ................................................................................................. 120
Using local VSAM access services ...................................................................... 121
  Application program calls .............................................................................. 121
  Processing option values ............................................................................... 122
  Considerations ................................................................................................. 123
Using remote VSAM access services .................................................................. 123
  Application program calls .............................................................................. 123
  Processing option values ............................................................................... 125
  VSAM file control table data set .................................................................... 125
  AR/CTL enhanced processing ....................................................................... 126
  CICS environment and connection information ............................................ 126
  CICS connection ID and APPLID tables ......................................................... 126
  CICS resource usage ...................................................................................... 127
  Using automatic syncpoints .......................................................................... 128
Using ASAM services through GSAM replacement ............................................ 130
  GSAM replacement call overview .................................................................. 130
  Data set options .............................................................................................. 131
  Processing option values ............................................................................... 131
Using ASAM services through the ASAM callable interface ............................. 132
  ASAM callable interface supported data set types ......................................... 132
  ASAM call ........................................................................................................ 132
The data services of AR/CTL focus on reestablishing the application program’s position within non-DBMS data sets at restart. Data services can also provide availability, integrity, and performance enhancements for supported data sets. AR/CTL provides the following data services:

- Sequential file interception is for data sets that an application program accesses through standard MVS QSAM requests. AR/CTL dynamically intercepts these requests so that it can reposition the data set at restart time. For more information, see “Using sequential file interception services” on page 119.

- Local VSAM access is for VSAM data sets accessed exclusively by a batch VSAM application program. The program issues native VSAM requests. AR/CTL for VSAM dynamically intercepts these requests to provide repositioning support. AR/CTL can perform logging and automatic backout of uncommitted changes to VSAM data sets made by the batch program. For more information, see “Using local VSAM access services” on page 121.

- Remote VSAM access (also known as VSAM file sharing) allows VSAM data sets to be shared and updated between a batch VSAM application program and one or more CICS regions on the same or a different MVS image. While a file is being shared, the CICS region owns the data sets; it manages all access to the data sets, logs changes made by those transactions, and backs out uncommitted changes in case of failure. AR/CTL dynamically intercepts the native VSAM requests issued by a batch VSAM application program, transforms the requests to online CICS commands, and ships the commands to the CICS region for service. For more information, see “Using remote VSAM access services” on page 123.

- IMS generalized sequential access method (GSAM) replacement is for IMS and IMS-compatible application programs. The program issues standard IMS GSAM calls. In response to these calls, AR/CTL replaces the IMS GSAM services and provides services that IMS does not provide. For more information, see “Using ASAM services through GSAM replacement” on page 130.
The application sequential access method (ASAM) callable interface is mainly for DB2 application programs to use if AR/CTL for VSAM is not available. A program can use the ASAM callable interface to obtain limited support for repositioning of VSAM entry-sequenced data sets (ESDSs), input VSAM key-sequenced data sets (KSDSs), and standard MVS sequential (QSAM) files. For more information, see “Using ASAM services through the ASAM callable interface” on page 132.

Automatic DB2 cursor repositioning is for DB2 application programs. You set up AR/CTL definitions to obtain cursor repositioning; no program changes are necessary. For more information, see the APPLICATION RESTART CONTROL Reference Manual.

Methods for using data services

Table 4 summarizes the types and functions of AR/CTL data services.

**NOTE**

You cannot mix AR/CTL-format calls with IMS-format calls in the same program.

<table>
<thead>
<tr>
<th>Data service</th>
<th>AR/CTL product</th>
<th>Functions provided</th>
<th>Format of program requests</th>
<th>More information</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequential file interception</td>
<td>available in all AR/CTL products</td>
<td>repositioning at restart with block-level and record-level repositioning methods; output staging; user exit</td>
<td>standard MVS QSAM requests</td>
<td>page 119</td>
</tr>
<tr>
<td>local VSAM access</td>
<td>available in AR/CTL for VSAM</td>
<td>repositioning at restart; user exit; logging and dynamic backout</td>
<td>standard VSAM requests</td>
<td>page 121</td>
</tr>
<tr>
<td>remote VSAM access</td>
<td>available in AR/CTL for VSAM</td>
<td>repositioning at restart; user exit; access from batch programs to VSAM files owned by a CICS region (file sharing) at several levels</td>
<td>standard VSAM requests</td>
<td>page 123</td>
</tr>
</tbody>
</table>
General data services considerations

You must consider the following topics when you use AR/CTL data services:

- using checkpoint/restart services with data services
- output sequential files
  - preallocating data sets
- using generation data groups (GDGs)
- using ASAM data set allocation parameters

### Using checkpoint/restart services with data services

If the application program uses AR/CTL checkpoint/restart services, it must issue the restart request before opening a data set that uses AR/CTL data services. It must also issue the same type of open request (input, output, or I/O) at restart time as the last type of open request it issued before the abend occurred.

If you use the Automatic Restart option, AR/CTL will issue the request for restart services automatically on behalf of the application program before it receives control.

### Output sequential files

After an application program abends, any output sequential files that the application creates must not be deleted. These files must be available for AR/CTL to reposition at the subsequent restart.

---

### Table 4 Data services summary (part 2 of 2)

<table>
<thead>
<tr>
<th>Data service</th>
<th>AR/CTL product</th>
<th>Functions provided</th>
<th>Format of program requests</th>
<th>More information</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASAM through GSAM replacement</td>
<td>available in AR/CTL for IMS</td>
<td>repositioning at restart with block-level and record-level repositioning methods; output staging; IMS record search argument (RSA) emulation; user exit</td>
<td>IMS GSAM calls (IMS format)</td>
<td>page 130</td>
</tr>
<tr>
<td>ASAM through the ASAM callable interface</td>
<td>available in AR/CTL for DB2 and AR/CTL for VSAM</td>
<td>repositioning at restart with block-level and record-level repositioning methods; output staging; IMS RSA emulation; user exit</td>
<td>calls to the ASAM callable interface (AR/CTL format)</td>
<td>page 132</td>
</tr>
</tbody>
</table>

---
If the JCL specifies `DISP=(NEW,CATLG,CATLG)` for the output file, a JCL change will be required at restart time. To avoid the JCL change, preallocate the output files.

**Preallocating data sets**

To avoid potential problems with some job scheduling packages, BMC Software recommends that you preallocate any data sets that use data services; preallocate them in a job step that is previous to the job step that may be restarted. If the restartable job step abends and the data set is allocated in the step with `DISP=(NEW,CATLG,...)`, some job scheduling packages automatically delete the data set. Potential GDG reference problems may also occur.

The following example shows the pre-allocation of an output sequential data set:

\[
//ALLOC EXEC PGM=IEFBR14
//DD1 DD DSN=your.output.dsn,
// DISP=(NEW,CATLG,CATLG),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),
// UNIT=SYSDA,SPACE=(CYL,(5,1))
\]

If you allocate the data set in a previous job step, you can specify `DISP=(OLD,CATLG,CATLG)` or `DISP=(SHR,CATLG,CATLG)` in the restartable job step. Do not allocate it with `DISP=(MOD,CATLG,CATLG)`.

**WARNING**

If a data set is allocated as `DISP=(NEW,CATLG,CATLG)` in a job step (and not preallocated in a previous job step), you must change the DD statement to `DISP=(OLD,CATLG,CATLG)` before the job step can be restarted.

You should code `DISP=(OLD/SHR,CATLG,CATLG)` instead of simply `DISP=OLD/SHR`. After the application program has been successfully restarted, it is possible for a sufficient number of records to be added to the sequential output file to cause the file to expand onto another volume. Without the abnormal-termination disposition of `CATLG`, these new volumes would not be registered in the catalog.

**NOTE**

If no additional volumes are added to the sequential file, the job output may contain the message `NOT RECTLGD 2`. This is normal and is not to be confused with the error message `NOT CATLGD 2`.

---

Chapter 4   Using AR/CTL data services   117
However, when AR/CTL performs application reattach processing for an application program that abends, the job step is still active in the MVS system and the disposition status of any files allocated in the job step does not change. Therefore, during application reattach, AR/CTL can reposition files defined in the job step as \texttt{DISP=(NEW,CATLG,CATLG)}.

**Using GDGs**

AR/CTL supports the uses of GDGs for input and output sequential data sets. BMC Software recommends that you preallocate GDGs in a separate job step that runs before the application program job step. Then you can use a GDG allocation of +1 and \texttt{DISP=(OLD,CATLG,CATLG)} or \texttt{DISP=(SHR,CATLG,CATLG)}, whether the job step is an initial start or a restart.

If you do not preallocate the GDG, you must change the restart JCL to allocate the data set with \texttt{DISP=(SHR,CATLG,CATLG)} or \texttt{DISP=(OLD,CATLG,CATLG)} instead of \texttt{DISP=(NEW,CATLG,CATLG)}.

**Using ASAM data set allocation parameters**

To provide compatibility with IMS GSAM, AR/CTL determines the block size and logical record length for ASAM data sets, much like IMS does for GSAM. AR/CTL supports the system-determined block size, just as GSAM does. The record format can be fixed, variable, or undefined, and it can contain optional control characters.

The following shows how AR/CTL determines the block size for a new ASAM data set:

- If either the file characteristics block (FCB) or the JCL has defined the block size, AR/CTL uses the defined block size as the data set block size.
- If the FCB and the JCL have defined conflicting block sizes, AR/CTL uses the JCL-defined value for data sets that use sequential interception; it uses the FCB value for data sets that use the ASAM callable interface.
- If the block size is not defined in the FCB or the JCL, AR/CTL uses the system-determined block size.

The following shows how AR/CTL determines the logical record length for a new ASAM data set:

- If the FCB or the JCL has defined the logical record length, AR/CTL uses the defined logical record length as the data set logical record length.
If the FCB and the JCL have defined conflicting logical record lengths, AR/CTL uses the JCL-defined value.

**Existing data set**

When an existing ASAM data set is opened, AR/CTL determines the block size and the logical record length in the following order:

1. AR/CTL saves the block size and the logical record length values from an existing GSAM DBD (if applicable).

2. AR/CTL overlays the saved values with the values from the JCL or from the data set label in the system catalog. For a data set on a non-labeled tape, AR/CTL overlays the saved values with the values from the DCB information.

AR/CTL overlays the saved values with the values from the FCB (if applicable).

**Using sequential file interception services**

This section describes the sequential file interception services of AR/CTL and summarizes the requirements for implementing sequential file interception services.

AR/CTL can dynamically intercept the standard MVS QSAM requests that an application program makes to non-VSAM sequential data sets. The sequential data set must use the queued sequential access method (QSAM). Through sequential file interception services, AR/CTL can provide file repositioning during restart without major changes to the application program. Sequential file interception services are available with AR/CTL for DB2, AR/CTL for IMS, and AR/CTL for VSAM.

**NOTE**

If an IMS program uses a program option member that contains any data set option members specifying sequential interception, local VSAM access, or remote VSAM access, AR/CTL forces the use of GSAM replacement.

**Application program calls**

To use sequential file interception, the application program issues standard MVS QSAM requests to the sequential data set. If the program contains existing QSAM requests, typically no change is necessary.
Processing option values

No specific processing option values specifically apply to the use of sequential file interception services.

Considerations

Keep the following considerations in mind when you use sequential file interception:

- The application program must issue the restart request before the first data set OPEN request. The program can issue the restart call (in one of several formats), or it may be able to use the Automatic Restart option.

- AR/CTL will call no DCB exits other than the end-of-data exit.

- AR/CTL does not support QSAM files opened in update mode (for example, with an OPEN I-O call).

- AR/CTL will not intercept I/O requests for files opened with OPEN TYPE=J calls.

- For data sets that use AR/CTL sequential interception and that are opened with the COBOL EXTEND option, AR/CTL cannot provide repositioning support if the data set is copied or reblocked, or if the record length or record format is changed, between abend and restart. Also, AR/CTL cannot provide repositioning support for this type of data set if AR/CTL was not present when the data set was originally created with EXTEND.

- AR/CTL does not support input QSAM files with a record format of VBS or output QSAM files with a record format of VBS or FBS.

- If the data set uses output staging and a failure occurs, AR/CTL discards any records written since the last checkpoint. If the data set does not use output staging and a failure occurs, the records written since the last checkpoint may be present in the data set before restart. The records are overwritten after the restart.

- Sequential data set repositioning requires each volume of a multi-volume output tape data set to be mounted and unloaded. Although only the last volume is read, the other volumes must be mounted to keep the catalog and tape headers correct and complete for future use.
Using local VSAM access services

This section describes how to use the local VSAM access services of AR/CTL for VSAM and summarizes the requirements for implementing local VSAM access services. Local VSAM access services allow AR/CTL to perform VSAM logging and dynamic backout and to reposition a VSAM data set that is allocated to a batch application program.

Application program calls

To use local VSAM access services, an application program issues native VSAM I/O requests to the VSAM data sets as normal. If the program contains existing VSAM I/O requests, typically no change is necessary. While you are setting up the program, keep the following consideration in mind:

- The application program must issue the restart request before the first data set OPEN request. The program can issue the restart call (in one of several formats), or it may be able to use the Automatic Restart option.

- If you want repositioning for an ESDS, KSDS, or RRDS, do not use the SPEED option during the load process. Use the default option, RECOVERY.

- To use repositioning, the program cannot use control interval (CI) access. Control interval access does not support the journaling exit, which is required for AR/CTL to provide restart and repositioning services.

- If you want repositioning for a VSAM data set, do not set the RESET attribute in the access method control block (ACB), and do not define the data set with the REUSE attribute. If you use these attributes, the data set will be empty at restart.

- You must provide manual backout for a VSAM ESDS that is updated through an alternate index. AR/CTL does not provide automatic backout in this situation.

- Do not modify the ACB or request parameter list (RPL) between an abend and a restart.

- If the application program maintains its own position within the VSAM data set, it must save the information needed to reposition the data set in a user area that is saved during checkpoint processing.

- If a COBOL program is accessing a KSDS through an alternate index (directly or sequentially), it must save the key information needed to reposition the alternate index in a user area that is saved during checkpoint processing.
If the application program is updating any VSAM files sequentially (deferred buffer writes), data integrity is compromised when the application task is canceled or abended with the STEP option on the Assembler language ABEND macro.

AR/CTL supports VSAM batch LSR for direct access VSAM data sets. AR/CTL does not support VSAM batch LSR for sequential access VSAM data sets.

The VSAM logging and dynamic backout component of local VSAM access services does not support the following types of VSAM data sets:

- AIX updated in native mode
- spanned ESDS
- path over an ESDS
- linear data set
- data sets updated in CI mode

**Processing option values**

The following processing options are used for application programs that use local VSAM access services:

- The Perform VSAM Logging option must be set to Y.
- The Perform VSAM Dynamic Backout option must be set to Y for AR/CTL to perform VSAM backout if a failure occurs. If AR/CTL cannot perform the backout, you must handle the backout manually.
- For a program that performs mainly sequential updates, you can set the Defer WAL Writes option to Y. If you do so and a system failure occurs or the job step is cancelled, AR/CTL cannot perform dynamic backout.
- You can set the VSAM Dynamic Backout Log ddname to the ddname to use for the VSAM dynamic backout log. The default ddname is ARVDBLOG. If you use this default ddname, you can create a dynamic allocation record to allocate the log dynamically. If you do not create this dynamic allocation record, you must change the application program JCL to include the VSAM dynamic backout log DD statement. For compatibility with previous levels of AR/CTL, ddname RCMDBLOG is valid in the JCL.

For more information about these options, see the *APPLICATION RESTART CONTROL Reference Manual*. 
Considerations

AR/CTL uses a relative byte address (RBA) to identify records contained within a VSAM ESDS. A compression product may change the RBA of an ESDS record as updates occur. Because the RBAs in a compressed ESDS may be unpredictable, this type of data set should be excluded from compression if you want AR/CTL to perform dynamic backout.

If the BMC Software RECOVERY PLUS for CICS/VSAM (RPCV™) product is active in the application execution environment with AR/CTL for VSAM, AR/CTL for VSAM performs the batch backout.

Using remote VSAM access services

This section describes how to use remote VSAM access services and summarizes the requirements for implementing these services. Remote VSAM access services (also known as file sharing) are available with AR/CTL for VSAM. These services allow a batch application program to access VSAM data sets that are allocated to a CICS region.

Application program calls

To use remote VSAM access services, an application program issues native VSAM requests to the remote data sets as normal. If the program contains existing VSAM I/O requests, typically no change is necessary. While you are setting up the program for remote VSAM access services, keep the following considerations in mind:

■ The application program must issue the restart request before the first data set OPEN request. The program can issue the restart call (in one of several formats), or it may be able to use the Automatic Restart option.

■ The program cannot use control interval access. Remote VSAM access services support record-level access only.

■ AR/CTL can pass back only the same type of access (key or address) as specified in the RPL when the application program issues a request to access a record. For example, if the application program passes a key to access a KSDS record through remote VSAM access services, AR/CTL can pass back only the key. In native VSAM, an application program that passes a key obtains the key and the RBA of the record accessed.

■ Remote VSAM access services do not support update of an ESDS when sequentially processed through a unique or nonunique PATH.
To access a VSAM data set through remote VSAM access services, the application program must have Resource Access Control Facility (RACF) authority to access the data set.

If you use checkpoint/restart services in addition to remote VSAM access services, keep the following considerations in mind:

- Do not modify the ACB or RPL between an abend and restart.
- Remote VSAM access services do not support restart of an application program that uses an ESDS in LOAD mode.
- Remote VSAM access services do not support restart of an application program that updates a KSDS through a nonunique PATH.
- If you want repositioning for an ESDS, KSDS, or RRDS, do not use the SPEED option during the load process. Use the default option, RECOVERY.
- If you want repositioning for a VSAM data set, do not set the RESET attribute in the ACB, and do not define the data set with the REUSE attribute. If you use these attributes, the data set will be empty at restart.
- If the application program maintains its own position within the VSAM data set, save the information needed to reposition the data set in a user area that is saved during checkpoint processing.
- If a COBOL program is accessing a KSDS through an alternate index (directly or sequentially), it must save the key information needed to reposition the alternate index in a user area that is saved during checkpoint processing.
- If the application program is updating any VSAM files sequentially (deferred buffer writes), data integrity is compromised when the application task is canceled or abended with the STEP option on the Assembler language ABEND macro.
- If the application program is using a VSAM data set in LOAD mode, the data set cannot be shared with CICS through remote VSAM access services. In the default or specific data set option member (also known as the file characteristics block or FCB), you must define the data set as a local VSAM data set or as a remote VSAM data set with access level 1 (exclusive use). If the data set option member is defined with remote VSAM access level 2 or 3 and the program is trying to access the data set in load mode, AR/CTL for VSAM abnormally terminates the task.
- If an Assembler language application program or subroutine uses the VSAM ACB macro or GENCB macro to generate an access method control block (ACB), the MACRF keyword must use the default value DDN (which means the subtask shared control block connection is based on common ddnames). The value DSN (which means the subtask shared control block connection is based on common data set names) is not valid.
A single batch application program can share files with multiple CICS regions. Each region must be prepared for AR/CTL file sharing.

**Processing option values**

The following processing option values are used for application programs that use remote VSAM access services:

- **The CICS SVC Number option** specifies the SVC number that AR/CTL uses when connecting to a CICS region. This SVC number corresponds to the SVC number specified in the MVS system for the CICS Type 2 (CICS 2.1) or Type 3 (CICS/ESA) SVC.

- **The VSAM File Sharing Only option** reduces elapsed time during initialization and virtual storage requirements for application programs that use only remote VSAM access services. This option also enables the use of automatic syncpoints.

- **The Autosync Trigger DDname option** and the **Autosync Trigger Count option** allow AR/CTL to issue automatic syncpoints based on I/O activity against the specified file.

- **The VSAM File Control Table DDname option** identifies the ddname to use for the VSAM file control table data set in the batch application execution.

For more information about these options, see the *APPLICATION RESTART CONTROL Reference Manual*.

**VSAM file control table data set**

To enable file sharing, AR/CTL uses the VSAM file control table data set. AR/CTL accesses this data set from each CICS region and batch application program execution that participates in file sharing. The data set name must be the same for all participants in file sharing across a SYSPLEX or stand-alone MVS image. Typically, this data set is allocated and formatted during AR/CTL configuration.

To provide access to the VSAM file control table data set in a CICS region, the CICS system programmer adds a DD statement to the CICS startup JCL; the ddname in the CICS JCL must be ARVFCTAB.
To provide access to the VSAM file control table data set in batch application program execution, you can use either of two methods:

- You can include a DD statement in the execution JCL, as explained in “Using AR/CTL DD statements” on page 92. You can use any ddname, but you must use the VSAM File Control Table DDname processing option to identify this ddname to AR/CTL.

- You can create a dynamic allocation record (ddname ARVFCTAB) in the REGISET, as explained in the APPLICATION RESTART CONTROL Reference Manual. The dynamic allocation record method requires no JCL changes.

AR/CTL enhanced processing

The VSAM Services function of AR/CTL enhanced processing must be installed in the MVS system where you want to use remote VSAM access services. Typically, this function is installed during AR/CTL configuration. For more information, see the APPLICATION RESTART CONTROL Configuration Guide.

CICS environment and connection information

CICS environment and connection information must be defined in the CICS environment and CICS regions in which you want to use remote VSAM access services. Typically, this information is defined by the CICS system programmer during AR/CTL customization. For more information, see the APPLICATION RESTART CONTROL Configuration Guide.

CICS connection ID and APPLID tables

Required CICS connection IDs are defined to AR/CTL in the CICS connection ID table. The required CICS application IDs (APPLIDs) can be defined to AR/CTL in the CICS APPLID table. AR/CTL uses the APPLIDs to identify the CICS regions that own the files that will be using remote VSAM access services. If a required APPLID is not defined in the table, the data set options member must contain the APPLID.

To provide remote access to VSAM data sets owned by an online CICS region (also known as file sharing), AR/CTL connects to the CICS region during application program execution. To make the connection, AR/CTL must use one of the CICS connection entries that the CICS system programmer defines for AR/CTL in the CICS
region. To identify the defined connections to AR/CTL, the CICS system programmer sets up a table of connection IDs in AR/CTL, where each connection ID is the Netname parameter defined in a connection entry in the CICS region. Typically, these tasks are performed during AR/CTL configuration.

Your site can define a table of CICS application IDs (APPLIDs) that may own shared data sets. AR/CTL uses the table of CICS APPLIDs to identify the CICS regions to search to determine which CICS region owns a file. If this table is not defined or does not contain the APPLID of the CICS region that owns a shared data set, the data set option member (FCB) for that data set must contain the APPLID. Typically, the CICS system programmer sets up the CICS APPLID table during AR/CTL configuration.

These definitions typically are set up by the CICS system programmer during AR/CTL customization. For more information, see the APPLICATION RESTART CONTROL Configuration Guide.

CICS resource usage

Because remote VSAM access services use CICS resources, you should evaluate the effects of remote VSAM access services on the CICS region and take steps to prevent problems. The primary concerns are resource locks and overflow of the CICS system log data sets.

Resource locks

CICS maintains locks on data updated by a batch VSAM application program through remote VSAM access services. These locks can affect the availability of this data to online transactions. If the program issues a checkpoint call, AR/CTL issues a SYNC call to CICS to release locks, making data available to online transactions. AR/CTL also issues a SYNC call at program termination.

If the batch VSAM application program does not issue checkpoint calls, CICS continues to hold the resource locks until program termination. Degraded performance and deadlock situations can result. To prevent these problems, BMC Software recommends that you use the checkpoint/restart services of AR/CTL; if you are not concerned with restart issues, you can use the automatic syncpoint options of AR/CTL instead of checkpoint/restart services.

Overflow of the CICS system log

Long-running transactions, such as those that may result from using remote VSAM access services, can cause a problem with the CICS system log. CICS uses two data sets to contain the system log. When one data set becomes full, CICS switches to the other data set; when that data set becomes full, CICS switches back to the first data set, and so on. In an environment that is set up for recovery, the log must be archived.
Before the alternate data set is reused, in an environment that is not set up for recovery, long-running transactions can cause CICS to begin writing records for the transaction to one data set, switch to the other data set and write transaction records to it, then switch back to the first data set and write transaction records. If CICS abends after this activity, an emergency restart will not be possible.

You should evaluate the size of the CICS system log to ensure that the data sets are large enough to accommodate the activity caused by batch VSAM application programs. To prevent system log overflow, BMC Software recommends that you use the checkpoint/restart services of AR/CTL; if you are not concerned with restart issues, you can use the automatic syncpoint options of AR/CTL instead of checkpoint/restart services.

Using automatic syncpoints

For application programs that use remote VSAM access (file sharing) only, AR/CTL can issue syncpoints automatically, based on criteria you set up externally to the application program. This section describes how to use automatic syncpoints in a VSAM file-sharing-only application program.

Restart issues

Automatic syncpoints can help prevent CICS resource usage problems inherent with file sharing without requiring changes to the application program; however, a job step using automatic syncpoints cannot be restarted. Before you decide to use automatic syncpoints, you must consider the significant problems that result from failure of an application program that uses automatic syncpoints.

A syncpoint causes CICS to release the locks held on the VSAM records updated by the application program and to clear the data in the dynamic transaction backout buffer. Other programs and transactions sharing the VSAM data set are then free to access and (possibly) update the records.

If a job step fails while it is using file sharing and checkpoint/restart, CICS backs out any uncommitted data since the last checkpoint and you restart the job step at the last completed checkpoint. If a job step fails while it is using file sharing but uses neither checkpoint/restart nor automatic syncpoints, CICS backs out the data to the beginning of the job step and you must start the job step over from the beginning.

If a job step fails while it is using file sharing and automatic syncpoints, the effort to recover from the failure can consume significant time and resources. In this case, you must deal with the effects of the failure on the failed job step and on all other programs and transactions that were accessing the data updated by the failed program.
You can restore all files updated by the failed job step and all files updated by all other affected programs and transactions; then you can start the failed job step and all other affected job steps from the beginning. Or you can make temporary changes to the application program to make the program start processing from the last syncpoint. Either method carries a risk of data integrity problems. Use of checkpoint/restart services avoids these problems.

**Requirements for automatic syncpoints**

To use automatic syncpoints, the VSAM File Sharing Only processing option must be set to Y for the application program execution. This option requires the application program to use no AR/CTL services except remote VSAM access services. For more information, see the *APPLICATION RESTART CONTROL Reference Manual*.

**Actions of automatic syncpoints**

When making the decision to issue an automatic syncpoint, AR/CTL compares the current status against all control options, which may be set with different values in each CICS region that shares files with the application program. Any option that indicates the need for an automatic syncpoint causes AR/CTL to issue one. The automatic syncpoint is performed in all CICS regions that share files with the application program.

When an automatic syncpoint is needed, AR/CTL allows any VSAM read-for-update request in progress to complete. The automatic syncpoint causes CICS to release locks, clear the DTB log, and reset all control option counters. Then AR/CTL issues a message to indicate that an automatic syncpoint has occurred.

**Methods for controlling automatic syncpoints**

You can use one, some, or all of the following methods to tell AR/CTL when to issue an automatic syncpoint:

- **Automatic syncpoint trigger file**
  AR/CTL can issue an automatic syncpoint based on I/O activity against a file that the application program is accessing through VSAM file sharing. You identify the ddname of the file with the Autosync Trigger DDname processing option. You set the number of I/O requests that will cause an automatic syncpoint with the Autosync Trigger Count processing option.

- **Maximum DTB KByte threshold**
  AR/CTL can issue an automatic syncpoint based on the amount of data that CICS is storing in the DTB buffer for the application program. You set the number of kilobytes stored in the DTB buffer that will cause an automatic syncpoint with the Maximum DTB KByte Threshold option in the CICS APPLID table.
Maximum locks threshold
AR/CTL can issue an automatic syncpoint based on the number of locks that CICS is holding for the application program. You set the number of locks that will cause an automatic syncpoint with the Maximum Locks Threshold option in the CICS APPLID table. Locks are counted for update requests and not read-for-update requests.

Using ASAM services through GSAM replacement

This section describes the ASAM services that are available through GSAM replacement and summarizes the requirements for implementing ASAM services through GSAM replacement. GSAM replacement is provided with AR/CTL for IMS. It is for use in IMS and IMS-compatible application programs.

NOTE
If an IMS program uses a program option member that contains any data set option members specifying sequential interception, local VSAM access, or remote VSAM access, AR/CTL forces the use of GSAM replacement.

GSAM replacement call overview

To use GSAM replacement, an application program issues IMS-format GSAM calls. These calls are identical to IMS GSAM calls; if the program already uses GSAM calls, no change is necessary.

Keep the following information in mind as you set up the calls:

- AR/CTL replaces GSAM with ASAM for all GSAM data sets accessed by the job step; replacement is not selective by data set.

- At restart of a program that is using only ASAM services, the program does not need to open an ASAM data set if the data set was open at the time of the last checkpoint.

- If the data set uses output staging and a failure occurs, AR/CTL discards any records written since the last checkpoint (if the staging area has not overflowed). If the data set does not use output staging and a failure occurs, the records written since the last checkpoint may be present in the data set before restart. The records are overwritten after the restart.
Sequential data set repositioning requires each volume of a multi-volume output tape data set to be mounted and unloaded. Although only the last volume is read, the other volumes must be mounted to keep the catalog and tape headers correct and complete for future use. This processing is consistent with GSAM processing.

AR/CTL GSAM replacement supports the use of multi-volume GSAM data sets. Make sure that the FCB is defined using record level repositioning.

Data set options

A data set option member (FCB) can be created for the data set. If you do not set up a data set option member, AR/CTL uses default values for the data set options as described in the APPLICATION RESTART CONTROL Reference Manual. For a GSAM-replacement data set that uses an IMS GSAM DBD, the default value for the Repositioning Options is RECORD. A program options member is not necessary for GSAM replacement; the program continues to use the IMS program specification block (PSB).

The name of the data set option member must be the same name as the GSAM database description (DBD) in the PCB that the application program uses with IMS. AR/CTL searches the AR/CTL control block libraries (ddname ARC) for a data set option member with the same name as the GSAM DBD. If AR/CTL finds this member, AR/CTL uses it instead of the IMS DBD; the data set option member completely replaces the GSAM DBD. If AR/CTL does not find this member, AR/CTL uses the GSAM DBD from the IMS control block libraries (ddname IMS).

To set up data set option members, you can use the AR/CTL ISPF interface, as described in the APPLICATION RESTART CONTROL Reference Manual.

Processing option values

To enable GSAM replacement, the Replace IMS GSAM with ASAM processing option must be set to Y. The default value is N.
Using ASAM services through the ASAM callable interface

This section describes the ASAM services of AR/CTL that are available through the ASAM callable interface and summarizes the requirements for implementing ASAM services through the ASAM callable interface. The ASAM callable interface is available with AR/CTL for DB2 and AR/CTL for VSAM. The callable interface supports physical sequential data sets, VSAM input and output ESDSs, and VSAM input KSDSs.

ASAM callable interface supported data set types

AR/CTL supports the following types of sequential data sets through the ASAM callable interface:

- tape input and output with repositioning on the basis of block count or record count
- sequential input and output on DASD
- print (output staging is especially valuable in simplifying the programming effort for print data sets)
- VSAM ESDS for input and output
- VSAM KSDS for input
- generation data groups (GDGs) for input and output

Some vendor products do not allow DASD data sets using the MVS NOTE/POINT macros to become multi-volume data sets. AR/CTL supports and can reposition multi-volume ASAM data sets.

ASAM call

To request ASAM services through the ASAM callable interface, an application program must issue ARCASAM calls. If you are replacing an existing sequential data set with ASAM, change the application program to remove any open calls, close calls, and access request calls (such as get, put, read, and write). Then make all data requests directly to AR/CTL using the ASAM callable interface.
Keep the following information in mind as you set up the calls:

- At restart of a program that is using ASAM services only, the program does not need to open an ASAM data set if the data set was open at the time of the last checkpoint.

- If the data set uses output staging and a failure occurs, AR/CTL discards any records written since the last checkpoint (if the staging area has not overflowed). If the data set does not use output staging and a failure occurs, the records written since the last checkpoint may be present in the data set before restart. The records are overwritten after the restart.

- Sequential data set repositioning requires each volume of a multi-volume output tape data set to be mounted and unloaded. Although only the last volume is read, the other volumes must be mounted to keep the catalog and tape headers correct and complete for future use.

### Data set and program options

A data set option member (FCB) must be created for each data set that the application program will access through ASAM services through the ASAM callable interface. A program option member (ASB) must be created to refer to the data set option members that the application program will use. For more information, see the *APPLICATION RESTART CONTROL Reference Manual*.

### Copying an ASAM data set

This section describes how to use the ASAM Copy utility to copy or move an ASAM or GSAM data set. This utility preserves the ability to reposition the data set during application restart. It uses the basic sequential access method (BSAM) to read an input data set and write an identical copy of the data set, maintaining the exact block content and relative block counts.

You can increase the space in the copy and use a different volume than the original. However, you must not change the data control block (DCB) parameters (record format, block size, logical record length). If you change any DCB parameters, AR/CTL cannot use high-speed repositioning techniques for this data set at restart. However, AR/CTL can use record-level repositioning if the Reposition Options in the data set option member (also known as the FCB) are set to **RECORD**.
JCL requirements

Figure 7 shows sample JCL for executing the ASAM Copy utility.

**Figure 7   ASAM Copy utility JCL**

```
//ASAMCOPY JOB standard job statement
//ARCTRC EXEC PGM=ARCYRC00,REGION=4096K
//STEPLIB DD DSN=BMC.AES.LOAD,DISP=SHR
// DSN=BMC.ARC.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=* 
//ARCPRINT DD SYSOUT=* 
//indd DD DSN=arc.asam.dsn,DISP=SHR
//outdd DD DSN=arc.asam.copy,DISP=(NEW,CATLG,DELETE),
// UNIT=SYSDA,SPACE=(CYL,(1,1)) 
//SYSIN DD *
FUNCTION=COPY,INDD=indd,OUTDD=outdd
//
```

**EXEC**

Required. The program name is ARCYRC00.

**STEPLIB DD**

Required. Defines load module libraries defined to the job step. Include the libraries containing the AR/CTL load modules.

**SYSPRINT DD**

Required. Defines the standard output data set to contain system and utility messages. It usually defines a SYSOUT-type data set.

**ARCPRINT DD**

Optional. Defines an output data set to contain the utility reports. If omitted, the utility writes its reports to the SYSPRINT data set. It usually defines a SYSOUT-type data set.

**indd DD**

Required. Defines the data set name of the input ASAM or GSAM data set to copy. The ddname must match the value of the INDD keyword in the SYSIN data set; the default value is ASAMIN.

**outdd DD**

Required. Defines the data set name of the output copy of the ASAM or GSAM data set. The ddname must match the value of the OUTDD keyword in the SYSIN data set; the default value is ASAMOUT. You can preallocate the data set, or the utility can create it as NEW. You can use a different space allocation, and you can use a different volume if it is of the same device type as the original.
For ASAM data sets, the utility can create a copy on a different device type than the original. At application restart, AR/CTL detects the difference and repositions the data set with a block-count read operation instead of with the MVS POINT macro services. IMS does not detect the difference; therefore, do not copy a GSAM data set to a different device type than the original.

**WARNING**

Do not change the DCB parameters from the original data set. You can omit the DCB parameters to use the DCB from the input data set.

**SYSIN DD**

Required. Defines the data set containing the control statement keywords.

**SYSIN keywords**

The following keywords are valid in the SYSIN data set for the ASAM Copy utility:

- **FUNCTION**
- **INDD**
- **OUTDD**

**FUNCTION**

Required. Defines the function to perform. To copy an ASAM data set, use the value **COPY**.

**INdd**

Required unless you use the default ddname **ASAMIN** on the input data set DD statement. Defines the ddname of the DD statement that describes the existing input data set to copy.

**OUTdd**

Required unless you use the default ddname **ASAMOUT** on the output data set DD statement. Defines the ddname of the DD statement that describes the new output data set.
Completion codes

The ASAM Copy utility can issue the following return and abend codes at completion:

0
   Execution was successful.

U1771
   The utility detected one or more errors. The SYSPRINT data set contains messages that explain the error.

ASAM Copy utility report

The ASAM Copy utility writes the Execution Summary report (Figure 8) to the ARCPRIINT data set (or to SYSPRINT if ARCPRIINT was not defined in the JCL). The fields on this report are self-explanatory. The number of blocks for the input and output data sets must be identical; otherwise, the data set cannot be successfully repositioned at application restart.

Figure 8 Execution Summary Report

<table>
<thead>
<tr>
<th>FUNCTION TYPE</th>
<th>COPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT DATA SET</td>
<td>BMC.ARCX.ARM13182.ASAM</td>
</tr>
<tr>
<td>OUTPUT DATA SET</td>
<td>BMC.ARCX.COPY</td>
</tr>
<tr>
<td>NUMBER OF BLOCKS ON INPUT</td>
<td>.00000418</td>
</tr>
<tr>
<td>NUMBER OF BLOCKS ON OUTPUT</td>
<td>.00000418</td>
</tr>
<tr>
<td>NUMBER OF VOLUMES IN INPUT DATA SET</td>
<td>.00000001</td>
</tr>
<tr>
<td>NAME OF VOLUMES IN INPUT DATA SET</td>
<td>DEV225</td>
</tr>
<tr>
<td>NUMBER OF VOLUMES IN OUTPUT DATA SET</td>
<td>.00000001</td>
</tr>
<tr>
<td>NAME OF VOLUMES IN OUTPUT DATA SET</td>
<td>DEV150</td>
</tr>
</tbody>
</table>
Using AR/CTL operational services

This chapter describes how to use the operational services available with APPLICATION RESTART CONTROL (AR/CTL). This chapter contains the following information:

- Overview ........................................ 138
- Using AR/CTL suspend-and-resume interfaces ........................................ 138
  - Interface with BMC Software products for IMS .................................. 140
  - Interface with the REORG PLUS for DB2 Online Feature ....................... 143
- Using early termination support ....................................................... 145
  - Uses for early termination support ............................................... 146
  - Backout with early termination ................................................... 146
  - Processing options for early termination ....................................... 147
  - OCI commands for early termination ............................................. 147
  - Early termination support with an IMS application program .................. 148
- Using on-demand snap dumps ....................................................... 149
- Using enhanced call tracing ......................................................... 149
  - Overview of enhanced call tracing ............................................... 150
  - Allocating the ARCDLTRC data set .............................................. 151
  - Starting and stopping enhanced call tracing .................................... 152
  - Using trace records ................................................................. 154
- Formatting trace data ................................................................. 155
  - JCL requirements ........................................................................ 155
  - SYSIN keywords ........................................................................ 157
- Using formatted trace reports .......................................................... 160
  - General information ..................................................................... 161
  - IMS Trace Header Information report ............................................ 161
  - DB2 Trace Header Information report ............................................. 162
  - IMS Trace Detail Information report ............................................... 163
  - DB2 Trace Detail Information report ............................................... 166
  - DB2 Trace Trailer Information report ............................................. 167
  - IMS Trace Trailer Information report ............................................. 167
  - Trace Summary report .................................................................. 168
  - Dump Format reports .................................................................... 168
Overview

The operational services of AR/CTL are available through the operator communications interface (OCI), through processing options (in some cases), or both. The OCI consists of a set of MVS MODIFY commands you can issue from the MVS system console or on the console log.

The following AR/CTL operational services are available:

- With checkpoint/restart services, AR/CTL offers a suspend-and-resume interface to other BMC Software products. Suspend-and-resume processing allows these products to obtain a point of consistency required for reorganization or recovery. For more information, see “Using AR/CTL suspend-and-resume interfaces” on page 138.

- Early termination support allows you to terminate an application execution at a scheduled time of day (after a completed checkpoint) or on command at any of several points during program processing. You can obtain some early termination support services through the OCI and others through processing options. For more information, see “Using early termination support” on page 145.

- An on-demand snap dump, available through the OCI, is a standard MVS snap dump of application program storage. You can use the snap dump to analyze application program problems. The snap dump does not cause program termination. For more information, see “Using on-demand snap dumps” on page 149.

- Enhanced call tracing can help you develop, test, debug, and audit application programs. You can turn on enhanced call tracing with a processing option or with the OCI. To turn off enhanced call tracing, you can use the OCI or you can let tracing continue to the end of the job step. For more information, see “Using enhanced call tracing” on page 149.

Using AR/CTL suspend-and-resume interfaces

This section describes the suspend-and-resume interfaces that AR/CTL provides for the following BMC Software products:

- Backup and Recovery Solution for IMS
- CONCURRENT REORG for IMS
- IMAGE COPY PLUS
- MAXM Reorg/Online for IMS
- RECOVERY MANAGER for IMS
- REORG PLUS for DB2 Online Feature
Suspend-and-resume overview

Suspend-and-resume processing allows a BMC Software product to obtain a point of consistency required for reorganization or recovery. (In recovery products, the process of obtaining this point of consistency is known as Get and Hold Recovery Point or Hold Point of Consistency.)

Using a combination of early termination and application reattach support, AR/CTL suspends batch application processing when needed so that the other BMC Software product can obtain a point of consistency. After this point is obtained, AR/CTL restarts the application program. Suspend-and-resume processing is automatic and transparent to the application program.

Batch application programs that are not under the control of AR/CTL cannot participate in suspend-and-resume processing.

AR/CTL suspend processing

To perform suspend processing, AR/CTL issues a user abend 1551 with return code X’00074436’ for the batch application program task, but the job step remains active in the MVS system. This user abend takes place after completion of the next application checkpoint issued by the batch program. The DBMS detects that the task has abended and releases resources. No backout is necessary because the task abends on a checkpoint boundary. Suspend processing occurs at the first checkpoint that the batch program issues after the suspend request, even if the checkpoint pacing decision is made to bypass the checkpoint. (This logic is different from early termination logic, in which early termination waits until the checkpoint pacing decision requires checkpoint processing to be performed.)

Suspending the batch programs should take less than a minute, excluding the time waiting for the batch programs to issue checkpoints.

AR/CTL resume processing

To perform resume processing, AR/CTL automatically reattaches the batch program task, and the batch program resumes processing at the checkpoint that was completed before the program was suspended. Reattaching the batch programs should take less than a minute. The reattach limit is not affected by reattaching a suspended batch program.
Interface with BMC Software products for IMS

The suspend-and-resume interface with BMC Software products for IMS works for automatic management of batch message processing (BMP) programs that are performing a process (such as a reorganization, a batch image copy, or a hold point of consistency function) that requires temporary exclusive access to a database through a /DBR (database recovery) request. AR/CTL suspends BMP processing to allow exclusive access and resumes BMP processing when exclusive access is no longer needed.

The suspend-and-resume interface was implemented first for the CONCURRENT REORG for IMS product. This support was extended to the BMC Software Image Copy utility (which is available with the Backup and Recovery Solution for IMS product and the IMAGE COPY PLUS product) through CONCURRENT REORG for IMS. The functionality of CONCURRENT REORG for IMS is also available with the MAXM Reorg/Online for IMS product.

The suspend-and-resume interface was adapted for the Hold Point of Consistency (HPC) function (which is available with the Backup and Recovery Solution for IMS product and the RECOVERY MANAGER for IMS product). The adapted interface works somewhat differently than the interface for CONCURRENT REORG for IMS. During HPC processing, the function must quiesce the database through a /DBR or /DBD request. AR/CTL suspends BMP processing to allow exclusive access and resumes BMP processing when exclusive access is no longer needed.

Requirements

The suspend-and-resume interface with BMC Software products for IMS has the following requirements:

- A license for AR/CTL for IMS is required.
- The other BMC Software product CPU authorization password must be available to the BCSS.
- In the MVS systems you use to execute the BMC Software product, the BCSS must be active and the AR/CTL IMS component of the BCSS must be initialized. In the MVS system you use to execute the BMPs, the BCSS must be active and the AR/CTL IMS component of the BCSS must be initialized. If the BMC Software product and BMPs execute on different MVS systems, both of these BCSSs must share the same REGISET.
- BMPs must use AR/CTL checkpoint/restart services.
- AR/CTL execution load modules must be available to the other BMC Software product job step. For the HPC function, the AR/CTL execution load modules must be available to the Recovery Manager started task or job.
A BMP cannot use the Ignore Job Name processing option of AR/CTL.

**CONCURRENT REORG interface processing**

The suspend-and-resume interface with CONCURRENT REORG for IMS works as follows for a database reorganization process. (The process for the Snapshot function of the Image Copy utility is similar.)

1. BMP job steps execute under the control of AR/CTL. Each BMP starts (or restarts) as usual.

2. CONCURRENT REORG for IMS initializes and determines which BMPs are accessing the database to be reorganized.

3. CONCURRENT REORG for IMS notifies AR/CTL to suspend processing for these BMPs.

4. AR/CTL performs suspend processing as described in “AR/CTL suspend processing” on page 139.

5. CONCURRENT REORG for IMS issues a /DBR request and waits until the request is successful. If it is not, CONCURRENT REORG for IMS issues a message to the operator.

6. When all databases have been stopped, CONCURRENT REORG for IMS enables the capture of updates for the database and issues /STA requests to start the database, BMP programs, and BMP transactions. Then CONCURRENT REORG for IMS notifies AR/CTL to resume processing on the suspended BMPs.

7. The BMP resumes processing at the checkpoint that was completed before the BMP was suspended.

8. CONCURRENT REORG for IMS begins reorganization and captures all updates that are made to the original database during the reorganization process.

9. When the database reorganization is done and CONCURRENT REORG for IMS is ready to swap the reorganized shadow database data sets with the original data sets, it performs the following actions:
   - determines which BMPs are accessing the database being reorganized
   - notifies AR/CTL to suspend these BMPs
   - issues a /DBR request to close and deallocate the database and unauthorize it from DBRC
   - swaps data set names
- issues DBRC commands
- stops recording
- issues /STA requests to start the database, BMP programs, and BMP transactions
- notifies AR/CTL to resume BMP processing

10. CONCURRENT REORG for IMS performs its final processing; the entire reorganization process is complete.

**HPC interface processing**

The following shows how the suspend-and-resume interface for the Hold Point of Consistency (HPC) function works:

1. BMP job steps execute under the control of the AR/CTL application supervisor and AR/CTL. Each BMP starts (or restarts) as usual.

2. The HPC function initializes and determines which BMPs are accessing the databases to be quiesced.

3. The HPC function notifies AR/CTL to suspend processing for these BMPs. AR/CTL performs suspend processing as described in “AR/CTL suspend processing” on page 139.

4. The HPC function waits until the BMPs are suspended or until the “stop BMPs” timeout value has been exceeded. The HPC function issues a task message for each suspended BMP.

5. The HPC function issues a /DBR or /DBD request and waits until the request is successful. If it is not, the HPC function issues a task message indicating that the database could not be quiesced.

6. When all databases have been stopped or the HPC timeout value has been exceeded, the HPC function issues /STA requests to start the databases, BMP programs, and BMP transactions. Then the HPC function notifies AR/CTL to resume processing on the suspended BMPs.

7. AR/CTL performs resume processing as described in “AR/CTL resume processing” on page 139.

8. The HPC function performs its final processing; the entire HPC process is complete.
Implementing the interface

To implement the suspend-and-resume interface with BMC Software products for IMS, perform the following steps:

1. Install and configure AR/CTL as documented in the *Installation System User Guide* and the *APPLICATION RESTART CONTROL Configuration Guide*.

2. Install and configure the other BMC Software products for IMS as documented for those products.

3. Make the CPU authorization password for the other BMC Software product available to the BCSS. You can include the library that contains the password module in the AESPAUTH DD statement concatenation in the BCSS startup procedure, or you can copy the password module to a library that is already in the AESPAUTH DD concatenation.

4. Set up the other BMC Software product batch job step as documented in the manual for that product. Make the AR/CTL execution load module library available to the job step. For the HPC function, set up the Recovery Manager started task or job as documented, and make the AR/CTL execution load module library available to the started task or job. You can add this library to the STEPLIB, JOBLIB, or LNKLST concatenation.

5. In each BMP that you want AR/CTL to control, implement AR/CTL checkpoint/restart services as described in “Getting started with AR/CTL implementation” on page 65.

6. Execute the job step or function.

Interface with the REORG PLUS for DB2 Online Feature

The suspend-and-resume interface with the REORG PLUS for DB2 Online Feature works for automatic management of batch DB2 programs that are accessing a table space that is being reorganized. During a SHRLEVEL CHANGE table space reorganization process, the Online Feature must obtain temporary exclusive access to the table space. AR/CTL suspends batch DB2 processing to allow exclusive access and resumes batch processing when exclusive access is no longer needed.

Requirements

The suspend-and-resume interface with the REORG PLUS for DB2 Online Feature has the following requirements:

- A license for AR/CTL for DB2 is required.
The CPU authorization password for the REORG PLUS for DB2 Online Feature must be available to the BCSS.

In the MVS system that you use to execute the batch DB2 application, the BCSS must be active and the AR/CTL non-IMS component of the BCSS must be initialized.

The batch program must use AR/CTL checkpoint/restart services.

**Interface Processing**

The following shows how the suspend-and-resume interface with the REORG PLUS for DB2 Online Feature works:

1. Batch job steps execute under the control of AR/CTL.
2. When REORG PLUS for DB2 is initialized, it notifies AR/CTL which table spaces and indexes it is using.
3. When an application program attempts an SQL access to a table that would normally receive an SQLCODE -911 or -904 because of REORG PLUS, AR/CTL performs suspend processing for the application as described in “AR/CTL suspend processing” on page 139.
4. When REORG PLUS completes the function that would cause the SQLCODE -911 or -904, it signals AR/CTL to resume processing on the application unit of work (as described in “AR/CTL resume processing” on page 139).

The following REORG PLUS functions can cause AR/CTL to suspend an application:

- table space QUIESCE processing during any phase
- all processing during the LOGAPPLY phase
- all processing during the LOGFINAL phase
- rename processing during the UTILTERM phase

**Implementing the interface**

To implement the suspend-and-resume interface with the REORG PLUS for DB2 Online Feature, perform the following steps:

1. Install and configure AR/CTL as documented in the *Installation System User Guide* and the *APPLICATION RESTART CONTROL Configuration Guide*.
2. Install REORG PLUS for DB2 (which includes the Online Feature) as documented for that product.
Using early termination support

3 Make the CPU authorization password for the REORG PLUS for DB2 Online Feature available to the BCSS. You can include the library that contains the password module in the AESPAUTH DD statement concatenation in the BCSS startup procedure, or you can copy the password module to a library that is already in the AESPAUTH DD concatenation.

4 Set up the REORG PLUS for DB2 Online Feature job step as documented for that product.

5 In each batch program that you want AR/CTL to control, implement AR/CTL checkpoint/restart services as described in “Getting started with AR/CTL implementation” on page 65.

6 Execute the reorganization job.

Using early termination support

Early termination support allows you to terminate an application job at a scheduled time of day (after a completed checkpoint), on command after a checkpoint, or on command immediately after the next call. Termination after a checkpoint requires no backout because the checkpoint signifies that the changes are committed. Immediate termination requires backout of uncommitted changes. In both cases, all restart information is left intact and ready for restarting the job at the last completed checkpoint.

NOTE

Specify ARC=YES in the installation options.

NOTE

The application program must use checkpoint/restart services for AR/CTL to provide early termination support.
Uses for early termination support

Early termination support can be useful in many situations, including these:

- You can stop a batch application program at the end of the batch window, in preparation for restarting it at a more appropriate time or as an online application program, and you can stop an online program (such as a BMP program) and restart it as a batch program (such as a DLI program).

- In a VSAM environment, you can use early termination support to stop a batch VSAM program that uses the local VSAM access services of AR/CTL, then restart the program to use remote VSAM access services. Or you can stop a program that uses remote VSAM access services, then restart the program to use local VSAM access services.

- You can stop an IMS batch message processing (BMP) program without the risk of abending the IMS control region.

- You can test the restart logic of an application program.

- You can stop the executing jobs in an MVS system to prepare for system maintenance; after normal operations resume, you can restart and complete the executing jobs.

Backout with early termination

After a STOP IMMEDIATE command or as specified with the TRMBEFORECHK option, backout of uncommitted changes is accomplished. The following shows the actions:

- For DL/I batch application programs, you can use the BATCH CONTROL FACILITY (BCF) component of AR/CTL to perform backout automatically. Make sure that the Automatic Batch Backout (AUTOBBO) option in the BCF component is set to Y. If the BCF component is not activated, you can execute the IMS Batch Backout utility manually.

- For VSAM batch application programs that use local VSAM access services, AR/CTL can perform backout automatically when the Perform VSAM Dynamic Backout (VSAMDDBO) option in AR/CTL is set to Y.

- IMS automatically performs database backout processing for BMP job steps.

- DB2 automatically performs dynamic backout of uncommitted changes to DB2 tables.
CICS automatically performs backout processing for VSAM data sets that the job step is accessing through remote VSAM access services.

Processing options for early termination

AR/CTL provides the following processing options that you can use to control early termination:

- Early Termination by Time of Day (ISPF panel field) and TRMA TTOD (ARCSYSIN keyword)
- Early Termination Time (ISPF panel field) and TRMATTIME (ARCSYSIN keyword)
- TRMAFTERMIN (ARCSYSIN keyword)
- TRMAFTERCKP (ARCSYSIN keyword)
- TRMBEFORCKP (ARCSYSIN keyword)
- TRMBEFORMAX (ARCSYSIN keyword)
- TRMAFTERMAX (ARCSYSIN keyword)

For more information, see the APPLICATION RESTART CONTROL Reference Manual.

OCI commands for early termination

The OCI provides two commands for early termination support:

STOP
Abnormally terminate the application program when the current or next application program checkpoint completes. If a checkpoint is bypassed because of checkpoint pacing, the STOP command takes effect at the next checkpoint that is not paced.

STOP IMMEDIATE
Abnormally terminate the application program when the current application program call completes or the next application program call is issued.
Examples

The following commands illustrate how to use early termination support from the System Display and Search Facility (SDSF) and the console:

- From SDSF, terminate job APPLTEST when the current or next checkpoint completes:

  ```
  /F APPLTEST,STOP
  ```

  You must be authorized to issue this command from SDSF.

- From the console, stop the job PAYROLL1 when the current or next application program call completes:

  ```
  F PAYROLL1,STOP IMMEDIATE
  ```

Early termination support with an IMS application program

You can use early termination support to stop an IMS application program and restart it with a different execution parameter (DLI or BMP). To prepare the program for switching back and forth, perform the following steps:

1. Before the job step is restarted, change the `PARM` value on the EXEC statement from BMP to DLI if the job was executing as a BMP, or change it from DLI to BMP if the job was executing as a batch program. You may be able to automate this action by using a different set of JCL, as long as the job name, job step name, procedure step name, PSB name, and program name are the same in both job steps. Other parameters specific to BMP or DLI processing may also need to be changed.

2. Make sure that the PSB is generated with `CMPAT=YES`.

3. Make sure that the program defines the necessary PCBs to run as a BMP. Otherwise, the application program can use the same logic.
Using on-demand snap dumps

This section describes how to obtain a standard MVS snap dump of application program storage by using the OCI SNAP command.

The following command is an example (PAYROLL1 is the job name):

```
F PAYROLL1.SNAP
```

When you enter the SNAP command, AR/CTL issues the MVS SNAP macro with the DDATA=ALL parameter at completion of the next AR/CTL or DL/I call. AR/CTL dynamically allocates the snap dump output data set to the default SYSOUT class. For a complete description of the SNAP macro, see the IBM MVS/ESA Supervisor Services and Macro Instructions manual.

An on-demand snap dump does not cause program termination.

Using enhanced call tracing

AR/CTL provides enhanced call tracing to help you develop, test, debug, and audit application programs. This section describes how to implement enhanced call tracing in an application program.

To use enhanced call tracing, you can allocate the trace data set (ddname ARCDLTRC) in the JCL, or AR/CTL can allocate it dynamically from an ARCDLTRC allocation member. Then you must use the OCI or the TRACE keyword in the ARCSYSIN control statement data set to tell AR/CTL when to start tracing. To stop tracing before the end of the job step, you must use the OCI. The following sections discuss these topics in detail.

**NOTE**

Although enhanced call tracing has an insignificant effect on overall system performance, it can degrade application program performance during tracing. You may want to use it only for diagnosing program problems or only during a portion of the program execution.
Overview of enhanced call tracing

Enhanced call tracing provides essential information about the DL/I, IMS Fast Path, SQL, and AR/CTL calls issued by an application program. Unlike many existing trace facilities, enhanced call tracing allows you to begin and end tracing while the application program is executing. Enhanced call tracing also places the trace data in a separate data set for easy access and does not significantly degrade overall system performance; however, enhanced call tracing may degrade performance in the application program address space.

**NOTE**

AR/CTL does not trace program requests that AR/CTL intercepts through dynamic call interception. AR/CTL provides sequential file interception, local VSAM access, and remote VSAM access services through dynamic call interception. AR/CTL traces SQL calls only if the program uses restart services; tracing cannot begin until after restart services are invoked.

Enhanced call tracing fits into the standard application program environment as a function of AR/CTL (see Figure 9).

**Figure 9  Enhanced call tracing structures**

Calls

During application program processing, the application program issues calls. AR/CTL intercepts these calls to produce trace data.

Trace records

When enhanced call tracing is active, AR/CTL builds trace records to capture pertinent information about calls.
Allocating the ARCDLTRC data set

Trace data set
AR/CTL writes the trace records to the trace data set (ddname ARCDLTRC). AR/CTL can allocate the trace data set dynamically with information from the ARCDLTRC allocation member of the REGISET, or the job step can explicitly allocate the trace data set in the job step JCL.

Processing options, ARCSYSIN control statements, or OCI
You can turn on enhanced call tracing with the Enhanced Call Trace option, with the TRACE keyword in the ARCSYSIN data set, or with the OCI. To turn off enhanced call tracing, you can use the OCI. Or you can let tracing continue to the end of the job step.

Trace Format utility and trace reports
AR/CTL provides the Trace Format batch utility, which formats or extracts the trace records you select. The formatted records are easier to read than the unformatted records. You can also use the unformatted trace records as input to a statistical analysis package or audit program.

Allocating the ARCDLTRC data set

AR/CTL writes the output of enhanced call tracing to the trace data set (ddname ARCDLTRC). AR/CTL can allocate this data set dynamically from the dynamic allocation record for ddname ARCDLTRC in the REGISET. For information about setting up this record, see the APPLICATION RESTART CONTROL Reference Manual. BMC Software recommends the dynamic allocation method.

NOTE
The REGISET must contain the global ARCDLTRC dynamic allocation record. This record is created during AR/CTL configuration.

If you don’t want to use a dynamic allocation record to allocate the trace data set, you can include the ARCDLTRC DD statement in the job step JCL. The following DD statement is an example:

```plaintext
//ARCDLTRC DD DSN=data.set.name, DISP=(NEW,CATLG,CATLG), UNIT=SYSDA,VOL=SER=volser, SPACE=(CYL,(2,1),RLSE), DCB=(DSORG=PS,RECFM=VB,LRECL=32756, BLKSIZE=32760,BUFNO=5)
```
AR/CTL automatically stops tracing if the ARCDLTRC data set becomes full. AR/CTL creates one trace record for each call; the potential size of the data set depends on the number of calls issued by the application program between the time that the trace begins and the time it ends, and on the parameters of the calls. If size is an issue, use a tape data set. You can use the volume count parameter to select the number of tape volumes to allow.

Use the following DCB parameters for the trace data set:

- The data set organization must be physical sequential (DSORG=PS).
- The record format must be variable blocked (RECFM=VB).
- AR/CTL forces a logical record length (LRECL) of 32756.
- AR/CTL forces a block size (BLKSIZE) of 32760.
- If you are concerned with performance, you can increase the number of buffers (BUFNO) allocated to the trace. If you increase the number of buffers, ensure that the job step has a sufficient region size for the buffers.

Starting and stopping enhanced call tracing

To control enhanced call tracing, you can use one or more of the following methods:

- Insert the TRACE=Y keyword in the ARCSYSIN data set. The only use for this option is to start tracing. For information about setting this option, see the APPLICATION RESTART CONTROL Reference Manual.

- Issue tracing commands with the OCI. The OCI consists of a set of MVS MODIFY commands issued from the MVS system console or on the console log. You can use these commands to start tracing, to stop tracing, and to display the tracing status.

**NOTE**

To use the OCI, the Deactivate Operator Communications Interface processing option must be set to N (the default); for more information, see the APPLICATION RESTART CONTROL Reference Manual.

The following commands control enhanced call tracing from the OCI:

**DISPLAY TRACE**

Use the DISPLAY TRACE command to display the current status of the enhanced call tracing process. AR/CTL writes message BMC74178I, BMC74179I, BMC74257I, or BMC74258I to the log or console. For information about these messages, see the APPLICATION RESTART CONTROL messages in the BMC Documentation Center.
SETTRACEON

Use the SET TRACE ON command to begin tracing. You can include the following keywords to further control tracing:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLI</td>
<td>Trace IMS DL/I, IMS Fast Path, and AR/CTL calls.</td>
</tr>
<tr>
<td>SQL</td>
<td>Trace SQL calls.</td>
</tr>
<tr>
<td>ALL</td>
<td>Trace IMS DL/I, IMS Fast Path, AR/CTL, and SQL calls.</td>
</tr>
<tr>
<td>START nnnnnnnn</td>
<td>Begin tracing after the application program issues the specified number of calls. The number can be an integer from one to nine digits. Tracing must be inactive when you issue a command with this keyword. If you do not use the START keyword to delay activity, tracing starts with the restart call.</td>
</tr>
<tr>
<td>FOR nnnnnnnn</td>
<td>End the tracing process after tracing the specified number of calls. The number can be an integer from one to nine digits. Tracing must be inactive when you issue a command with this keyword. If you do not use the FOR keyword to end tracing, tracing ends at job step termination.</td>
</tr>
</tbody>
</table>

SET TRACE OFF

Use the SET TRACE OFF command to end tracing. AR/CTL ends tracing even if the trace status is PENDING.

Examples

The following commands illustrate how to control tracing in several situations:

- From SDSF, start tracing for job APPLTEST (trace SQL calls only, beginning with the next SQL call):

  
  /F APPLTEST,SET TRACE ON SQL

  You must be authorized to issue this command from SDSF.

- From the console, display the tracing status of job PAYROLL1:

  
  F PAYROLL1,DISPLAY TRACE

- In the ARCSYSIN data set, include the keyword to turn on tracing for all application program calls:

  
  //ARCSYSIN DD *
  TRACE=Y
  /*
Using trace records

Table 5 shows the types of trace records AR/CTL can write to the ARCDLTRC data set, the record code that identifies the type of record, the member of the sample library that contains a dummy section (DSECT) describing the record, and the page number of a sample of the formatted record.

Table 5  Trace record types

<table>
<thead>
<tr>
<th>Record type</th>
<th>Record code</th>
<th>Sample library member</th>
<th>Page</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS trace header</td>
<td>X'01'</td>
<td>$ARCITRH</td>
<td>page 161</td>
<td>one record for each trace if IMS or AR/CTL tracing is active</td>
</tr>
<tr>
<td>IMS trace detail</td>
<td>X'02'</td>
<td>$ARCITRD</td>
<td>page 163</td>
<td>one record for each IMS or AR/CTL call, beginning after tracing becomes active for these calls</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Each record maps up to 18 program call parameters.</td>
</tr>
<tr>
<td>IMS trace trailer</td>
<td>X'03'</td>
<td>$ARCITRT</td>
<td>page 167</td>
<td>one record for each trace if IMS or AR/CTL tracing is active</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This record is present only if AR/CTL trace termination is successful.</td>
</tr>
<tr>
<td>DB2 trace header</td>
<td>X'04'</td>
<td>$ARC2TRH</td>
<td>page 162</td>
<td>one record for each trace if DB2 tracing is active</td>
</tr>
<tr>
<td>DB2 trace detail</td>
<td>X'05'</td>
<td>$ARC2TRD</td>
<td>page 166</td>
<td>one record for each DB2 call, beginning after tracing becomes active for DB2 calls</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If the application program uses IMS, calls are not traced until after the restart call is issued.</td>
</tr>
<tr>
<td>DB2 trace trailer</td>
<td>X'06'</td>
<td>$ARC2TRT</td>
<td>page 167</td>
<td>one record for each trace if DB2 tracing is active</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This record is present only if AR/CTL trace termination is successful.</td>
</tr>
</tbody>
</table>

**NOTE**

Enhanced call tracing places checkpoint/restart and application sequential access method (ASAM) calls in the category of IMS calls. AR/CTL places the DB2 commit calls (which trigger checkpoint calls) in the category of DB2 calls; however, AR/CTL also reports generated checkpoint calls as IMS calls. Enhanced call tracing does not trace native reads and writes that are dynamically intercepted and translated to ASAM calls.
Formatting trace data

This section describes how to execute the Trace Format utility and use its output. You can use the Trace Format utility to print or copy trace records. Keywords control the selection and format of the records.

JCL requirements

Figure 10 shows JCL for executing the Trace Format utility. Member #ARCTRC0 of the sample library contains a sample of this JCL.

Figure 10  Trace Format utility JCL

```
//ARCTRCFM JOB (acct), 'name', MSGCLASS=x,
//   CLASS=x, NOTIFY=userid
//*
//ARCTRC EXEC PGM=ARCTRC00, REGION=4096K
//STEPLIB DD DSN=BMC.AES.LOAD, DISP=SHR
//   DD DSN=BMC.ARC.LOAD, DISP=SHR
//SYSUDUMP DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//TRACEOUT DD SYSOUT=*  
//TRACECOP DD DSN=bmc.arctrcfm.copy, SPACE=(CYL.(1,1)),
//   DISP=SHR, DCB=BLKSIZE=32760
//TRACEIN DD DSN=bmc.arctrace.data, DISP=SHR
//*
//SYSIN DD *
control statement
//*
```

EXEC
Required. The program name is ARCTRC00. BMC Software recommends a region of at least 4096 KB.

STEPLIB DD
Required. Include the AR/CTL execution library in the STEPLIB or JOBLIB concatenation.

SYSUDUMP DD
Optional. Include this statement to obtain a system dump in case of an abend. This data set typically is defined as SYSOUT=*.

SYSPRINT DD
Required. The Trace Format utility writes messages to this data set. This data set typically is defined as SYSOUT=*.
TRACEOUT DD
Required for printing trace records. This data set contains the output of the print function. The ddname corresponds to the value of the OUTDD keyword. You can use any ddname; however, if you use any ddname other than the default (TRACEOUT), the OUTDD keyword is required in the SYSIN data set.

The output of the print function is a formatted hexadecimal dump or a fully formatted report, depending on the value of the FORMAT keyword. The content of the trace output depends on the type of records (selected with the SUBSYS and PCBNUMBER keywords) and on the subset of data (selected with the DLIDATA keyword).

TRACECOP DD
Required for copying trace records. This data set contains the output of the copy function. The ddname corresponds to the value of the COPYDD keyword. You can use any ddname; however, if you use any ddname other than the default (TRACECOP), the COPYDD keyword is required in the SYSIN data set.

Use the same allocation parameters as those used for the trace data set, as described in “Allocating the ARCDLTRC data set” on page 151.

TRACEIN DD
Required. This data set contains the input records or data to print or copy. The ddname corresponds to the value of the INDD keyword. You can use any ddname; however, if you use any ddname other than the default (TRACEIN), the INDD keyword is required in the SYSIN data set.

If AR/CTL allocated the trace data set dynamically during execution of the application program, the data set name of the trace data set is defined by the DSN option in the dynamic allocation record.

SYSIN DD
Required. This data set contains the utility control statements. It typically is specified instream, although it can be a data set on DASD. The next section describes the control statements you can include in the SYSIN data set.
SYSIN keywords

The control statements for the Trace Format utility use the same format and syntax rules that the ARCSYSIN data set uses, as described in the APPLICATION RESTART CONTROL Reference Manual. You can use the following keywords to control trace printing:

**FUNCTION**

Required. Select the action to perform on the trace data. The table shows possible values. The default value is PRINT.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT</td>
<td>Format the selected records, and print them to the output or print data set that the OUTDD keyword specifies. You can use keywords to select the records to print.</td>
</tr>
<tr>
<td>COPY</td>
<td>Copy the selected records to the trace data set that the COPYDD keyword specifies. The records are not formatted. You can use keywords to copy a subset of the records.</td>
</tr>
</tbody>
</table>

**INdd**

Optional if you use the default ddname TRACEIN; required otherwise. Specify the ddname (one to eight characters) that identifies the input trace data set in the utility JCL. This data set contains the trace records to format. The default value is TRACEIN.

**OUTdd**

Optional if you specify FUNCTION=PRINT and you use the default ddname TRACEOUT; required otherwise. If you specify FUNCTION=COPY, this keyword is not applicable. Specify the ddname (one to eight characters) that identifies the output or print data set in the utility JCL. This data set receives the formatted trace output. The default value is TRACEOUT.

**COPYdd**

Optional if you specify FUNCTION=COPY and you use the default ddname TRACECOP; required otherwise. If you specify FUNCTION=PRINT, this keyword is not applicable. Specify the ddname (one to eight characters) that identifies the copy data set in the utility JCL. This data set receives a copy of the formatted trace output. The default value is TRACECOP.

**PAGE**

Optional. Select the number of lines (1 to 999) to print per page of the report; the default value is 55.
WIDth
Optional. Select the number of columns per line. AR/CTL automatically adds one character to the width for the control character. Set one of the following values; the default value is 120:

- 80
- 120
- 132

TYPe
Optional. Select the type of trace report to produce. The table shows possible values. The default value is FULL.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL</td>
<td>Report individual trace records.</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>Produce the trace summary report only.</td>
</tr>
</tbody>
</table>

SUBsys
Optional. Select only the trace records produced by the indicated subsystem. The table shows possible values. The default value is DLI.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLI</td>
<td>Print trace records for IMS DL/I and Fast Path calls, AR/CTL checkpoint/restart support calls, and ASAM support calls.</td>
</tr>
<tr>
<td>SQL</td>
<td>Print trace records for DB2 calls.</td>
</tr>
<tr>
<td>ALL</td>
<td>Print trace records for all subsystem calls.</td>
</tr>
</tbody>
</table>

STARTaft
Optional. For the selected subsystem types, select the number (one to nine digits) of trace records to bypass before beginning the formatting. To determine when to begin trace formatting, AR/CTL counts trace records only for the requested subsystem types. To start selecting records at the beginning of the trace data set, omit this keyword.

STOPaft
Optional. For the selected subsystem types, select the number (one to nine digits) of trace records to format (starting with the first record selected for formatting) before ending the formatting. To determine when to end trace formatting, AR/CTL counts trace records only for the requested subsystem types. To continue selecting records until the end, omit this keyword.

PCBNumb
Optional. For IMS and AR/CTL calls, select only the trace records for a particular program communication block (PCB). Specify the two-digit number of the PCB that indicates its position within the PSB or ASB; for example, the second PCB is 02. To select all PCBs, omit this keyword.
FORMAT
Optional. Select the format of the printed output. The table shows possible values. The default value is **DUMP**.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUMP</td>
<td>Print the output as a formatted hexadecimal dump.</td>
</tr>
<tr>
<td>YES</td>
<td>Print the output as a formatted report.</td>
</tr>
</tbody>
</table>

**DLIData**
Optional. Limit the type of IMS DL/I or AR/CTL trace data included in the selection. The table shows possible values. The default value is **PCB**:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCB</td>
<td>Select PCB data used with the call.</td>
</tr>
<tr>
<td>CALL</td>
<td>List call parameters used with the call</td>
</tr>
<tr>
<td>KEYFB</td>
<td>List the contents of the key feedback area.</td>
</tr>
<tr>
<td>IOA</td>
<td>List partial contents of the I/O area containing the call results.</td>
</tr>
<tr>
<td>ALL</td>
<td>List all available DL/I trace data.</td>
</tr>
</tbody>
</table>

**Examples**

The following examples illustrate how to use the Trace Format utility control statements in different situations:

- The following control statement selects records from a data set with the ddname `TRACEIN` and prints them in dump format to a data set with the ddname `TRACEOUT`. The utility selects all SQL trace records for printing and uses the default values of the other keywords.

  ```
  FUNCTION=PRINT, INDD=TRACEIN, OUTDD=TRACEOUT,
  SUBSYS=SQL, DLIDATA=KEYFB
  ```

- The following control statement copies records in their original format from a data set with the ddname `OLDTRACE` to a data set with the ddname `NEWTRACE`. The utility selects only SQL records for copying but starts selecting records beginning with SQL record 100 and ending after formatting 200 records (SQL record 300). It uses the default values of the other keywords.

  ```
  FUNCTION=COPY, INDD=OLDTRACE, COPYDD=NEWTRACE,
  SUBSYS=SQL, STARTAFT=99, ENDAFT=200
  ```
The following control statement selects records from a data set with the ddname TRACE, formats them, and prints them to a data set with the ddname REPORTS. The utility selects all records for printing, but prints only the PCB for DL/I trace records. It formats the trace report with 66 lines per page and 80 characters per line. It uses the default values of the other keywords.

```
FUNCTION=PRINT, INDD=TRACE, OUTDD=REPORTS,
FORMAT=YES, SUBSYS=ALL, PAGE=66, WIDTH=80
```

The following control statement reports trace information about the second PCB in the ASB. The utility prints the Trace Summary report only. It reads the trace data set with the ddname TRACE and writes the report to a data set with the ddname REPORTS. It selects DL/I and AR/CTL trace records only and uses the default values of the other keywords.

```
FUNCTION=PRINT, INDD=TRACE, OUTDD=REPORTS,
SUBSYS=DLI, TYPE=SUMMARY, PCBNUMB=02
```

### Using formatted trace reports

If you specify `FUNCTION=PRINT`, the output of the Trace Format utility is a set of reports. This section discusses the reports in the order that the Trace Format utility usually writes them to the output data set.

The Trace Format utility can produce the following types of reports:

- IMS Trace Header Information
- DB2 Trace Header Information
- IMS Trace Detail Information
- DB2 Trace Detail Information
- DB2 Trace Trailer Information
- IMS Trace Trailer Information
- Trace Summary

If you specify `FORMAT=YES`, one report is present for each selected record. Unless you sort the trace data set before processing with the Trace Format utility, IMS and DB2 Trace Detail Information reports for individual calls are listed in the order that they were issued by the application program.

If you specify `FORMAT=DUMP`, the records are formatted as a hexadecimal dump.
General information

Most fields on the trace reports are self-explanatory, with the following exceptions:

- Date fields are reported as Julian dates in yyyy/ddd format, where yyyy is the year and ddd is the day of the year (1 to 366).

- Time fields are reported in hh:mm:ss.th format, where hh is the hour (00 to 23), mm is the minute (00 to 59), ss is the second (00 to 59), t is the tenth second (0 to 9), and h is the hundredth second (0 to 9).

- Fields that report a count of records provide the count of records selected for processing by the utility, not the count of records in the trace data set.

- Fields that report the base product and distribution levels refer to the AR/CTL level that produced the original trace records.

- The TRACE REQUESTOR field provides the ID of the entity that requested the trace. If the processing options turned on tracing, the value is $ARCDIR. If the operator initiated tracing through the OCI, the value is the console number.

- Fields that report an elapsed time are calculated by subtracting the system clock time at the start of the action from the system clock time at the end of the action. This time can include system, subsystem, and other delays that may not be related directly to the action.

- Fields that report user data (such as the key feedback and I/O area) are shown in character format and in hexadecimal format, similar to the display of hexadecimal characters under the ISPF Browse function.

IMS Trace Header Information report

The IMS Trace Header Information report (Figure 11) provides job-related information that does not change during job step execution. The fields on this report correspond to the fields in the IMS trace header record.

Figure 11  IMS Trace Header Information report

+---------------------------------------------+
| IMS TRACE HEADER INFORMATION               |
+---------------------------------------------+

| JOB NAME FROM JOB CARD | ABCMRHA3 |
| JOB STEP NAME          | DBRUN    |
| PROCEDURE NAME         | ARCSTEP  |
| PROGRAM SPECIFICATION BLOCK NAME | ARCASB   |
The DB2 Trace Header Information report (Figure 12) provides information about an application program job step. The fields on this report correspond to the fields in the DB2 trace header record.

**Figure 12** DB2 Trace Header Information report

<table>
<thead>
<tr>
<th>DB2 TRACE HEADER INFORMATION</th>
</tr>
</thead>
</table>

| JOB NAME FROM JOB CARD       | ABCMRHA3   |
| JOB STEP NAME                | DBRUN      |
| PROCEDURE NAME               | ARCSTEP    |
| DB2 IDENTIFIER               | DBAB       |
| NETWORK ORIGINAL NODE NAME   | SYSO       |
| SUBMITTING USER IDENTIFIER   | ABC        |
| BASE PRODUCT LEVEL           | V2.0.00    |
| DATE DISTRIBUTION LEVEL      | 3/17/1997  |
| TIME DISTRIBUTION LEVEL      | 14.42.00   |
| ORIGINAL JOB ENTRY SUBSYSTEM ASSIGNED JOB IDENTIFIER | JOB08414 |
| EXECUTION JOB ENTRY SUBSYSTEM ASSIGNED JOB IDENTIFIER | JOB08414 |
| NETWORK EXECUTION NODE NAME  | SYSO       |
| EXECUTION START DATE          | 97.024     |
| EXECUTION START TIME          | 18.34.17   |
| LAST CHECKPOINT IDENTIFIER   | **COLD**   |
| LAST CHECKPOINT DATE          |            |
| LAST CHECKPOINT TIME          |            |
| TRACE START DATE              | 97.024     |
| TRACE START TIME              | 18.34.45   |
| TRACE REQUESTOR               | $ARCDIR    |
| TRACE LEVEL                   | 001        |
An IMS Trace Detail Information report provides information for each selected IMS or AR/CTL call and its results. The report contains some variable information, depending on whether the call used an I/O PCB (Figure 13); an ASAM, generalized sequential access method (GSAM), or database program communication block (DB PCB) (Figure 14 on page 164); or an application interface block (AIB) (Figure 15 on page 164). The fields on this report correspond to the fields in the IMS trace detail record.

**Figure 13  IMS Trace Detail Information report (I/O PCB variation) (part 1 of 2)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCB RELATIVE NUMBER</td>
<td>00000</td>
</tr>
<tr>
<td>PCB TYPE</td>
<td>IOPCB</td>
</tr>
<tr>
<td>IMS CALL</td>
<td>XRST</td>
</tr>
<tr>
<td>IMS CALL ELAPSED TIME</td>
<td>00:00:12.45</td>
</tr>
<tr>
<td>PCB DATA BASE NAME</td>
<td>IOPCB</td>
</tr>
<tr>
<td>PCB SEGMENT LEVEL</td>
<td></td>
</tr>
<tr>
<td>PCB STATUS CODE</td>
<td></td>
</tr>
<tr>
<td>PCB PROCESSING OPTIONS</td>
<td></td>
</tr>
<tr>
<td>PCB SEGMENT FEEDBACK</td>
<td></td>
</tr>
<tr>
<td>PCB KEYLENGTH</td>
<td>0000000000</td>
</tr>
<tr>
<td>PCB NUMBER OF SENSITIVE SEGMENTS</td>
<td>0000000000</td>
</tr>
<tr>
<td>PCB KEY FEEDBACK AREA</td>
<td>444444444444</td>
</tr>
<tr>
<td>INPUT/OUTPUT AREA LENGTH</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>INPUT/OUTPUT AREA</td>
<td>444444444444</td>
</tr>
<tr>
<td>FIRST USER AREA LENGTH</td>
<td>&lt; 0004000C</td>
</tr>
<tr>
<td>FIRST USER AREA</td>
<td>000000 W-DATES FFFFFFF0E6CECE 000000006041352</td>
</tr>
<tr>
<td>LIT</td>
<td>DCE4444444440000 3930000000000000</td>
</tr>
<tr>
<td>SECOND USER AREA LENGTH</td>
<td>H 000C</td>
</tr>
</tbody>
</table>

**Figure 13** IMS Trace Detail Information Report (I/O PCB variation) (part 2 of 2)

| SECOND USER AREA | 0048 CKP 444444444444CDD 0000000000000327 |

**Figure 14** IMS Trace Detail Information report (non-I/O PCB variation)

| PCB RELATIVE NUMBER | 00007 |
| PCB TYPE | GSAM PCB |
| IMS CALL | ISRT |
| IMS CALL ELAPSED TIME | 00:00:02.03 |
| PCB DATA BASE NAME | ASAM01 |
| PCB SEGMENT LEVEL | |
| PCB STATUS CODE | |
| PCB PROCESSING OPTIONS | L |
| PCB SEGMENT FEEDBACK | PCBASAM1 |
| PCB KEYLENGTH | 00000012 |
| PCB NUMBER OF SENSITIVE SEGMENTS | 00065535 |
| PCB KEY FEEDBACK AREA | 4444444440000 0000000000000 |
| INPUT/OUTPUT AREA | |

**Figure 15** IMS Trace Detail Information report (AIB variation)

<p>| PCB RELATIVE NUMBER | 00000 |
| PCB TYPE | AIB BLOCK |
| IMS CALL | XRST |
| IMS CALL ELAPSED TIME | 00:00:00.06 |
| APPLICATION INTERFACE BLOCK ID | DFSAIB |
| DFSAIB ALLOCATED LENGTH | |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBFUNCTION CODE</td>
<td>0008 0000</td>
</tr>
<tr>
<td>RESOURCE NAME 1</td>
<td>IOPCB</td>
</tr>
<tr>
<td>OUTPUT AREA LENGTH (MAX)</td>
<td>(data omitted)</td>
</tr>
<tr>
<td>OUTPUT AREA LENGTH USED</td>
<td>(data omitted)</td>
</tr>
<tr>
<td>RETURN CODE</td>
<td>(data omitted)</td>
</tr>
<tr>
<td>REASON CODE</td>
<td>(data omitted)</td>
</tr>
<tr>
<td>RESOURCE ADDRESS 1</td>
<td>(&amp;.</td>
</tr>
<tr>
<td>RESOURCE ADDRESS 2</td>
<td>(data omitted)</td>
</tr>
<tr>
<td>RESOURCE ADDRESS 3</td>
<td>(data omitted)</td>
</tr>
<tr>
<td>INPUT/OUTPUT AREA LENGTH</td>
<td>(data omitted)</td>
</tr>
<tr>
<td>INPUT/OUTPUT AREA</td>
<td>CKP 4444444444444444CDD</td>
</tr>
<tr>
<td></td>
<td>0000000000000000327</td>
</tr>
<tr>
<td>FIRST USER AREA LENGTH</td>
<td>4</td>
</tr>
<tr>
<td>FIRST USER AREA</td>
<td>(data omitted)</td>
</tr>
<tr>
<td>SECOND USER AREA LENGTH</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>(data omitted)</td>
</tr>
</tbody>
</table>
The DB2 Trace Detail Information report (Figure 16) provides information for each selected DB2 call. The fields on this report correspond to the fields in the DB2 trace detail record. The SQL statement types are defined in member DSNXRDI of the DSNMACS library distributed with DB2.

Most fields on this report are self-explanatory, with the following exceptions:

**CODE POINTER**
Pointer to the SQL communication area (SQLCA).

**VPARAM POINTER**
Pointer to the SQL statement variable parameter list.
AUXILIAR PARAM POINTER
Pointer to the SQL data area structure (SQLDA).

DB2 Trace Trailer Information report

The DB2 Trace Trailer Information report (Figure 17) provides information gathered at the end of the application program job step. The fields on this report correspond to the fields in the DB2 trace trailer record.

Figure 17  DB2 Trace Trailer Information report

<table>
<thead>
<tr>
<th>DB2 TRACE TRAILER INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACE END DATE ........ 97.024</td>
</tr>
<tr>
<td>TRACE END TIME ........ 18.35.26</td>
</tr>
<tr>
<td>TRACE REQUESTOR .......... $ARCDIR</td>
</tr>
<tr>
<td>LAST CHECKPOINT IDENTIFIER . 00000017</td>
</tr>
<tr>
<td>LAST CHECKPOINT DATE .... 97.024</td>
</tr>
<tr>
<td>LAST CHECKPOINT TIME .... 18.35.00</td>
</tr>
</tbody>
</table>

IMS Trace Trailer Information report

The IMS Trace Trailer Information report (Figure 18) provides information gathered at the end of the application program job step. The fields on this report correspond to the fields in the IMS trace trailer record.

Figure 18  IMS Trace Trailer Information report

<table>
<thead>
<tr>
<th>IMS TRACE TRAILER INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACE END DATE ........ 97.024</td>
</tr>
<tr>
<td>TRACE END TIME ........ 18.35.26</td>
</tr>
<tr>
<td>TRACE REQUESTOR .......... $ARCDIR</td>
</tr>
<tr>
<td>LAST CHECKPOINT IDENTIFIER . 00000017</td>
</tr>
<tr>
<td>LAST CHECKPOINT DATE .... 97.024</td>
</tr>
<tr>
<td>LAST CHECKPOINT TIME .... 18.35.00</td>
</tr>
</tbody>
</table>
Trace Summary report

At the end of the trace format, the Trace Format utility produces the Trace Summary report (Figure 19). It summarizes information about the job step that was traced and about the records processed during formatting.

Figure 19  Trace Summary report

| TRACE SUMMARY REPORT |
+----------------------+

JOB NAME FROM JOB CARD .................. ABCMRHA3
JOB STEP NAME .......................... DBRUN
PROCEDURE NAME ......................... ARSTEP

EXECUTION JOB ENTRY SUBSYSTEM ASSIGNED JOB IDENTIFIER .... JOB08414
EXECUTION START DATE .................... 97.024
EXECUTION START TIME ................... 18.34.17

TRACE START DATE ....................... 97.024
TRACE START TIME ....................... 18.34.45

NUMBER OF IMS HEADER RECORDS ............ 1
NUMBER OF IMS DETAIL RECORDS ............ 91
NUMBER OF IMS TRAILER RECORDS ........... 1

NUMBER OF DB2 HEADER RECORDS ............ 1
NUMBER OF DB2 DETAIL RECORDS ............ 57
NUMBER OF DB2 TRAILER RECORDS ........... 1

TRACE END DATE .......................... 97.024
TRACE END TIME .......................... 18.35.26
TRACE REQUESTOR ....................... $ARCDIR

Dump Format reports

Figure 20 shows an example of one of the trace reports (in dump format) that the Trace Format utility produces when you specify FORMAT=DUMP. This format is similar to those of typical operating system dumps. A header line indicates the beginning of each selected record. The left-most column indicates the offset of the first byte of the next column. The middle columns contain the data; the number of columns depends on the width selected with the WIDTH keyword. The right-most column contains a character representation of the data in the line.

Figure 20  Dump Format report

<table>
<thead>
<tr>
<th>IMS HEADER RECORD</th>
<th>IMS TRAILER RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>00A40000 01000000 0001E5F2 4BF04BF0 F0F0F261 * U V2.0.0002/ *</td>
<td>0014 F2F961F1 F9F9F7F1 F44BF4F2 4BF0F000 94024F18 *29/199714.42.00 M</td>
</tr>
<tr>
<td>0028 3445685B C1D9C3C4 C9D94001 E2E8E2D6 40404040 *</td>
<td>$ARCDIR SYS0</td>
</tr>
</tbody>
</table>
### Dump Format reports

#### Chapter 5 Using AR/CTL operational services

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>003C</td>
<td>D1D6C2F0</td>
</tr>
<tr>
<td>0050</td>
<td>40404040</td>
</tr>
<tr>
<td>0064</td>
<td>C1C2C3D4</td>
</tr>
<tr>
<td>0078</td>
<td>E3C507D4</td>
</tr>
<tr>
<td>008C</td>
<td>0094024F</td>
</tr>
<tr>
<td>00A0</td>
<td>00000000</td>
</tr>
</tbody>
</table>

**IMM DETAIL RECORD**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>005F</td>
<td>40404040</td>
</tr>
<tr>
<td>0028</td>
<td>000003C9</td>
</tr>
<tr>
<td>003C</td>
<td>600988D7</td>
</tr>
<tr>
<td>0050</td>
<td>40404040</td>
</tr>
</tbody>
</table>

**IMM DETAIL RECORD**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>005F</td>
<td>40404040</td>
</tr>
<tr>
<td>0028</td>
<td>000003C9</td>
</tr>
<tr>
<td>003C</td>
<td>600A98D7</td>
</tr>
<tr>
<td>0050</td>
<td>40404040</td>
</tr>
</tbody>
</table>

**DB2 HEADERS RECORD**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>009B</td>
<td>40404040</td>
</tr>
<tr>
<td>0050</td>
<td>40404040</td>
</tr>
</tbody>
</table>

**DB2 DETAIL RECORD**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0048</td>
<td>00480000</td>
</tr>
<tr>
<td>0028</td>
<td>D305E035</td>
</tr>
<tr>
<td>003C</td>
<td>000AF988</td>
</tr>
</tbody>
</table>

**DB2 DETAIL RECORD**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0048</td>
<td>00480000</td>
</tr>
<tr>
<td>0028</td>
<td>D305E035</td>
</tr>
<tr>
<td>003C</td>
<td>000AF988</td>
</tr>
</tbody>
</table>
Changing COBOL application programs

This chapter tells how to change COBOL application programs to use the explicit calls provided by the APPLICATION RESTART CONTROL (AR/CTL) products. This chapter contains the following information:

Overview .......................................................... 172
Supported versions of COBOL .................................. 172
Types of AR/CTL calls ........................................... 173
Sample COBOL programs ...................................... 173
EXEC parameters .................................................. 173
Compile and link issues .......................................... 174
Considerations for using Language Environment .......... 175
Considerations for using AR/CTL with COBOL programs ...
Using AR/CTL-format checkpoint/restart calls ............. 177
  Using the restart call .......................................... 177
  Coding a restart ID area ...................................... 177
  Using the AR/CTL checkpoint call ......................... 179
  Coding the checkpoint ARB .................................. 181
Managing virtual storage for subprograms .................. 182
  Overview of the ARCSPVS API ............................ 182
  Implementing the ARCSPVS API ........................... 184
  Identifying virtual storage areas to be managed .......... 185
  Creating the $ARCSPVS parameter block ................. 186
  Using the Define function .................................. 188
  Using the Refresh function ................................ 189
  Using the Delete function .................................. 190
Using AR/CTL common calls .................................... 191
  Using the AR/CTL common call ............................ 191
  Coding an I/O area .......................................... 193
  Example program using the AR/CTL common call ......... 194
Using IMS-compatible checkpoint/restart calls ............ 195
  Entry and return ............................................ 195
  Call function codes .......................................... 196
  Using the XRST call ......................................... 197
Overview

The easiest way to obtain AR/CTL services in an application program is with automated methods, which require no program changes. If you are not able to use automated methods, you can change the application program to issue explicit requests for AR/CTL services. This section provides general information about making changes to application programs written in COBOL.

Before using the information in this chapter, review the overview information about AR/CTL checkpoint/restart and data services in Chapter 3, "Using AR/CTL checkpoint/restart services."

Supported versions of COBOL

AR/CTL fully supports application programs written in COBOL/VS, COBOL II, and COBOL/370.

AR/CTL provides toleration support for application programs that use the COBOL for MVS and VM or COBOL for OS/390 and VM compiler. If a program that uses this compiler exploits features that are not available with COBOL/370 or earlier, the program should not use the Automatic Restart and Automatic Checkpoints processing options; results are unpredictable.
Types of AR/CTL calls

AR/CTL supports several formats for the calls that request AR/CTL checkpoint/restart services:

- AR/CTL format
- AR/CTL common format
- IMS-compatible

AR/CTL supports several formats for the calls that request application sequential access method (ASAM) services:

- IMS generalized sequential access method (GSAM) replacement
- ASAM callable interface

Sample COBOL programs

BMC Software provides executable samples of application programs that use AR/CTL calls. For a list of these samples, see Appendix B, “Locating samples.”

EXEC parameters

The following sections provide information that is related to EXEC statement parameters.

Length of parameter list

Normally, MVS allows a 100-byte PARM string for an EXEC PGM statement. However, AR/CTL uses some of these bytes during processing. Therefore, 61 bytes is the maximum allowable length of the PARM string on the EXEC statement for an application program that uses AR/CTL.

Original operating system parameters

According to standard MVS linkage conventions, register 1 points to the original operating system parameter list. However, AR/CTL uses this register for other purposes and provides access to a copy of the original parameter list through MVS name token services.
Member ASMORIGP of the sample library contains model code (written in Assembler language) that you can adapt and include in an application program if the program needs access to a copy of the original parameter list. This sample shows how to fetch the operating system parameter list. The address of the list is located at offset +8 (byte 9) of the TOKEN field.

This sample also shows a quick way for the application program to detect whether AR/CTL is active. The sample sets the return code to zero if AR/CTL is active and to a nonzero value if AR/CTL is not active or a problem is detected.

**Compile and link issues**

An application program can use any explicit AR/CTL call (AR/CTL-format checkpoint/restart calls, AR/CTL common calls, ASAM callable interface calls) with dynamic linking or with static linking (such as specified for COBOL application programs with the NODYNAM compiler option). Considerations for other compiler options may also apply to the use of AR/CTL with COBOL.

**Dynamic linking**

If the program uses dynamic linking, the AR/CTL libraries are automatically included at execution time. No change is required for your current link-edit process.

**IMS environment**

If an existing program uses IMS, the program does not need to be recompiled or relinked unless you made program changes.

**Static linking**

If a non-DB2 program uses static linking, you must include the AR/CTL execution library in the SYSLIB concatenation of the link-edit JCL. The order of the libraries in the concatenation is not important.

If a DB2 application program uses static linking, it must be link-edited with the AR/CTL language interface module (ARCLI000) to replace the DB2 language interface module (DSNALI, DSNELI, or DSNHLI). Member #DB2LNK in the AR/CTL sample library contains sample JCL for link-editing the application program with AR/CTL. Insert the AR/CTL execution load module library ahead of the DB2 load library (DSNLOAD). The following order of libraries is correct:

1. AR/CTL load library
2. compiler (such as COBOL) run time library
3. IMS RESLIB library (if applicable)

4. DB2 DSNLOAD library (if applicable)

5. user load library (containing the external routines that are specific to your environment)

**Other compiler options**

AR/CTL does not support the FASTSRT compiler option for internal sorts.

AR/CTL does not support the RTEREUS(ON) option in COBOL for MVS and VM.

**Linking of IMS-compatible programs**

In an IMS-compatible program, the program uses IMS calls and structures but IMS is not active. If a program works with IMS and you want to execute the program in IMS-compatible mode, it does not need to be recompiled or relinked. If you no longer have access to the IMS libraries and you change an existing IMS-compatible program or create a new program, you must change the link-edit SYSLIN step. Include the AR/CTL ARCLI000 module from the AR/CTL load library instead of the IMS DFSLI000 module from the IMS RESLIB library:

```plaintext
INCLUDE ARCLIB(ARCLI000)
ENTRY DLITCBL
...
//ARCLIB DD DSN=BMCARC.LOAD
```

**Considerations for using Language Environment**

The following Language Environment (LE) parameters are for non-IMS programs only and are recommended for non-CICS COBOL programs for restartable applications:

```plaintext
//GO EXEC PGM=applpgm,
// PARM=('RPTOPTS(ON),ABTERMENC(ABEND)',
// 'TERMTHDACT(UADUMP)')
```

Use the RPTOPTS(ON) parameter to obtain a report that shows the run-time options in effect. The report is written to the SYSOUT data set by default. (The LE run-time options report is not written if the job step terminates abnormally).
Other ways to specify LE run-time options for either IMS or non-IMS programs include the following parameters:

- You can specify LE run-time parameters system wide in the CEEDOPT module.
- You can also assemble a CEEUOPT module, which is linked into the application.

See the appropriate IBM LE documentation for complete details on all LE run-time options.

The following LE run-time parameters are recommended for the CEEDOPT macro or the CEEUOPT macro:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABTERMENC(ABEND)</td>
<td>RETCODE</td>
</tr>
<tr>
<td>TERMTHDACT(UADUMP)</td>
<td>TRACE</td>
</tr>
<tr>
<td>RPTOPTS(ON)</td>
<td>Report to SYSOUT</td>
</tr>
</tbody>
</table>

Considerations for using AR/CTL with COBOL programs

Keep the following considerations in mind when you use AR/CTL with COBOL programs:

- For COBOL application programs running under AR/CTL, turn off the abend interception functions of COBOL.

- AR/CTL does not checkpoint or reposition files from which COBOL ACCEPT is performed, even if it is a permanent file in DASD.

- AR/CTL does not support COBOL external data if the Automatic Restart option is set to Y. COBOL can place external data in a different working storage location at restart time than its location for the normal start. To avoid restart problems, use a single 01 structure to define the external data to save in a checkpoint user save area.

- For data sets that use AR/CTL sequential interception and that are opened with the COBOL EXTEND option, AR/CTL cannot provide repositioning support if the data set is copied or reblocked, or if the record length or record format is changed, between abend and restart. AR/CTL cannot provide repositioning support for this type of data set if AR/CTL was not present when the data set was originally created with EXTEND.

- For proper AR/CTL termination processing, use the COBOL GOBACK statement instead of the STOP RUN statement.
When the application program uses the Automatic Restart options, the start of the working storage area that AR/CTL saves during checkpoint processing (and can restore at restart) is the beginning address of working storage. AR/CTL determines the end of the working storage area to save and restore:

1. It searches for the following 30-byte AR/CTL constant that you may code in the program:

   ```cobol
   01 FILLER PIC X(30) VALUE ' ** CHKP AREA END FOR AR/CTL **'.
   ```

2. If the AR/CTL constant is not present in the program and the program uses DB2, AR/CTL searches for the beginning of DB2-generated areas, as indicated by the INCLUDE SQLCA statement, and stops restoring storage at this point (DB2-generated areas are not restored, and no other storage after the statement is restored). The AR/CTL constant should not follow the INCLUDE SQLCA statement.

3. If the AR/CTL constant is not present and if the program does not use DB2, the end of working storage is as constructed by COBOL.

### Using AR/CTL-format checkpoint/restart calls

This section describes how to use AR/CTL-format calls to obtain checkpoint/restart services. These calls are valid in non-IMS environments only. Table 6 summarizes the AR/CTL-format calls available for an application program to request checkpoint/restart services.

<table>
<thead>
<tr>
<th>Call type</th>
<th>Call name</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>restart</td>
<td>ARCXRST</td>
<td>set up the extended checkpoint/restart environment and invoke AR/CTL processing</td>
</tr>
<tr>
<td>checkpoint</td>
<td>ARCECHK or ARCCHKP</td>
<td>initiate checkpoint processing</td>
</tr>
</tbody>
</table>

### Using the restart call

To initiate the checkpoint/restart environment and restart processing, an application program can issue a restart call to AR/CTL.
Using the restart call

**WARNING**
If an application program uses checkpoint/restart support or any AR/CTL data services, it must issue the restart call once as the first call and before any data set open processing.

The restart call should specify user areas identical to those specified in the checkpoint calls. If the areas are different, the areas in the checkpoint calls will be used.

**NOTE**
For ease of maintenance, BMC Software recommends that the program specify user areas on the restart call and omit them from the checkpoint calls.

```
CALL 'ARCXRST' USING restart-id, beginarea-1, endarea-1, beginarea-2, endarea-2,...beginarea-n, endarea-n
```

**ARCXRST**
Required. The literal that identifies the call as the AR/CTL restart call.

**restart-id**
Required. The name of the 12-byte field that receives the restart identifier from AR/CTL. For more information, see “Coding a restart ID area” on page 179.

**NOTE**
It is not necessary for any of the following user areas to begin or end on a fullword boundary.

**beginarea-1**
Required. The name of a field that contains the beginning address of the first program data area to save in the checkpoint record or to restore at restart.

**endarea-1**
Required. The name of a field that marks the ending address of the first program data area to save in the checkpoint record or to restore at restart. This field is not included in the save area. Logically, this field is the ending address plus one byte.

**beginarea-n**
Optional. The name of a field that contains the beginning address of the next program data area to save in the checkpoint record or to restore at restart. You can specify a maximum of seven areas.

**endarea-n**
Optional. The name of a field that marks the ending address of the next program data area to save in the checkpoint record or to restore at restart. This field is not included in the save area. Logically, this field is the ending address plus one byte.
Coding a restart ID area

In the working storage of the application program, identify a 12-byte I/O area to contain the restart ID returned from the restart call. Initialize the field to blanks before issuing the restart call. After AR/CTL processes the restart call, this field contains blanks (X’4040’) if an application program is started normally. The field contains the restart checkpoint identifier used for restart if the application program is restarted.

The following example shows the restart ID area as coded in COBOL:

```
IDENTIFICATION DIVISION.
... 
DATA DIVISION.
WORKING-STORAGE SECTION.
 01 XRSTID-AREA.
    02 XRSTID PICTURE X(12).
```

Using the AR/CTL checkpoint call

To initiate checkpoint processing, an application program can issue a checkpoint call to AR/CTL. A checkpoint call is appropriate to issue at the following points during processing:

- immediately after the restart call
- at the end of any logical unit of work (UOW)
- immediately before termination

AR/CTL automatically generates and increments the checkpoint ID when the application program issues the AR/CTL-format checkpoint request.

The application program can issue a checkpoint call with or without defining working storage areas to save. If the checkpoint call omits working storage areas, AR/CTL saves the working storage areas defined in the restart (ARCXRST) call.

**NOTE**

For ease of maintenance, BMC Software recommends that the program specify user areas on the restart call and omit them from the checkpoint calls.

The formats of the ARCECHK checkpoint call in COBOL follow:

```
CALL 'ARCECHK' USING arc-arb
CALL 'ARCECHK' USING arc-arb, beginarea-1, endarea-1,
    beginarea-2, endarea-2,...,beginarea-n, endarea-n
```
The formats of the **ARCCHKP** call in COBOL follow:

```cobol
CALL 'ARCCHKP'
CALL 'ARCCHKP' USING beginarea-1, endarea-1,
    beginarea-2, endarea-2,...beginarea-n, endarea-n
```

**ARCECHK** or **ARCCHKP**
Required. The literal that identifies the call as the AR/CTL checkpoint call.

**arc-arb**
Required on the **ARCECHK** call; not valid on the **ARCCHKP** call. The name of a fullword-aligned field that contains the address of the application request block (ARB). The ARB is defined in the working storage of the application program. For more information, see “Coding the checkpoint ARB” on page 181.

Before issuing the **ARCECHK** call, the application program can set the field in the ARB that indicates the function to use (**FORCE**), as shown in the following COBOL example:

```cobol
MOVE 'FORCE' TO ARB-FUNC.
```

**NOTE**
It is not necessary for any of the following user areas to begin or end on a fullword boundary; however, the addresses of the user areas must be aligned on a fullword. Usually, this alignment is performed automatically in high-level languages.

**beginarea-1**
Optional; if omitted, AR/CTL saves the first area identified with the restart call. The name of a field that contains the beginning address of the first application program data area to save in the checkpoint record.

**endarea-1**
Optional; if omitted, AR/CTL saves the first area identified with the restart call. The name of a field that marks the ending address of the first application program data area to save in the checkpoint record. This field is not included in the save area.

**beginarea-n**
Optional. The name of a field that contains the beginning address of the next application program data area to save in the checkpoint record. You can specify a maximum of seven areas.

**endarea-n**
Optional. The name of a field that marks the ending address of the next application program data area to save in the checkpoint record. This field is not included in the save area.
Coding the checkpoint ARB

The application program and AR/CTL use the ARB to exchange information such as call functions and results. Code the ARB in the working storage area of the application program. The program can use a single ARB for all calls, or it can use a separate ARB for checkpoint support and for each sequential file that uses ASAM support. If the program uses a single ARB for multiple call types, it must reinitialize the ARB for each call.

The following example shows the ARB in COBOL:

```cobol
IDENTIFICATION DIVISION.
... DATA DIVISION.
WORKING-STORAGE SECTION.
01 ARCARB.
   02 ARCARB-TYPE PIC X(8) VALUE '$ARCARB '.
   02 FUNC-CODE PIC X(8) VALUE ' '.
   02 FILLER PIC X(8) VALUE SPACES.
   02 STATUS-CODE PIC X(2) VALUE SPACES.
   02 FILLER PIC X(2) VALUE SPACES.
   02 FILLER PIC X(4) VALUE SPACES.
```

**ARB Type**
This field serves to identify the ARB.

**Function Code**
This field (at offset 8 from the beginning byte 0) indicates the type of access you want for the next call. If the call is a checkpoint call, you can leave the function code field blank or set it to `FORCE`. A function code of `FORCE` overrides the checkpoint pacing decision and forces checkpoint processing to occur for the checkpoint identified in the call. You can use the `FORCE` function to ensure that a restart point is established after a long-running procedure such as a sort.

---

**NOTE**
AR/CTL saves data from the beginning address up to, but not including, the ending address. The user areas do not need to begin or end on a fullword boundary.
Managing virtual storage for subprograms

AR/CTL provides the Subprogram Virtual Storage (ARCSPVS) application program interface (API) for managing defined areas of virtual storage for any application program, including a subprogram, in a restartable environment.

If an application subprogram already uses the subprogram support that is provided with the QUICKSTART for MVS product, no program changes are required for the use of AR/CTL support for subprograms. For more information, see “QUICKSTART-to-AR/CTL bridge” on page 32.

Overview of the ARCSPVS API

You can use the ARCSPVS API to define the virtual storage areas that you want AR/CTL to save during checkpoint processing and to restore automatically during initialization processing or manually on demand. You can define these areas in addition to the user areas that you can define with checkpoint (CHKP) and restart (XRST) calls. You can think of the API as a technique that complements the standard checkpoint/restart technique for managing user areas.

The ARCSPVS API works in all AR/CTL restartable environments and supports both IMS and non-IMS programs.

Virtual storage areas

A virtual storage area is defined by the beginning address of the area and the delimiter of the area, which is the address of the end of the area and is not included in the area itself. Subtracting the beginning address from the delimiter provides the length of the area. Defined virtual storage areas may not overlap.

---

**NOTE**

If a checkpoint pacing exit routine returns a code that tells AR/CTL to bypass the checkpoint, AR/CTL bypasses checkpoint processing even if FORCE is specified.

**Status Code**

This field contains the status code that the application program can check to determine the results of a call to AR/CTL. The status codes that AR/CTL can return are documented with the calls.
The definition of the virtual storage area persists until you explicitly delete the area (by using the Delete function of the ARCSPVS API) or the job step is cold-started. The name of each virtual storage area must be unique within a job step while the area remains defined.

**Checkpoint data set**

AR/CTL stores the defined virtual storage areas in the checkpoint data set with other checkpoint data. The ARCSPVS API supports an unlimited number of virtual storage areas in the application program. The amount of available space in the checkpoint data set is the only limitation on the size and number of the virtual storage areas that you can define.

Before you implement the ARCSPVS API for an application program, verify that space allocation parameters for the checkpoint data set are large enough to provide sufficient space for the virtual storage areas to be managed. For more information, see “Setting up dynamic allocation options” on page 82.

**ARCSPVS API structures and functions**

The ARCSPVS API is simple to implement in the application program. The API consists of the following elements:

- a single entry point, ARCSPVS, to call from the application program
- a single parameter block, $ARCSPVS, to pass with the call to ARCSPVS
  - The parameter block is common to all calls, regardless of the function.
- an additional entry point, ARCSPAD, to call if your compiler does not support the SET field TO ADDRESS OF field statement

The ARCSPVS API supports the following functions:

- The Define function assigns a name to a virtual storage area and defines the characteristics of the area. You can also use this function at program restart to refresh the virtual storage area automatically to its state at the last time it was saved.

- The Refresh function manually refreshes the virtual storage area to its state at the last time it was saved. This function is valid only if you specify MANUAL for the Refresh Interval parameter when you use the Define function.

- The Delete function removes the definition of a virtual storage area when the application program no longer requires AR/CTL to manage the area.
Program changes before restart

If you need to make program changes after an abend and before a restart, keep the following considerations in mind:

- At restart, AR/CTL compares the length of each virtual storage area in the checkpoint data set with the length of the area that is currently defined in the program. If the lengths are different, AR/CTL uses the smaller of the two lengths.

- The name of each virtual storage area must be the same at restart as it was during the initial Define function.

Implementing the ARCSPVS API

To implement the ARCSPVS API in an application program (including main programs and subprograms), perform the following steps. For a sample COBOL program that shows how to use the ARCSPVS API, see member ARCSVSCP of the AR/CTL sample library.

1 In the working storage section of the application program, identify each virtual storage area to be managed.

You must label the beginning and the delimiter of the area. For more information, see “Identifying virtual storage areas to be managed” on page 185.

2 In the working storage section of the application program, create the $ARCSPVS parameter block.

For more information, see “Creating the $ARCSPVS parameter block” on page 186.

3 For each virtual storage area to be managed, use the Define function.

This step consists of initializing the $ARCSPVS parameter block and specifying DEFINE for the Function parameter (along with the name of the virtual storage area to be defined), issuing a call to ARCSPVS, and passing the initialized parameter block in the call.

If your compiler does not support the set field to address of field statement, you can issue a call to ARCSPAD to obtain the addresses of the beginning and the delimiter of the virtual storage area.

For more information, see “Using the Define function” on page 188.

4 If you have specified MANUAL for the Refresh Interval parameter, use the Refresh function to restore the virtual storage area manually.
This step consists of specifying REFRESH for the Function parameter in the $ARCSPVS parameter block (along with the name of the virtual storage area to be refreshed), issuing a call to ARCSPVS, and passing the initialized parameter block in the call. For more information, see “Using the Refresh function” on page 189.

5 At termination of the subprogram, before the subprogram returns to the main program, you can use the Delete function to remove the virtual storage area, or simply issue the call when the area is no longer needed.

This step consists of specifying DELETE for the Function parameter in the $ARCSPVS parameter block (along with the name of the virtual storage area to be deleted), issuing a call to ARCSPVS, and passing the initialized parameter block in the call. For more information, see “Using the Delete function” on page 190.

Identifying virtual storage areas to be managed

In the working storage section of the application program, identify each virtual storage area that you want AR/CTL to manage as shown in Figure 21.

Figure 21 Virtual storage areas in an application program

```
WORKING-STORAGE SECTION.
...
  01 VIRTUAL-STORAGE-0.
  05 VSO-BEGIN       PICTURE IS X(32)
          VALUE IS
          '< SUBPROGRAM VIRTUAL STORAGE 0 >'.
  . additional saved working storage areas here
  .
  05 VSO-DELIMITER   PICTURE IS X
          VALUE IS HIGH-VALUES.
...
```

A virtual storage area is defined by the beginning address of the area and the delimiter of the area, which is the address of the end of the area and is not included in the area itself. Subtracting the beginning address from the delimiter provides the length of the area. Defined virtual storage areas may not overlap.

The definition of the virtual storage area persists until you explicitly delete the area (by using the Delete function of the ARCSPVS API) or the job step is cold-started. You cannot modify the characteristics of a virtual storage area after you define the area. The name of each virtual storage area must be unique within a job step while the area remains defined.
Creating the $ARCSPVS parameter block

At the end of the working storage section of the application program, create the $ARCSPVS parameter block as shown in Figure 22. For a sample COBOL copybook that you can copy, see member ARCSVSCB of the AR/CTL sample library.

Figure 22  $ARCSPVS parameter block

```
*  AR/CTL SUBPROGRAM VIRTUAL STORAGE PROCESSOR PARAMETER BLOCK.
01  ARCSPVS-PARAMETER-BLOCK.
   *  PARAMETER BLOCK IDENTIFIER.  VALID VALUE IS $ARCSPVS.
   03  SPVS-IDENTIFIER         PICTURE IS X(8)
       VALUE IS '$ARCSPVS'.
   *
   *  FUNCTION CODE.  VALID VALUES ARE DEFINE, REFRESH, AND DELETE.
   03  SPVS-FUNCTION           PICTURE IS X(16)
       VALUE IS ALL SPACES.
   *
   *  APPLICATION SUBPROGRAM VIRTUAL STORAGE NAME.  MUST BE NOT BLANK AND NOT LOW-VALUES.
   03  SPVS-NAME               PICTURE IS X(32)
       VALUE IS ALL SPACES.
   *
   *  RETURN CODE.
   03  SPVS-RETURN             PICTURE IS S9(8)
       USAGE IS COMPUTATIONAL
       VALUE IS ALL ZEROES.
       88  SPVS-RETURN-SUCCESSFUL VALUE IS ALL ZEROES.
       88  SPVS-RETURN-REFRESHED VALUE IS 4.
   *
   *  BEGIN ADDRESS OF APPLICATION SUBPROGRAM VIRTUAL STORAGE AREA.
   03  SPVS-BEGIN              USAGE IS POINTER
       VALUE IS ALL NULLS.
   *
   *  ADDRESS OF APPLICATION SUBPROGRAM VIRTUAL STORAGE AREA DELIMITER.
   03  SPVS-DELIMITER          USAGE IS POINTER
       VALUE IS ALL NULLS.
   *
   *  APPLICATION SUBPROGRAM VIRTUAL STORAGE AREA REFRESH INTERVAL.  VALID VALUES ARE DEFINE AND MANUAL.
   03  SPVS-REFRESH-INTERVAL   PICTURE IS X(16)
       VALUE IS ALL SPACES.
   *
   *  APPLICATION SUBPROGRAM VIRTUAL STORAGE AREA SAVE INTERVAL.  VALID VALUE IS UOW.
   03  SPVS-SAVE-INTERVAL      PICTURE IS X(16)
       VALUE IS ALL SPACES.
   *
   *  RESERVED AREA.  MUST BE LOW-VALUES.
   03  SPVS-RESERVED           PICTURE IS X(412)
       VALUE IS ALL LOW-VALUES.
```
Table 7 describes the required fields in the $ARCSPVS parameter block.

**Table 7  $ARCSPVS parameter block fields**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Contents</th>
<th>Format and length</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>the identifier for the parameter block</td>
<td>X(8)</td>
<td>$ARCSPVS</td>
</tr>
<tr>
<td></td>
<td>The ARCSPVS API requires the value to be $ARCSPVS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>the function to be performed in response to the call to the ARCSPVS API</td>
<td>X(16)</td>
<td>DEFINE REFRESH DELETE</td>
</tr>
<tr>
<td>Name</td>
<td>the name of the virtual storage area to be managed</td>
<td>X(32)</td>
<td>When the parameter block is passed to the ARCSPVS API, the value cannot be blank or low values.</td>
</tr>
<tr>
<td>Return Code</td>
<td>the return code that the ARCSPVS API uses to communicate the results of the function</td>
<td>S9(8)</td>
<td>ALL ZEROES 4</td>
</tr>
<tr>
<td></td>
<td>If the function is successful, the API sets the value to 0 before returning control to the application program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If the virtual storage area is refreshed, the API sets the value to 4 before returning control to the application program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Begin</td>
<td>the address of the beginning of the virtual storage area to be managed</td>
<td>pointer</td>
<td>beginning address</td>
</tr>
<tr>
<td>Delimiter</td>
<td>the address of the delimiter of the virtual storage area to be managed</td>
<td>pointer</td>
<td>delimiter address</td>
</tr>
<tr>
<td></td>
<td>(this delimiter is not part of the virtual storage area)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refresh Interval</td>
<td>the option that tells AR/CTL when to restore the virtual storage area to its state at the last time that the area was saved</td>
<td>X(16)</td>
<td>DEFINE MANUAL</td>
</tr>
<tr>
<td>Save Interval (UOW)</td>
<td>the parameter that tells AR/CTL when to save the contents of the virtual storage area to the checkpoint data set</td>
<td>X(16)</td>
<td>UOW</td>
</tr>
<tr>
<td></td>
<td>Currently, AR/CTL always saves the virtual storage area at the unit of work (UOW) boundary, as indicated by the checkpoint call.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>the portion of the $ARCSPVS parameter block that is reserved for use by AR/CTL</td>
<td>X(412)</td>
<td>LOW-VALUES</td>
</tr>
<tr>
<td></td>
<td>This parameter is required.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using the Define function

For each virtual storage area to be managed, use the Define function of the ARCS PV S API by initializing the parameters in the $ARCSPVS parameter block with the values that you want to use. Then issue a call to ARCS PV S and pass the initialized parameter block with the call.

If your compiler supports the use of the \texttt{SET field TO ADDRESS OF field} statement, you can use the Define function as shown in Figure 23.

Figure 23 Define function of the ARCS PV S API with SET usage

\begin{verbatim}
MOVE '$ARCSPVS' TO SPVS-IDENTIFIER.
MOVE 'DEFINE' TO SPVS-FUNCTION.
MOVE 'ARCSPVSC-0' TO SPVS-NAME.
MOVE ALL ZEROES TO SPVS-RETURN.
SET SPVS-BEGIN TO ADDRESS OF VS0-BEGIN.
SET SPVS-DELIMITER TO ADDRESS OF VS0-DELIMITER.
MOVE 'DEFINE' TO SPVS-REFRESH-INTERVAL.
MOVE 'UOW' TO SPVS-SAVE-INTERVAL.
MOVE ALL LOW-VALUES TO SPVS-RESERVED.
CALL 'ARCSPVS' USING
    ARCS PV S-PARAMETER-BLOCK.
\end{verbatim}

If your compiler does not support the use of the \texttt{SET field TO ADDRESS OF field} statement, you can call ARCS PAD to provide the addresses of the beginning and delimiter of the virtual storage area as shown in Figure 24.

Figure 24 Define function of the ARCS PV S API with ARCS PAD usage

\begin{verbatim}
MOVE '$ARCSPVS' TO SPVS-IDENTIFIER.
MOVE 'DEFINE' TO SPVS-FUNCTION.
MOVE 'ARCSPVSC-0' TO SPVS-NAME.
MOVE ALL ZEROES TO SPVS-RETURN.
CALL 'ARCSPAD' USING
    VS0-BEGIN,
    SPVS-BEGIN.
CALL 'ARCSPAD' USING
    VS0-DELIMITER,
    SPVS-DELIMITER.
MOVE 'DEFINE' TO SPVS-REFRESH-INTERVAL.
MOVE 'UOW' TO SPVS-SAVE-INTERVAL.
MOVE ALL LOW-VALUES TO SPVS-RESERVED.
CALL 'ARCSPVS' USING
    ARCS PV S-PARAMETER-BLOCK.
\end{verbatim}
For the Define function, set the parameter values as described in Table 8.

**Table 8  $ARCSPVS parameter block fields for the Define function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Define function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>$ARCSPVS.</td>
</tr>
<tr>
<td>Function</td>
<td>DEFINE.</td>
</tr>
<tr>
<td>Name</td>
<td>the name of the virtual storage area to be managed</td>
</tr>
<tr>
<td></td>
<td>The name must not be blank or null, and must be unique within the job step until it</td>
</tr>
<tr>
<td></td>
<td>is deleted or the job is cold-started.</td>
</tr>
<tr>
<td>Return Code</td>
<td>ALL ZEROES</td>
</tr>
<tr>
<td></td>
<td>The application program can check the Return Code parameter after the ARCSPVS API</td>
</tr>
<tr>
<td></td>
<td>returns control to the program. If the value is 4, the requested operation was</td>
</tr>
<tr>
<td></td>
<td>successful and the virtual storage area was refreshed.</td>
</tr>
<tr>
<td>Begin</td>
<td>beginning address that defines the location of the virtual storage area</td>
</tr>
<tr>
<td>Delimiter</td>
<td>address that terminates the virtual storage area (and is not included in the area)</td>
</tr>
<tr>
<td>Refresh Interval</td>
<td>DEFINE if you want AR/CTL to restore the area during initialization when the program</td>
</tr>
<tr>
<td></td>
<td>calls the ARCSPVS API with the DEFINE function code</td>
</tr>
<tr>
<td></td>
<td>MANUAL if you want AR/CTL to restore the area when the program call the ARCSPVS API</td>
</tr>
<tr>
<td></td>
<td>with the REFRESH function code</td>
</tr>
<tr>
<td>Save Interval</td>
<td>UOW</td>
</tr>
<tr>
<td>Reserved</td>
<td>LOW-VALUES</td>
</tr>
<tr>
<td></td>
<td>This area is required and might be used by AR/CTL.</td>
</tr>
</tbody>
</table>

**Using the Refresh function**

If you set the REFRESH value for the Refresh Interval parameter of the Define function, use the Refresh function as shown in Figure 25. This function manually refreshes the virtual storage area to its state at the last time it was saved.

**Figure 25  Refresh function of the ARCSPVS API**

```
MOVE '$ARCSPVS' TO SPVS-IDENTIFIER.
MOVE 'REFRESH' TO SPVS-FUNCTION.
MOVE 'ARCEPVSC-0' TO SPVS-NAME.
CALL 'ARCSPVS' USING
ARCSPVS-PARAMETER-BLOCK.
```
Using the Delete function

For the Refresh function, set the parameter values as described in Table 9.

Table 9  $ARCSPVS parameter block fields for the Refresh function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Refresh function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>$ARCSPVS</td>
</tr>
<tr>
<td>Function</td>
<td>REFRESH</td>
</tr>
<tr>
<td>Name</td>
<td>must identify a previously defined virtual storage area</td>
</tr>
<tr>
<td>Return Code</td>
<td>ALL ZEROES</td>
</tr>
<tr>
<td></td>
<td>The application program can check the Return Code parameter after the ARCSPVS API returns control to the program. If the value is 4, the requested operation was successful and the virtual storage area was refreshed.</td>
</tr>
<tr>
<td>Beginning Address</td>
<td>ignored</td>
</tr>
<tr>
<td>Delimiter</td>
<td>ignored</td>
</tr>
<tr>
<td>Refresh Interval</td>
<td>ignored</td>
</tr>
<tr>
<td>Save Interval</td>
<td>ignored</td>
</tr>
<tr>
<td>Reserved</td>
<td>LOW-VALUES</td>
</tr>
<tr>
<td></td>
<td>This area is required and might be used by AR/CTL.</td>
</tr>
</tbody>
</table>

Using the Delete function

When the application program no longer needs AR/CTL to manage the virtual storage area, you can use the Delete function as shown in Figure 26. This function removes the definition of the virtual storage area, and AR/CTL no longer manages the area.

Figure 26  Delete function of the ARCSPVS API

MOVE '$ARCSPVS' TO SPVS-IDENTIFIER.
MOVE 'DELETE' TO SPVS-FUNCTION.
MOVE 'ARCSPVSC-0' TO SPVS-NAME.
CALL 'ARCSPVS' USING
ARCSPVS-PARAMETER-BLOCK.

For the Delete function, set the parameter values as described in Table 10.

Table 10  $ARCSPVS parameter block fields for the Delete function (part 1 of 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Delete function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>$ARCSPVS</td>
</tr>
<tr>
<td>Function</td>
<td>DELETE</td>
</tr>
<tr>
<td>Name</td>
<td>must identify a previously defined virtual storage area</td>
</tr>
</tbody>
</table>
Using AR/CTL common calls

This section describes how to use AR/CTL common calls to obtain checkpoint/restart services. These calls are valid in all environments that AR/CTL supports. You can use AR/CTL common calls and AR/CTL-format checkpoint restart calls in the same program, or you can use AR/CTL common calls and IMS-format checkpoint restart calls in the same program.

Using the AR/CTL common call

The function code of the common call determines whether AR/CTL performs restart or checkpoint processing. The restart call should specify user areas identical to those specified in the checkpoint calls. If the areas are different, the areas in the checkpoint calls will be used; however, the lengths must be identical for AR/CTL to accomplish restart.

**WARNING**

If an application program uses checkpoint/restart support or any AR/CTL data services, it must issue the restart call once as the first call and before any data set open processing.

**Call Format**

The following shows the format of the common call in COBOL. All parameters are positional.

```
CALL 'CBLTARC' USING function,statuscd,forceopt.
```
Call parameters

The following parameters are valid on the AR/CTL restart call:

**langTARC**
Required. The literal that identifies the call as the AR/CTL common call, where `lang` is one of the following programming languages:

<table>
<thead>
<tr>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBL</td>
<td>COBOL</td>
</tr>
<tr>
<td>ASM</td>
<td>Assembler</td>
</tr>
<tr>
<td>PLI</td>
<td>PL/I</td>
</tr>
</tbody>
</table>

**function**
Required. The name of the 4-byte field that identifies the function of the call. The following functions are valid:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XRST</td>
<td>restart request</td>
</tr>
<tr>
<td>CHKP</td>
<td>checkpoint request</td>
</tr>
</tbody>
</table>

**statuscd**
Required. The name of the 2-byte field that receives the status code from AR/CTL.

**forceopt**
Required; ignored for the XRST function code. The name of the 3-byte field that contains the checkpoint force option. The following values are valid:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>Forces checkpoint processing to complete.</td>
</tr>
<tr>
<td>NO</td>
<td>Accepts the checkpoint pacing recommendation. To pad the field the value must be followed by a blank (X'40').</td>
</tr>
</tbody>
</table>

**ioarealn**
Required. The name of the 4-byte binary field, PICTURE S9(8) COMP, that contains the length of the I/O area. The minimum length is 12 bytes for the XRST function code and 8 bytes for the CHKP function code. For more information, see “Coding an I/O area” on page 193.
**ioarea**
Required. The name of the field that receives the restart identifier from AR/CTL and that the program can use to communicate a checkpoint identifier to AR/CTL. The program can leave this area blank. For more information, see “Coding an I/O area” on page 193.

It is *not* necessary for any of the following user areas to begin or end on a fullword boundary.

**beginarea-1**
Required. The name of a field that contains the beginning address of the first program data area to save in the checkpoint record or to restore at restart.

**endarea-1**
Required. The name of a field that marks the ending address of the first program data area to save in the checkpoint record or to restore at restart. This field is not included in the save area.

**beginarea-n**
Optional. The name of a field that contains the beginning address of the next program data area to save in the checkpoint record or to restore at restart. You can specify a maximum of seven areas.

**endarea-n**
Optional. The name of a field that marks the ending address of the next program data area to save in the checkpoint record or to restore at restart. This field is not included in the save area.

---

**Coding an I/O area**

In the working storage of the application program, identify the I/O area to use for the common call. AR/CTL uses this area to communicate a restart ID to the program, and the program can use this area to communicate a checkpoint ID to AR/CTL.

The application program must initialize the I/O area to blanks before issuing the restart call.

In response to the restart request, AR/CTL inserts the restart ID into the I/O area if the program is restarting. The program should look at this area and take appropriate action for a restart, if necessary. AR/CTL leaves this area as blanks (X'40') if the application program is starting normally.
The program can insert a checkpoint ID into this area before issuing a checkpoint request, or the program can leave this area blank. If this area is blank, AR/CTL automatically generates a checkpoint ID for the program:

- For IMS programs, AR/CTL returns the generated checkpoint ID or message from IMS in this area.
- For IMS-compatible, DB2-only, VSAM-only, and DB2/VSAM programs, AR/CTL returns the generated checkpoint ID in this area.

The checkpoint identifier will be incremented automatically for all AR/CTL common CHKP calls if the checkpoint identifier in the first AR/CTL common CHKP call contains all blanks (X'40').

The following example shows the restart ID area as coded in COBOL:

```cobol
IDENTIFICATION DIVISION.  
...  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
01 XRSTID-AREA.  
   02 XRSTID PICTURE X(12).  
```

**Example program using the AR/CTL common call**

*Figure 27* shows fragments of a COBOL application program that uses AR/CTL common calls.

### Figure 27  AR/CTL common call example COBOL program

```cobol
WORKING-STORAGE SECTION.  
01 FUNCTION PICTURE X(4).  
01 STATUSCD PICTURE X(2)  
   VALUE SPACES.  
01 FORCEOPT PICTURE X(3)  
   VALUE 'NO '.  
01 IOAREALN PICTURE S9(8)  
   USAGE COMPUTATIONAL  
   VALUE +12.  
01 IOAREA PICTURE X(12)  
   VALUE SPACES.  
01 AREA0BEG...  
01 AREA0END...  
01 AREA1BEG...  
01 AREA1END...  
01 AREA2BEG...  
01 AREA2END...  
01 AREA3BEG...  
01 AREA3END...  
```
Using IMS-compatible checkpoint/restart calls

This section describes how to use IMS-compatible checkpoint/restart calls. Table 11 summarizes information about the IMS-format calls that an IMS-compatible application program can issue to request checkpoint/restart support.

For convenience, this manual refers to the calls that the application program issues to the langTDLI module (where lang is the programming language) by the function code in the calls. For example, the XRST call is used to refer to the call to langTDLI with the XRST function code.

Table 11  IMS-compatible checkpoint/restart call summary

<table>
<thead>
<tr>
<th>Function code</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XRST</td>
<td>checkpoint/restart</td>
<td>set up the extended checkpoint/restart environment, invoke AR/CTL to perform restart in restart situations</td>
</tr>
<tr>
<td>CHKP</td>
<td>checkpoint/restart</td>
<td>initiate checkpoint processing</td>
</tr>
<tr>
<td></td>
<td>support</td>
<td>Checkpoint pacing requirements determine whether this call results in an actual checkpoint taken.</td>
</tr>
</tbody>
</table>

Figure 27  AR/CTL common call example COBOL program

```
PROCEDURE DIVISION.
MOVE 'XRST' TO FUNCTION.
CALL 'CBLTARC' USING FUNCTION,STATUSCD,FORCEOPT,IOAREALN,
       IOAREA,AREA0BEG,AREA0END,AREA1BEG,AREA1END,
       AREA2BEG,AREA2END,AREA3BEG,AREA3END,
       AREA4BEG,AREA4END,AREA5BEG,AREA5END,
       AREA6BEG,AREA6END.
IF STATUSCD IS EQUAL TO SPACES
   THEN
   NEXT SENTENCE
   ELSE
   PERFORM STATUSCDERROR.
```
Entry and return

In an application program, you must identify the program entry point—the first program instruction to execute when the program gets control from AR/CTL. The program must also return control to AR/CTL. If the program already contains these structures for IMS, no changes are necessary.

At the entry point, the application program must define the list of program communication blocks (PCBs) it will access. The list of PCBs must be in the same order that the PCBs are coded in the PSB or application specification block (ASB). The checkpoint (I/O) PCB, if used, must be the first PCB in the list. Each AR/CTL call must access one of these PCBs to use for the call.

Use the PROCEDURE statement or the ENTRY statement as the entry point. The following example shows the use of the procedure statement:

```
PROCEDURE DIVISION USING CHKPPCB, ASAMPCB1, ASAMPCB2, ...
```

If you use the ENTRY statement, it must be the first executable statement in the procedure division. The ENTRY statement must list the names of the PCBs. The following example shows the use of the ENTRY statement:

```
ENTRY 'ANYNAME' USING CHKPPCB, ASAMPCB1, ASAMPCB2, ...
```

Use the GOBACK statement to return control to AR/CTL.

```
GOBACK
```

Call function codes

In the application program, identify the function codes the program will use in the IMS-compatible calls to AR/CTL. The program uses a variable to identify each function code; the value of the variable is the actual function code. If the program already contains these structures for IMS, no changes are necessary.

You can use any name for the function variables, including the value of the function itself, within the conventions of the programming language. The value must be 4 bytes long. For function codes less than 4 bytes (such as GU and GN), pad the value with trailing blanks.
Define call function codes in the working storage section. Use level 77 or level 01–49 entries. You can use a COBOL COPY statement to include these standard descriptions in the application program:

```
77 XRST PIC X(4) VALUE 'XRST'.
77 CHKP PIC X(4) VALUE 'CHKP'.
```

### Using the XRST call

Use the XRST call to invoke AR/CTL checkpoint/restart support and to enable AR/CTL data services. This call results in a normal start, cold start, or restart environment, depending on the AR/CTL restart determination routines.

---

**WARNING**

If an application program uses checkpoint/restart support or any AR/CTL data services, it must issue the restart call once as the first call and before any data set open processing.

---

When the application program issues the XRST call, AR/CTL repositions the database and GSAM files. If the application program is using DB2 but does not use the Automatic DB2 Cursor Repositioning processing option, the application program must reposition the DB2 tables by using the key information in a data area saved in the CHKP call.

---

**NOTE**

The XRST call must specify user areas identical to those specified in the CHKP calls.

---

### Call Format

The following example shows the XRST call in COBOL:

```
CALL 'CBLTDLI' USING XRST, chkppcb, chkplen, chkparea,
     userlen-1, userarea-1, userlen-n, userarea-n
```

### Call Parameters

The following parameters are valid on the XRST call

- **chkppcb**
  - Required. The checkpoint PCB. This PCB is defined in the ASB.

- **chkplen**
  - Required. The address of any four-byte area in the program. AR/CTL does not use this area, but a value is required for compatibility with IMS.
Using the XRST call

chkarea
Required. The name of the 8-byte checkpoint I/O area. For information about how to define this area in the application program, see “Coding the checkpoint I/O area” on page 201.

**NOTE**
At least one user area is required in an IMS-compatible environment (where IMS is not present during the execution). It is not necessary for any user area to begin or end on a fullword boundary.

userlen-1
Required. The name of a 4-byte field that contains the binary length of the first application program data area to restore. These values must define areas that were saved during checkpoint processing.

**NOTE**
If a user area length is specified incorrectly, results will be unpredictable.

userarea-1
Required. The name of the first data area to save in the checkpoint record. It is usually a structure name.

userlen-n, userarea-n
Optional. The address of the length and the name of the next data area to save in the checkpoint record. Provide a pair of userlen-n and userarea-n parameters for each defined data area you want to restore. You can identify a maximum of seven data areas.

Status codes
AR/CTL does not issue any status codes for this call.

**NOTE**
After the XRST call, the application program should check any ASAM PCBs for proper status codes. AR/CTL places the status codes associated with opening/repositioning ASAM data sets in the PCB for that data set.
Using the CHKP call

Use the CHKP call to invoke AR/CTL checkpoint/restart support to initiate an application program checkpoint. The checkpoint routine may not take an actual checkpoint, depending on the action taken by the checkpoint pacing routines and/or checkpoint exits. The application program can check an optional user-designated status code returned to the checkpoint PCB to determine whether an actual checkpoint occurred.

The CHKP call includes the checkpoint ID to identify a specific checkpoint. The CHKP call also identifies one to seven areas of program storage for AR/CTL to save during the CHKP call and to restore, if necessary, during the XRST call.

NOTE

The application program should save the same user areas in the CHKP call as the areas it uses in the XRST call. Saving different areas is not recommended. If the application program saves different areas, it must ensure that AR/CTL restores these fields during restart by matching them on the XRST call. Saving areas of varying lengths in different CHKP calls is not a valid coding practice.

The application program should issue the CHKP call at the end of an application program’s logical UOW. The application program also should issue the CHKP call immediately after the XRST call and at the end of an update job step before termination.

Issuing the CHKP call at the end of an update job step is especially important if the application program is using both IMS and DB2. If the application program does not issue this final CHKP call and a failure occurs after the program terminated but before the job step ended, it is possible for DB2 to commit the data before DL/I commits the data. The DB2 data would be out of sync with the DL/I changes. Issue a CHKP call at the end of any update job step to coordinate the commit of the outstanding UOW for IMS and DB2. If the job step uses checkpoint pacing, BMC Software recommends that the program ensure the checkpoint is not bypassed by using the AR/CTL common checkpoint call with the force option or by using a checkpoint user exit routine.

CURSOR WITH HOLD Option

AR/CTL does not change the cursor to use the WITH HOLD option. If the program uses AR/CTL checkpoint services and the cursor does not use the WITH HOLD option, the program must tolerate cursor position being lost when a commit is performed.

If the application program is using CURSOR WITH HOLD, the program does not need to reposition the DB2 tables after a CHKP call. One of the main reasons for issuing checkpoint calls/commits when using CURSOR WITH HOLD is to release DB2 locks. These locks will be released during checkpoint/commit processing, CLOSE CURSOR, or termination.
Using the CHKP call

```call 'CBLTDLI' using chkppcb, chkplen, chkparea, userlen-1, userarea-1, ..., userlen-n, userarea-n```

chkppcb
- Required. The checkpoint PCB. This PCB is defined in the ASB.

chkplen
- Required. The address of any 4-byte area in the program. AR/CTL does not use this area, but a value is required for compatibility with IMS.

chkparea
- Required. The name of the 8-byte checkpoint I/O area. For information about how to define this area in the application program, see "Coding the checkpoint I/O area" on page 201.

**NOTE**

At least one user area is required in an IMS-compatible environment (where IMS is not present during the execution). It is not necessary for any user area to begin or end on a fullword boundary.

userlen-1
- Required. The name of a 4-byte field that contains the binary length of the first application program data area to save in the checkpoint record. The following example shows the COBOL specification for the user area length on the checkpoint (CHKP) call:

```cobol
01 AREALEN1 PIC S9(5) COMP.
```

**NOTE**

If a user area length is specified incorrectly, results will be unpredictable.

userarea-1
- Required. The name of the first data area to save in the checkpoint record. It is usually a structure name.

userlen-n, userarea-n
- Optional. The address of the length and the name of the next data area to save in the checkpoint record. Provide a pair of `userlen-n` and `userarea-n` parameters for each defined data area you want to save. You can identify a maximum of seven data areas.
Coding the checkpoint I/O area

An application program must identify a checkpoint I/O area for the checkpoint ID that it passes to AR/CTL as a parameter of the CHK call. The application program also can use this area for a checkpoint ID to use with the XRST call. To be compatible with IMS, the length must be set to 12 bytes.

The application program must initialize the checkpoint I/O area to blanks before issuing the restart call.

Define the checkpoint I/O area as a level 01 entry in working storage. You can further define the area with level 02 entries. The following example shows a typical checkpoint I/O area in COBOL:

```
IDENTIFICATION DIVISION.
... 
DATA DIVISION.
WORKING-STORAGE SECTION.
01 CKPAREA.
   02 CHKPID PICTURE X(8).
   02 FILLER PICTURE X(4).
```

Coding the checkpoint PCB mask

An application program uses the checkpoint PCB mask to check the status code that AR/CTL returns to report the results of checkpoint, restart, or rollback processing. The checkpoint PCB mask contains the same fields in the same order as defined in the checkpoint PCB.

The checkpoint PCB mask is compatible with the IMS I/O PCB mask. The only significant field for an IMS-compatible program is the two-character status code, which is located at offset 10.

Table 12 describes the fields in the checkpoint PCB mask.

<table>
<thead>
<tr>
<th>Field</th>
<th>Length (bytes)</th>
<th>Offset (bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reserved</td>
<td>10</td>
<td>0</td>
<td>retained for compatibility with IMS</td>
</tr>
</tbody>
</table>
Using ASAM services through GSAM replacement

This section describes the ASAM services that are available through GSAM replacement and summarizes the requirements for implementing ASAM services through GSAM replacement. GSAM replacement is provided with AR/CTL for IMS. It is for use in IMS and IMS-compatible application programs.

**NOTE**

If an IMS program uses a program option member that contains any data set option members specifying sequential interception, local VSAM access, or remote VSAM access, AR/CTL forces the use of GSAM replacement.

To use GSAM replacement, the application program issues IMS-format GSAM calls. These calls are identical to IMS GSAM calls; if the program already uses GSAM calls, no change is necessary.

Table 13 summarizes information about the IMS-format calls that an IMS or IMS-compatible application program can issue to request ASAM support.

---

**Table 12  Checkpoint PCB mask fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Length (bytes)</th>
<th>Offset (bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status code</td>
<td>2</td>
<td>10</td>
<td>describes the results of the call</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AR/CTL places a status code in this field after each call. The application program should check the status code to determine whether the call was successful.</td>
</tr>
<tr>
<td>reserved</td>
<td>28</td>
<td>12</td>
<td>retained for compatibility with IMS</td>
</tr>
</tbody>
</table>

Define the checkpoint PCB mask as a level 01 linkage section entry. The following example shows a checkpoint PCB mask in COBOL:

```
LINKAGE DIVISION.
...
01 CHKPPCB.
   02 FILLER PICTURE X(10).
   02 CHKP-STATUS PICTURE XX.
   02 FILLER PICTURE X(28).
```
Entry and return

Use the PROCEDURE statement or the ENTRY statement as the entry point. The following example shows the use of the PROCEDURE statement:

```
PROCEDURE DIVISION USING CHKPPCB, ASAMPCB1, ASAMPCB2, ...
```

If you use the ENTRY statement, it must be the first executable statement in the procedure division. The ENTRY statement must list the names of the PCBs. The following example shows the use of the ENTRY statement:

```
ENTRY 'ANYNAME' USING CHKPPCB, ASAMPCB1, ASAMPCB2, ...
```

Use the GOBACK statement to return control to AR/CTL:

```
GOBACK
```

Call function codes

Define call function codes in the working storage section. Use level 77 or level 01 to 49 entries. You can use a COBOL COPY statement to include these standard descriptions in the application program:

```
77 GU PIC X(4) VALUE 'GU'.
77 GN PIC X(4) VALUE 'GN'.
77 ISRT PIC X(4) VALUE 'ISRT'.
```

Table 13  GSAM replacement ASAM call summary

<table>
<thead>
<tr>
<th>Function code</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLSE</td>
<td>Explicitly close an ASAM data set.</td>
<td>page 208</td>
</tr>
<tr>
<td>GN</td>
<td>Retrieve the next sequential record from an ASAM data set.</td>
<td>page 204</td>
</tr>
<tr>
<td>GU</td>
<td>Retrieve a unique record from an ASAM data set, as identified by the RSA.</td>
<td>page 206</td>
</tr>
<tr>
<td>ISRT</td>
<td>Insert a segment in an ASAM data set.</td>
<td>page 207</td>
</tr>
<tr>
<td>OPEN</td>
<td>Explicitly open an ASAM data set.</td>
<td>page 204</td>
</tr>
</tbody>
</table>
OPEN call

Use the OPEN call to open an ASAM data set explicitly. This call normally is unnecessary because AR/CTL opens the ASAM data set automatically during XRST call processing. An application program can issue the OPEN call to open an ASAM data set for additional processing after AR/CTL has closed it automatically or because the application program issued the CLSE call.

**WARNING**

If the application program uses checkpoint/restart services, the restart call must precede data set open processing.

```call
CALL 'CBLTDLI' USING OPEN, asampcb, openmode
```

**asampcb**
Required. The ASAM PCB. This PCB is defined in the ASB or PSB.

**openmode**
Optional. The address of a code that indicates the mode for opening the ASAM data set. The following values are valid:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INP</td>
<td>input</td>
</tr>
<tr>
<td>OUT</td>
<td>standard output</td>
</tr>
<tr>
<td>OUTA</td>
<td>output with ISO/ANSI control characters</td>
</tr>
<tr>
<td>OUTM</td>
<td>output with machine control characters</td>
</tr>
<tr>
<td>blank</td>
<td>The call worked correctly.</td>
</tr>
<tr>
<td>AD</td>
<td>A call function is invalid.</td>
</tr>
<tr>
<td>AI</td>
<td>AR/CTL could not open the data set.</td>
</tr>
</tbody>
</table>

GN call

Use the GN call to get (read) the next sequential record from an ASAM data set. If this call is the first request for a record, AR/CTL opens the data set.
CALL 'CBLTDLI' USING GN, asampcb, asamarea, rsa

asampcb
Required. The ASAM PCB. This PCB is defined in the ASB or PSB.

asamarea
Required. The name of the ASAM I/O area that contains the data retrieved from or written to the ASAM data set. For more information, see “ASAM I/O area” on page 209.

rsa
Optional. The name of a field containing the record search argument (RSA). Include this parameter to request the RSA for the record. If the FCB was generated with the WRITERSA=NO keyword and the data set resides on DASD, AR/CTL returns the RBN instead of the TTR or the RBA. The RBN cannot be used with the GU call if the data set resides on DASD. For more information, see “Record search arguments” on page 211.

**NOTE**
Use of the RSA on a GN call causes significant overhead. BMC Software recommends that you do not use it for large data sets.

blank
The call worked correctly. The I/O area contains the requested data.

AD
A call function is invalid.

AF
The data set contains variable length records, and the record contains an invalid length field.

AI
AR/CTL could not open the data set.

AM
The request is invalid for this data set.

AO
An unrecoverable I/O error has occurred for the file.

GB
AR/CTL has reached the end of the input data set. If the application program issues another get request, AR/CTL reads the first record in the data set.
GU call

Use the GU call to get (read) a specific record from an ASAM data set. If this call is the first request for a record, AR/CTL opens the data set. The call must identify the record to read with the RSA. The WRTRSA(YES) keyword must be set in the FCB generation.

Call Format

The following example shows the format of the GU call in COBOL:

```
CALL 'CBLTDLI' USING GU, asampcb, asamarea, rsa
```

Call parameters

The following parameters are valid on the GU call:

**asampcb**
Required. The ASAM PCB. This PCB is defined in the ASB or PSB.

**asamarea**
Required. The name of the ASAM I/O area that contains the data retrieved from or written to the ASAM data set. For more information, see “ASAM I/O area” on page 209.

**rsa**
Required. The name of a field containing the RSA or the name of the field that is to receive the RSA from an ISRT call. The application program can use a special RSA, value 0000000100000000, to retrieve the first record in the data set. For more information, see “Record search arguments” on page 211.

**blank**
The call worked correctly. The I/O area contains the requested data.

**AD**
A call function is invalid.

**AF**
The data set contains variable length records, and the record contains an invalid length field.

**AH**
The application program issued a GU call without a valid RSA, or the RSA parameter was not included in the call.

**AI**
AR/CTL could not open the data set.
ISRT call

Use the ISRT call to insert (add) a new record to the end of the ASAM data set. If the FCB has requested output staging for the data set, AR/CTL can delay writing the record to the data set until the next CHKP call or termination.

Call Format

The following example shows the ISRT call in COBOL:

```cobol
CALL 'CBLTDLI' USING ISRT, asampcb, asamout, rsa
```

Call parameters

The following parameters are valid on the ISRT call:

**asampcb**
Required. The PCB. This PCB is defined in the ASB or PSB.

**asamarea**
Required. The name of the ASAM I/O area that contains the data to write to the ASAM data set. For more information, see “ASAM I/O area” on page 209.

**rsa**
Optional. The name of the field to receive the RSA from the call. If the WRITERSA=YES keyword is set in the FCB generation, the call can request that AR/CTL return the RSA for the added record by including the RSA parameter. If AR/CTL is staging output for a DASD data set, it returns the RBN instead of the RSA. It cannot return the TTR; therefore, the GU call is not valid for a staged-output DASD data set. For more information, see “Record search arguments” on page 211.

**blank**
The call worked correctly. The data has been added to the data set.
AD
A call function is invalid.

AI
AR/CTL could not open the data set.

AM
The request is invalid for this data set.

AO
An I/O error occurred during data set processing.

IX
The application program attempted to add a record to the data set after an I/O error occurred, or the data set is no longer open.

V1
The data set contains variable length records, and the record contains an invalid length field.

**CLSE call**

Use the \texttt{CLSE} call to close an ASAM data set explicitly. This call is optional because AR/CTL closes the ASAM data set automatically during termination processing. An application program can issue the \texttt{CLSE} call to close an ASAM data set it has explicitly opened or in preparation for opening it for other processing. If the application program does not use checkpoint/restart support (and, therefore, does not issue the \texttt{XRST} call), the program should issue the \texttt{CLSE} call before it terminates.

The use of the \texttt{CLSE} call allows the application program to ensure that all records have been written successfully to the data set. If the application program terminates without a \texttt{CLSE} call, output records in buffers could experience write errors that the application program does not detect.

\begin{verbatim}
CALL 'CBLTDLI' USING CLSE, asampcb
\end{verbatim}

\texttt{asampcb}
The ASAM PCB. This PCB is defined in the ASB or PSB.
ASAM I/O area

An application program must identify an I/O area to contain the segments to read from or write to the ASAM data set. The format of this I/O area is the same as an IMS GSAM I/O area:

- If the record format is variable, the first two positions of the I/O area contain the length of the record.
- If the data set contains control characters, the first position contains the control character for fixed and undefined record types and the third position contains the control character for variable record formats.

The length of the rest of the ASAM I/O area depends on the length of the data to store in (or retrieve from) the ASAM data set.

Define the ASAM I/O area as a level 01 entry in working storage. You can further define the area with level 02 entries. The following example shows a typical ASAM I/O area in COBOL. The area contains three character-type fields (4 bytes, 16 bytes, and 32 bytes, respectively):

```
IDENTIFICATION DIVISION.
... DATA DIVISION.
WORKING-STORAGE SECTION.
01 ASAMAREA.
   02 ASAMFLD1 PICTURE X(4).
   02 ASAMFLD2 PICTURE X(16).
   02 ASAMFLD3 PICTURE X(32).
```

ASAM PCB mask

An application program must define one ASAM PCB mask for each ASAM data set it accesses. The application program uses the ASAM PCB mask to check the status code that AR/CTL returns to report the results of an ASAM data set call. The ASAM PCB mask contains the same fields in the same order as defined in the ASAM PCB. Table 14 on page 210 describes the fields in the ASAM PCB mask.

When AR/CTL returns control to the application program, it places a two-byte status code in a field of the ASAM PCB to report the results of the call. The application program should check this status code field after each call. AR/CTL supports all status codes that IMS supports for GSAM. This guide documents only the codes that AR/CTL issues in an IMS-compatible program. For codes that IMS issues, see the appropriate IMS documentation.
ASAM can provide the same RSA that IMS returns to the application program for GSAM. ASAM returns the RSA only when the call parameter list contains a request to return the RSA.

### Table 14  ASAM PCB mask fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Length (bytes)</th>
<th>Offset (bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database name</td>
<td>8</td>
<td>0</td>
<td>ddname of the ASAM data set</td>
</tr>
<tr>
<td>reserved</td>
<td>2</td>
<td>8</td>
<td>retained for compatibility with IMS</td>
</tr>
<tr>
<td>status code</td>
<td>2</td>
<td>10</td>
<td>status code that describes the results of the call</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AR/CTL places a status code in this field after each call to the ASAM data set. The application program should check the status code to determine whether the call was successful.</td>
</tr>
<tr>
<td>processing option</td>
<td>4</td>
<td>12</td>
<td>processing option code that defines the types of calls the application program can issue</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The value in this field is the value of the PROCOPT parameter on the ARCPBCB or PCB statement in the ASB or PSB. This value does not change during application program execution.</td>
</tr>
<tr>
<td>reserved</td>
<td>4</td>
<td>16</td>
<td>retained for compatibility with IMS</td>
</tr>
<tr>
<td>reserved</td>
<td>8</td>
<td>20</td>
<td>retained for compatibility with IMS</td>
</tr>
<tr>
<td>field length</td>
<td>4</td>
<td>28</td>
<td>decimal value 12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The value in this field is the sum of the lengths of the record search argument fields and undefined-length records area field in the ASAM PCB mask.</td>
</tr>
<tr>
<td>reserved</td>
<td>4</td>
<td>32</td>
<td>retained for compatibility with IMS</td>
</tr>
<tr>
<td>record search argument area</td>
<td>8</td>
<td>36</td>
<td>the address of the record that AR/CTL returned as the result of a get-type call</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To retrieve this record directly, you can use the address of a field containing this RSA as one of the parameters of a get unique (GU) call.</td>
</tr>
<tr>
<td>length of undefined-length records</td>
<td>4</td>
<td>42</td>
<td>if the ASAM data set has an undefined record format (RECFM=U), the binary length of the record being processed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For GU or GN calls, AR/CTL places the binary length of the retrieved record in this field. For ISRT calls, the application program must place the binary length of the inserted record in this field before issuing the ISRT call.</td>
</tr>
</tbody>
</table>

Define the ASAM PCB mask as a level 01 linkage section entry. The following example shows a typical ASAM PCB mask in COBOL:

```cobol
LINKAGE DIVISION.
...
```
Record search arguments

By including the RSA parameter in the call parameter list, an application program can identify a particular record inserted to or retrieved from an ASAM data set. The RSA is required on a GU call. AR/CTL can emulate the RSA that IMS returns to the application program for GSAM support. The RSA is eight bytes long and has the format that is described in Table 15, depending on the data set type.

<table>
<thead>
<tr>
<th>Data set type</th>
<th>Offset</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DASD (non-VSAM)</td>
<td>0</td>
<td>4</td>
<td>TTR (the disk address in track-track-record format, which is the standard relative offset into a DASD data set)</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>volume sequence number</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>location of the record in the block</td>
</tr>
<tr>
<td>VSAM</td>
<td>0</td>
<td>4</td>
<td>relative byte address (RBA)</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>binary zeros</td>
</tr>
<tr>
<td>other</td>
<td>0</td>
<td>4</td>
<td>RBN</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>volume sequence number</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>location of the record in the block</td>
</tr>
</tbody>
</table>

The volume sequence number field is divided into two sections. The first byte contains the concatenation number if the data set is for input and is a member of a concatenated group. The second byte is the relative volume number in a data set.

For output data sets on DASD, the collection of RSAs can result in increased CPU usage. AR/CTL can bypass RSA emulation for these files. If the ISRT call includes the RSA parameter in the parameter list and if the WRITERSA(YES) keyword is coded in the FCB for the output data set, AR/CTL emulates RSAs for output data sets on DASD. Otherwise, AR/CTL does not emulate RSAs.
If the application program call includes an RSA parameter and AR/CTL is not emulating RSAs for the data set, AR/CTL returns the RBN of the block instead of the normal TTR or RBA. If the application program attempts to use an RSA and AR/CTL is not emulating RSAs for the DASD data set, AR/CTL returns the status code AH to the application program.

You can use a special RSA value of 0000000100000000 with the GU call to retrieve the first record in the data set.

### Using ASAM services through the ASAM callable interface

This section describes the ASAM services of AR/CTL that are available through the ASAM callable interface and summarizes the requirements for implementing ASAM services through the ASAM callable interface. The ASAM callable interface is available with AR/CTL for DB2 and AR/CTL for VSAM. The callable interface supports QSAM data sets, VSAM input and output ESDSs, and VSAM input KSDSs.

### ASAM call

To request ASAM services through the ASAM callable interface, an application program must issue ARCASAM calls. To replace an existing sequential data set with ASAM, change the application program to remove any open calls, close calls, and access request calls (such as get, put, read, and write). Then make all data requests directly to AR/CTL through the ASAM callable interface.

**WARNING**

If the application program uses checkpoint/restart services, the restart call must precede data set open processing.

```
CALL 'ARCASAM' USING arc-arb, asam-ioarea
```

**ARCASAM**

Required. The literal that identifies the call as the ASAM call.

**arc-arb**

Required. The name of a fullword-aligned field that contains the address of the ASAM ARB. The ARB is defined in the working storage of the application program. For more information, see “ASAM ARB” on page 214.
Before making an ASAM call, set the ARB field that indicates the function to use (OPEN, CLOSE, READ, WRITE, ROLD), as shown in the following example:

```
MOVE 'READ' TO ARB-FUNC.
```

**asam-ioarea**
Required. The name of the ASAM I/O area that contains the data retrieved from or written to the ASAM data set. For more information, see “ASAM I/O areas” on page 213.

**blank**
The call worked correctly. The I/O area contains the requested data for an input file. The data has been added to the file for an output file.

**AD**
A call function is invalid.

**AI**
AR/CTL could not open the data set.

**AJ**
The ASAM I/O area is invalid for the open function.

**AO**
An unrecoverable I/O error has occurred for the file, or the file has failed to open properly.

**GB**
AR/CTL has reached the end of the data set on input. If the application program issues another read-type call, AR/CTL reads the first record in the data set.

**XX**
An internal ASAM error has occurred.

---

**ASAM I/O areas**

An application program must identify an I/O area in working storage to contain the data to read from or write to each ASAM data set. The format of this I/O area uses standard conventions:

- If the record format is variable, the first two positions of the I/O area contain the length of the record.
If the data set contains control characters, the first position contains the control character for fixed and undefined record types and the third position contains the control character for variable record formats.

The length of the rest of the ASAM I/O area depends on the length of the data to store in (or retrieve from) the ASAM data set.

For an OPEN function, the ASAM I/O area can contain the following operands:

INP
input

OUT
standard output

OUTA
output with ISO/ANSI control characters

OUTM
output with machine control characters

The following example shows a typical ASAM I/O area in COBOL. The area contains three character-type fields (4 bytes, 16 bytes, and 32 bytes, respectively):

```
IDENTIFICATION DIVISION.
...
DATA DIVISION.
WORKING-STORAGE SECTION.
 01 ASAMAREA.
    02 ASAMFLD1 PICTURE X(4).
    02 ASAMFLD2 PICTURE X(16).
    02 ASAMFLD3 PICTURE X(32).
```

**ASAM ARB**

Application programs and AR/CTL use the ARB to exchange information such as call functions and results. Code the ARB in the working storage area of an application program. The program can use a single ARB for all calls, or it can use a separate ARB for checkpoint support and for each sequential file that uses ASAM support. If the program uses a single ARB for multiple call types, it must reinitialize the ARB for each call.

The following example shows the ARB in COBOL:

```
IDENTIFICATION DIVISION.
...
```
### ARB Type
This field identifies the ARB.

### Function Code
This field (at offset 8 from the beginning byte 0) indicates the type of access you want for the next call.

### DDname
This field identifies the ddname specified in the associated data set option member (also known as the FCB).

### Status Code
This field contains the status code that the application program can check to determine the results of a call to AR/CTL. The status codes that AR/CTL can return are documented with the calls.

### ULRECL
If the data set has an undefined record format and you are writing a record, set this field to the length of the record being written. If AR/CTL is reading a record, it sets this field to the length of the record being read.

### OPEN
Open the ASAM data set identified in the call. The ASAM I/O area can contain operands to indicate data set usage. For more information see “ASAM I/O areas” on page 213.

### CLSE
Close the ASAM data set identified in the call.

### READ
Read records from the ASAM data set identified in the call.

### WRITE
Write records to the ASAM data set identified in the call.

---

**DATA DIVISION.**
**WORKING-STORAGE SECTION.**
01 ARCARB.
   02 ARCARB-TYPE PIC X(8) VALUE 'ARB'.
   02 FUNC-CODE PIC X(8) VALUE 'READ'.
   02 DDNAME PIC X(8) VALUE 'ddname'.
   02 STATUS-CODE PIC X(2) VALUE SPACES.
   02 FILLER PIC X(2) VALUE SPACES.
   02 ULRECL PIC S9(8) COMP.
ROLD

Restore the staged-output ASAM data set identified in the call to the last application program checkpoint by purging all uncommitted data from the staging area. The **ROLD** function does not result in checkpoint processing, and AR/CTL deletes only the staged output for the ASAM data set referenced by the ARB you specify in the ASAM call. AR/CTL does not delete any staged output for other ASAM data sets and does not signal any DBMS to back out uncommitted updates. The **ROLD** function can help simplify application program logic. For example, if the application program holds output records in a buffer until a commit point, it can now issue the **WRITE** function to write the output records to a staged-output ASAM data set immediately. If an error occurs, the program can issue the ASAM call with the **ROLD** function to back out the changes. This function repositions an ASAM data set only if it uses output staging.

**NOTE**

ROLD processing is not available if the job step is using the local or remote VSAM access services of AR/CTL.
This chapter tells how to change PL/I application programs to use the explicit calls provided by the APPLICATION RESTART CONTROL (AR/CTL) products. This chapter contains the following information:

Overview ................................................................. 217
Supported versions of PL/I ........................................... 218
Types of AR/CTL calls ............................................... 218
Sample PL/I programs ............................................... 218
Coding considerations ............................................... 219
EXEC parameters .................................................... 219
Compile and link issues ............................................. 220
Run-time considerations ........................................... 221
Using AR/CTL-format checkpoint/restart calls ................ 221
Managing virtual storage for subprograms ....................... 222
Implementing the ARCSPVS API .................................. 223
Identifying virtual storage areas to be managed ................. 224
Creating the $ARCSPVS parameter block ....................... 225
Using the Define function ........................................ 227
Using the Refresh function ....................................... 229
Using the Delete function ........................................ 230
Using AR/CTL common calls .................................... 231
Using IMS-compatible calls ..................................... 232
Using ASAM callable interface calls ............................ 234

Overview

The easiest way to obtain AR/CTL services in an application program is with automated methods, which require no program changes. If you are not able to use the automated methods, you can change the application program to issue explicit requests for AR/CTL services. This section provides general information about making changes to application programs written in PL/I.
Before using the information in this chapter, review the overview information about AR/CTL checkpoint/restart and data services in Chapter 3, “Using AR/CTL checkpoint/restart services.”

**Supported versions of PL/I**

AR/CTL fully supports application programs written in OS PL/I and IBM PL/I for MVS and VM.

AR/CTL also supports the PL/I Enterprise compiler with considerations. For more information, see “Compile and link issues” on page 220.

**Types of AR/CTL calls**

AR/CTL supports several formats for the calls that request AR/CTL checkpoint/restart services:

- AR/CTL-format
- AR/CTL common format
- IMS-compatible

AR/CTL supports several formats for the calls that request application sequential access method (ASAM) services:

- IMS generalized sequential access method (GSAM) replacement
- ASAM callable interface

**Sample PL/I programs**

BMC Software provides executable samples of application programs that use AR/CTL calls. For a list of these samples, see Appendix B, “Locating samples.”
Coding considerations

If you code explicit checkpoint/restart calls, set up the user areas defined in the call as one aggregate variable (one DCL with multiple structures). This technique prevents problems if the PL/I optimizing compiler rearranges working storage. Here is an example:

```
DCL 1 ARC_WORK,
   5 AREA0BEG CHAR(12) INIT('ARC START'),
   5 ... 
   5 AREA0END CHAR(12) INIT('ARC END');
```

For PL/I application programs running under AR/CTL, turn off the abend interception functions of PL/I. Here is an example of one method:

```
ON ERROR
BEGIN;
   CALL PLITDLI (1,'ROLL');
END;
```

AR/CTL provides Assembler language subroutines to service explicit AR/CTL calls. In the declare statement for these calls, specify OPTIONS attribute keywords as shown in the following example:

```
DCL ARCXRST OPTIONS(ASM INTER);
```

In DB2 PL/I application programs that execute with AR/CTL, the following declare statement must not be in the application program:

```
DCL DSNALI OPTIONS(ASM INTER);
```

EXEC parameters

The following sections provide information that is related to EXEC statement parameters.

Length of parameter list

Normally, MVS allows a 100-byte PARM string for an EXEC PGM statement. However, ARC/CTL uses some of these bytes during processing. Therefore, 61 bytes is the maximum allowable length of the PARM string on the EXEC statement for an application program that uses AR/CTL.
Original operating system parameters

According to standard MVS linkage conventions, register 1 points to the original operating system parameter list. However, AR/CTL uses this register for other purposes and provides access to a copy of the original parameter list through MVS name token services.

Member ASMORIGP of the sample library contains model code (written in Assembler language) that you can adapt and include in an application program if the program needs access to a copy of the original parameter list. This sample shows how to fetch the operating system parameter list. The address of the list is located at offset +8 (byte 9) of the TOKEN field.

This sample also shows a quick way for the application program to detect whether AR/CTL is active. The sample sets the return code to zero if AR/CTL is active and to a nonzero value if AR/CTL is not active or a problem is detected.

Compile and link issues

A non-DB2 application program that uses any explicit AR/CTL call (AR/CTL-format checkpoint/restart calls, AR/CTL common calls, ASAM callable interface calls) must include the AR/CTL execution library in the SYSLIB concatenation of the link-edit JCL. The order of the libraries in the concatenation is not important.

A DB2 application program that uses any explicit AR/CTL call must be link-edited with the AR/CTL language interface module (ARCLI000) to replace the DB2 DSNHLI module. Member #DB2LNK in the AR/CTL sample library contains sample JCL for link-editing the application program with AR/CTL. Insert the AR/CTL execution load module library ahead of the DB2 load library (DSNLOAD).

The following order of libraries is correct:

1. AR/CTL load library
2. compiler (such as PL/I) run time library
3. IMS RESLIB library (if applicable)
4. DB2 DSNLOAD library (if applicable)
5. user load library (containing the external routines that are specific to your environment)
If you are using the PL/I Enterprise compiler and you are using AR/CTL to reposition sequential files, you must specify the BLOCKEDIO parameter when you compile the PL/I program. AR/CTL handles the data blocking and externalizes the data at the appropriate time (when the buffers are full or a checkpoint or commit is issued); therefore, additional overhead in performance is insignificant for jobs that use the BLOCKEDIO parameter and run under AR/CTL.

Run-time considerations

For PL/I application programs running under AR/CTL, PL/I must not intercept an abend and convert it to a return code. The abend must be allowed to occur; therefore, turn off the abend interception functions of PL/I (specify NOSPIE and NOSTAE run-time options). You can declare run-time options in the program with the following statement:

```plaintext
DCL PLIXOPT CHAR(255) VAR STATIC EXTERNAL INIT('NOSPIE,NOSTAE');
```

You can also specify run-time options with the PARM parameter of the EXEC statement.

The proper PL/I run-time libraries must be available to the job step.

The PL/I (PLISTART or CEESTART) CSECT must be at offset 0 of the load module. This is necessary so AR/CTL can correctly determine that application is a PL/I program.

Using AR/CTL-format checkpoint/restart calls

This section describes the how to code AR/CTL-format checkpoint/restart calls and structures in a PL/I application program. For more information, see “Using AR/CTL-format checkpoint/restart calls” on page 177.

Call format

The following example shows the AR/CTL checkpoint/restart call:

```plaintext
CALL ARCfunc (parmcount, parmlist...);
```

Call parameters

The following parameters are valid on the AR/CTL checkpoint/restart call:

**ARFunc**

Required. The name of the AR/CTL module to handle the call.
parmcount
Required. The address of a 4-byte field that contains the number of parameters in this call, not counting the parmcount parameter.

parmlist
Required. A list of parameters that the application program passes to AR/CTL with the call. The parameters to include in the list depend on the call.

Define the user areas to save in the checkpoint call as part of a structure. The beginning of each user area must be in the same structure as the end of that user area. The length field must be declared as a 31-byte fixed binary field, as shown in the following example:

```
DCL 1 WORKAREA
  3 ULENGTH FIXED BIN(31) INIT(500)
```

Restart ID area
The following is an example of the restart ID area code:

```
DCL 1 XRST_AREA,
  2 XRST_ID CHAR(8),
  2 XRST_PD CHAR(4);
```

Checkpoint ARB
The following is an example of the checkpoint ARB code:

```
DCL 1 ARCARB,
  2 ARCARB-TYPE CHAR(8) INIT('$ARCARB '),
  2 FUNC-CODE CHAR(8) INIT(' ',''),
  2 UNUSED CHAR(8) INIT(' ',''),
  2 STATUS-CODE CHAR(2) INIT(' ',''),
  2 UNUSED CHAR(2) INIT(' ',''),
  2 UNUSED CHAR(4);
```

Managing virtual storage for subprograms

AR/CTL provides the Subprogram Virtual Storage (ARCSPVS) application program interface (API) for managing defined areas of virtual storage for any application program, including a subprogram, in a restartable environment.

If an application subprogram already uses the subprogram support that is provided with the QUICKSTART for MVS product, no program changes are required for the use of AR/CTL support for subprograms. For more information, see “QUICKSTART-to-AR/CTL bridge” on page 32.
Implementing the ARCSPVS API

The following sections explain how to implement the ARCSPVS API in a PL/I application program.

For background information, see “Overview of the ARCSPVS API” on page 182.

Implemeting the ARCSPVS API

To implement the ARCSPVS API in an application program (including main programs and subprograms), perform the following steps:

1 Identify the virtual storage areas to be managed.

You should set up each virtual storage area as one aggregate variable (one DCL with multiple structures). This technique prevents problems if the PL/I optimizing compiler rearranges working storage. Label the beginning and the delimiter of the area. For more information, see “Identifying virtual storage areas to be managed” on page 224.

2 Create the $ARCSPVS parameter block.

For more information, see “Creating the $ARCSPVS parameter block” on page 225.

3 For each virtual storage area to be managed, use the Define function.

This step consists of initializing the $ARCSPVS parameter block and specifying DEFINE for the Function parameter (along with the name of the virtual storage area to be defined), issuing a call to ARCSPVS, and passing the initialized parameter block in the call.

If you prefer not to specify addresses explicitly, you can issue a call to ARCSPAD to obtain the addresses of the beginning and the delimiter of the virtual storage area. For more information, see “Using the Define function” on page 227.

4 If you have specified MANUAL for the Refresh Interval parameter, use the Refresh function to restore the virtual storage area manually.

This step consists of specifying REFRESH for the Function parameter in the $ARCSPVS parameter block (along with the name of the virtual storage area to be refreshed), issuing a call to ARCSPVS, and passing the initialized parameter block in the call. For more information, see “Using the Refresh function” on page 229.

5 At termination of the subprogram, before the subprogram returns to the main program, you can use the Delete function to remove the virtual storage area, or simply issue the call when the area is no longer needed.
Identifying virtual storage areas to be managed

This step consists of specifying DELETE for the Function parameter in the $ARCSPVS parameter block (along with the name of the virtual storage area to be deleted), issuing a call to ARCSPVS, and passing the initialized parameter block in the call. For more information, see “Using the Delete function” on page 230.

Identifying virtual storage areas to be managed

In the working storage section of the application program, identify each virtual storage area that you want AR/CTL to manage as shown in Figure 28.

Figure 28 Virtual storage areas in an application program

```c
/*                                 *********************************/
/*                                 *START OF VIRTUAL STORAGE AREA_1*/
/*                                 *********************************/
DCL 1 VIRTUAL_STORAGE_1             STATIC,
  5 VIR_STG_START_1             CHAR(21)        INIT('START VIRTUAL AREA_1:'),
/*                                 *********************************/
/*                                 * END OF VIRTUAL STORAGE AREA_1 */
/*                                 *********************************/
```

A virtual storage area is defined by the beginning address of the area and the delimiter of the area, which is the address of the end of the area and is not included in the area itself. Subtracting the beginning address from the delimiter provides the length of the area. Defined virtual storage areas may not overlap.

The definition of the virtual storage area persists until you explicitly delete the area (by using the Delete function of the ARCSPVS API) or the job step is cold-started. You cannot modify the characteristics of a virtual storage area after you define the area. The name of each virtual storage area must be unique within a job step while the area remains defined.
Creating the $ARCSPVS parameter block

At the end of the working storage section of the application program, create the $ARCSPVS parameter block as shown in Figure 29. For a sample PL/I parameter block that you can copy, see member ARCSPVSP of the AR/CTL sample library.

Figure 29  $ARCSPVS parameter block  (part 1 of 2)

```
DCL 01 ARCSPVS_PARAMETER_BLOCK,
                                                                                     
/* AR/CTL SUBPROGRAM VIRTUAL STORAGE PROCESSOR PARAMETER BLOCK. */                  
DCL 01 ARCSPVS_PARAMETER_BLOCK,
                                                                                     
/* PARAMETER BLOCK IDENTIFIER. VALID VALUE IS $ARCSPVS. */                          
03 SPVS_IDENTIFIER       CHAR (08) INIT('$ARCSPVS'),
                                                                                     
/* FUNCTION CODE. VALID VALUES ARE DEFINE, REFRESH, AND DELETE. */                
03 SPVS_FUNCTION         CHAR (16) INIT(' '),
                                                                                     
/* APPLICATION SUBPROGRAM VIRTUAL STORAGE NAME. MUST NOT BE BLANK AND NOT LOW_VALUES. */ 
03 SPVS_NAME             CHAR (32) INIT(' '),
                                                                                     
/* RETURN CODE. 0 = SUCCESSFUL RETURN 4 = REFRESHED */                             
03 SPVS_RETURN           FIXED BIN(31) INIT(0),
                                                                                     
/* BEGIN ADDRESS OF APPLICATION SUBPROGRAM VIRTUAL STORAGE AREA. */              
03 SPVS_BEGIN            POINTER,
                                                                                     
/* ADDRESS OF APPLICATION SUBPROGRAM VIRTUAL STORAGE AREA DELIMITER. */          
03 SPVS_DELIMITER        POINTER,
                                                                                     
/* APPLICATION SUBPROGRAM VIRTUAL STORAGE AREA REFRESH INTERVAL. VALID VALUES ARE DEFINE AND MANUAL. */ 
03 SPVS_REFRESH_INTERVAL  CHAR (16) INIT(' '),
```
Figure 29  $ARCSPVS parameter block (part 2 of 2)

```c
/* APPLICATION SUBPROGRAM VIRTUAL STORAGE AREA SAVE INTERVAL. VALID VALUE IS UOW. */
03 SPVS_SAVE_INTERVAL CHAR (16) INIT(' '),

/* RESERVED AREA. MUST BE LOW_VALUES. */
03 SPVS_RESERVED CHAR (412) INIT('0'B);
```

Table 16 describes the required fields in the $ARCSPVS parameter block.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Contents</th>
<th>Format and length</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>the identifier for the parameter block</td>
<td>CHAR(8)</td>
<td>$ARCSPVS</td>
</tr>
<tr>
<td></td>
<td>The ARCSPVS API requires the value to be $ARCSPVS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>the function to be performed in response to the call to the ARCSPVS API</td>
<td>CHAR(16)</td>
<td>DEFINE, REFRESH, DELETE</td>
</tr>
<tr>
<td>Name</td>
<td>the name of the virtual storage area to be managed</td>
<td>CHAR(32)</td>
<td>When the parameter block is passed to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the ARCSPVS API, the value cannot be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>blank or low values.</td>
</tr>
<tr>
<td>Return Code</td>
<td>the return code that the ARCSPVS API uses to communicate the results of the function</td>
<td>FIXED BIN(31)</td>
<td>ALL ZEROES</td>
</tr>
<tr>
<td></td>
<td>If the function is successful, the API sets the value to 0 before returning control to the</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>application program.</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>If the virtual storage area is refreshed, the API sets the value to 4 before returning</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>control to the application program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Begin</td>
<td>the address of the beginning of the virtual storage area to be managed</td>
<td>pointer</td>
<td>beginning address</td>
</tr>
<tr>
<td>Delimiter</td>
<td>the address of the delimiter of the virtual storage area to be managed (this delimiter is</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>not part of the virtual storage area)</td>
<td>pointer</td>
<td>delimiter address</td>
</tr>
<tr>
<td>Refresh Interval</td>
<td>the option that tells AR/CTL when to restore the virtual storage area to its state at the</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>last time that the area was saved</td>
<td>CHAR(16)</td>
<td>DEFINE, MANUAL</td>
</tr>
</tbody>
</table>
Using the Define function

For each virtual storage area to be managed, use the Define function of the ARCSPVS API by initializing the parameters in the $ARCSPVS parameter block with the values that you want to use. Then issue a call to ARCSPVS and pass the initialized parameter block with the call.

If you want to provide the addresses directly, you can use the Define function as shown in Figure 30.

Figure 30 Define function of the ARCSPVS API with direct addressing

```plaintext
/**********************************************/
/*              D E F I N E _ A R E A _ 1                     */
/**********************************************/
DEFINE_AREA_1:
PROC;
  WS_LABEL              = 'DEFINE_AREA_1';
  SPVS_IDENTIFIER       = '$ARCSPVS';
  SPVS_FUNCTION         = 'DEFINE';
  SPVS_NAME             = 'P1BUBBG_1';
  SPVS_RETURN           = 0;
  PARM_COUNT            = 2;
  VIR_STG_START_1       = ADDR(SPVS_BEGIN);
  VIR_STG_END_1         = ADDR(SPVS_DELIMITER);
  SPVS_REFRESH_INTERVAL = 'DEFINE';
  SPVS_SAVE_INTERVAL    = 'UOW';
  SPVS_RESERVED         = '  ';
```

Table 16 $ARCSPVS parameter block fields

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Contents</th>
<th>Format and length</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save Interval (UOW)</td>
<td>the parameter that tells AR/CTL when to save the contents of the virtual storage area to the checkpoint data set</td>
<td>CHAR(16)</td>
<td>UOW</td>
</tr>
<tr>
<td></td>
<td>Currently, AR/CTL always saves the virtual storage area at the unit of work (UOW) boundary, as indicated by the checkpoint call.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>the portion of the $ARCSPVS parameter block that is reserved for use by AR/CTL</td>
<td>CHAR(412)</td>
<td>LOW-VALUES</td>
</tr>
<tr>
<td></td>
<td>This parameter is required.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
If you prefer not to specify addresses explicitly, you can issue a call to ARCSPAD to obtain the addresses of the beginning and the delimiter of the virtual storage area.

For the Define function, set the parameter values as described in Table 17.

Table 17  $ARCSPVS parameter block fields for the Define function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Refresh function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>$ARCSPVS.</td>
</tr>
<tr>
<td>Function</td>
<td>DEFINE.</td>
</tr>
</tbody>
</table>
| Name            | the name of the virtual storage area to be managed  
                 | The name must not be blank or null, and must be unique within the job step until it is deleted or the job is cold-started. |
| Return Code     | ALL ZEROES             |
|                 | The application program can check the Return Code parameter after the ARCSPVS API returns control to the program. If the value is 4, the requested operation was successful and the virtual storage area was refreshed. |
| Begin           | beginning address that defines the location of the virtual storage area |
| Delimiter       | address that terminates the virtual storage area (and is not included in the area) |
| Refresh Interval| DEFINE if you want AR/CTL to restore the area during initialization when the program calls the ARCSPVS API with the DEFINE function code  
                 | MANUAL if you want AR/CTL to restore the area when the program call the ARCSPVS API with the REFRESH function code |
| Save Interval   | UOW                    |
| Reserved        | LOW-VALUES             |
|                 | This area is required and might be used by AR/CTL. |
Using the Refresh function

If you set the REFRESH value for the Refresh Interval parameter of the Define function, use the Refresh function as shown in Figure 31. This function manually refreshes the virtual storage area to its state at the last time that it was saved.

![Figure 31 Refresh function of the ARCSPVS API](image)

For the Refresh function, set the parameter values as described in Table 18.

**Table 18 $ARCSPVS parameter block fields for the Refresh function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Refresh function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>$ARCSPVS</td>
</tr>
<tr>
<td>Function</td>
<td>REFRESH</td>
</tr>
<tr>
<td>Name</td>
<td>must identify a previously defined virtual storage area</td>
</tr>
<tr>
<td>Return Code</td>
<td>ALL ZEROES</td>
</tr>
<tr>
<td></td>
<td>The application program can check the Return Code parameter after the ARCSPVS API returns control to the program. If the value is 4, the requested operation was successful and the virtual storage area was refreshed.</td>
</tr>
<tr>
<td>Beginning Address</td>
<td>ignored</td>
</tr>
<tr>
<td>Delimiter</td>
<td>ignored</td>
</tr>
<tr>
<td>Refresh Interval</td>
<td>ignored</td>
</tr>
<tr>
<td>Save Interval</td>
<td>ignored</td>
</tr>
<tr>
<td>Reserved</td>
<td>LOW-VALUES</td>
</tr>
<tr>
<td></td>
<td>This area is required and might be used by AR/CTL.</td>
</tr>
</tbody>
</table>
Using the Delete function

When the application program no longer needs AR/CTL to manage the virtual storage area, you can use the Delete function as shown in Figure 32. This function removes the definition of the virtual storage area, and AR/CTL no longer manages the area.

Figure 32  Delete function of the ARCSPVS API

```
/***************************************************************/
/*                      D E L E T E _ A R E A _ 1              */
/***************************************************************/
DELETE_AREA_1:
    PROC;
      WS_LABEL             = 'DELETE_AREA_1';
      SPVS_IDENTIFIER      = '$ARCSPVS';
      SPVS_FUNCTION        = 'DELETE';
      SPVS_NAME             = 'PIUBBG_1';
      PARM_COUNT            = 1;
      CALL ARCSPVS
        (PARM_COUNT,
         ARCSPVS_PARAMETER_BLOCK);
DELETE_AREA_1_EXIT:
      END DELETE_AREA_1;
```

For the Delete function, set the parameter values as described in Table 19.

Table 19  $ARCSPVS parameter block fields for the Delete function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Refresh function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>$ARCSPVS</td>
</tr>
<tr>
<td>Function</td>
<td>DELETE</td>
</tr>
<tr>
<td>Name</td>
<td>must identify a previously defined virtual storage area</td>
</tr>
<tr>
<td>Return Code</td>
<td>ALL ZEROES</td>
</tr>
<tr>
<td>Beginning Address</td>
<td>ignored</td>
</tr>
<tr>
<td>Delimiter</td>
<td>ignored</td>
</tr>
<tr>
<td>Refresh Interval</td>
<td>ignored</td>
</tr>
<tr>
<td>Save Interval</td>
<td>ignored</td>
</tr>
<tr>
<td>Reserved</td>
<td>LOW-VALUES</td>
</tr>
</tbody>
</table>

The application program can check the Return Code parameter after the ARCSPVS API returns control to the program. If the value is not zero, the requested operation has failed.

This area is required and might be used by AR/CTL.
Using AR/CTL common calls

Figure 33 shows fragments of a PL/I application program that uses AR/CTL common calls. For more information, see “Using AR/CTL common calls” on page 191.

Figure 33   AR/CTL common call example PL/I program

```
DCL ARCCALL OPTIONS(ASM INTER);
DCL PARMCNT   FIXED BINARY (31) STATIC
              INIT(19);
DCL FUNCTION  CHAR (4);
DCL STATUSCD  CHAR (2)
              INIT ('  ');
DCL FORCEOPT  CHAR (3)
              INIT ('YES');
DCL IOAREALN  FIXED BINARY (31) STATIC
              INIT(12).
DCL IOAREA    CHAR (12)
              INIT ('            ');
DCL 1 AREA0BEG.
DCL 1 AREA0END.
DCL 1 AREA1BEG.
DCL 1 AREA1END.
DCL 1 AREA2BEG.
DCL 1 AREA2END.
DCL 1 AREA3BEG.
DCL 1 AREA3END.
DCL 1 AREA4BEG.
DCL 1 AREA4END.
DCL 1 AREA5BEG.
DCL 1 AREA5END.
DCL 1 AREA6BEG.
DCL 1 AREA6END.
FUNCTION = 'XRST';

CALL PLITARC (PARMCNT,FUNCTION,STATUSCD,FORCEOPT,IOAREALN,IOAREA,
              AREA0BEG,AREA0END,AREA1BEG,AREA1END,AREA2BEG,AREA2END,
              AREA3BEG,AREA3END,AREA4BEG,AREA4END,AREA5BEG,AREA5END,
              AREA6BEG,AREA6END);

IF STATUSCD = ''
THEN
  DO;
  END;
ELSE
  DO;
  END;
```
Using IMS-compatible calls

This section describes how to code IMS-compatible calls and structures in a PL/I application program. For more information, see “Using IMS-compatible checkpoint/restart calls” on page 195 and “Using ASAM services through GSAM replacement” on page 130.

Entry and return
The entry statement must appear as the first executable statement in the program. When AR/CTL passes control to the application program, it passes the addresses, in the form of pointers, to each of the PCBs. The entry statement must list the pointers to the PCBs—not the PCB names:

```pli
ANYNAME: PROCEDURE (CHKPPCB,ASAMPCB1,ASAMPCB2, ...)
  OPTIONS (MAIN):
  ...
  RETURN;
```

Checkpoint I/O area
The easiest way to define the checkpoint I/O area is as a fixed-length character string and then pass the name of the string to AR/CTL as the call parameter:

```pli
DECLARE  1 CHKAREA,
  2 CHKPID CHAR(8),
  2 CHKPPAD CHAR(4);
```

You can define the checkpoint I/O area as a major structure, a connected array, or an adjustable character string, and then pass the pointer variable that points to that definition. You can also define the checkpoint I/O area as a minor structure if you use a pointer to the minor structure as the parameter. Do not use a character string with the VARYING attribute. If you use substructures or elements of an array, use the DEFINED or BASED attribute.

ASAM I/O area
The easiest way to define the ASAM I/O area is as a fixed-length character string, and then pass the name of the string to AR/CTL as the call parameter:

```pli
DECLARE  1 ASAMAREA,
  2 ASAMFLD1 CHAR(4),
  2 ASAMFLD2 CHAR(16),
  2 ASAMFLD3 CHAR(32);
```
Checkpoint PCB mask
Define the checkpoint PCB mask as a level 1 declaration:

```plaintext
DECLARE CHKPPCB_PTR POINTER;
DECLARE 1 CHKPPCB_NAME BASED (CHKPPCB_PTR),
  2 CHKP_FILL1 CHAR(28),
  2 CHKP_STATUS CHAR(2),
  2 CHKP_FILL2 CHAR(28);
```

ASAM PCB mask
Define the ASAM PCB as a level 1 declaration:

```plaintext
DECLARE ASAMPCB_PTR POINTER;
DECLARE 1 ASAMPCB_NAME BASED (ASAMPCB_PTR),
  2 ASAMPCB_NAME CHAR(8),
  2 ASAM_FILL1 CHAR(2),
  2 ASAM_STATUS CHAR(2),
  2 PROCOPT CHAR(4),
  2 ASAM_reserve1 FIXED BIN(31,0),
  2 ASAM_FILL2 CHAR(8),
  2 LEN_FILDS FIXED BIN(31,0),
  2 ASAM_FILL3 FIXED BIN(31,0),
  2 RSA CHAR(8),
  2 UREC_LEN FIXED BIN(31,0);
```

Call function codes
Define call function codes with declarations:

```plaintext
DCL FUNC_XRST CHAR(4) INIT('XRST');
DCL FUNC_CHK P CHAR(4) INIT('CHKP');
DCL FUNC_GU CHAR(4) INIT('GU ');
DCL FUNC_GN CHAR(4) INIT('GN ');
DCL FUNC_ISRT CHAR(4) INIT('ISRT');
```

AR/CTL calls
The following shows how to code AR/CTL call:

```plaintext
CALL PLITDLI (parmcount, function, parmlist...);
```

Call Parameters
The following parameters are valid on the AR/CTL call:

**PLITDLI**
Required. The name of the AR/CTL module to call for support functions. The module names are identical to those used in IMS for compatibility purposes.
parmcount
Required. The address of a four-byte field that contains the number of parameters in this call, not counting this parameter. This field must be first on all PL/I calls.

function
Required. The address of the area that contains the four-byte function code identifying the type of call.

parmlist
Required. A list of parameters that the application program passes to AR/CTL with the call. AR/CTL calls use the same parameter lists required by IMS. The parameters to include in the list depend on the call function. If the call requires more than one parameter, code them in the order listed.

Define the user areas to save in the checkpoint request or CHKP call as part of a structure. The user area cannot be part of a major structure or a stand-alone value. The length field must be declared as a 31-byte fixed binary field, as shown in the following example:

```
DCL 1 WORKAREA,
    3 ULENGTH FIXED BIN(31) INIT(500);
```

---

**Using ASAM callable interface calls**

This section describes how to code ASAM callable interface calls and structures in a PL/I application program. For more information, see “Using ASAM services through GSAM replacement” on page 130.

**AR/CTL calls**

The following shows how to code the AR/CTL callable interface:

```
CALL ARFunc (parmcount, parmlist...);
```

**Call parameters**

The following parameters are valid on the AR/CTL callable interface calls.

**ARFunc**

Required. The name of the AR/CTL module to handle the call.

**parmcount**

Required. The address of a 4-byte field that contains the number of parameters in this call, not counting the parmcount parameter.
parmlist
Required. A list of parameters that the application program passes to AR/CTL with the call. The parameters to include in the list depend on the call.

Define the user areas to save in the checkpoint call as part of a structure. It cannot be part of a major structure or a stand-alone value. The length field must be declared as a 31-byte fixed binary field, as shown in the following example:

```asm
DCL 1 WORKAREA, 3 ULENGTH FIXED BIN(31) INIT(500)
```

**ASAM I/O area**
Code the ASAM I/O area according to the following example. The example area contains three character-type fields, which are 4 bytes, 16 bytes, and 32 bytes long, respectively.

```asm
DCL 1 ASAMAREA,
  2 ASAMFLD1 CHAR(4),
  2 ASAMFLD2 CHAR(16),
  2 ASAMFLD3 CHAR(32);
```

**ASAM ARB**
Code the ARB according to the following example:

```asm
DCL 1 ARCARB,
  2 ARCARB-TYPE CHAR(8) INIT('$ARCARB '),
  2 FUNC-CODE CHAR(8) INIT('READ '),
  2 DDNAME CHAR(8) INIT('ddname '),
  2 STATUS-CODE CHAR(2) INIT(' '),
  2 UNUSED CHAR(2) INIT(' '),
  2 ULRECL FIXED BIN(31);
```
This chapter tells how to change Assembler language application programs to use the explicit calls provided by the APPLICATION RESTART CONTROL (AR/CTL) products. This chapter contains the following information:

Overview ................................................................. 237
Types of AR/CTL calls .................................................. 238
Sample Assembler language programs ............................ 238
EXEC parameters ...................................................... 238
Assemble and link issues .............................................. 239
Using AR/CTL-format checkpoint/restart calls ................. 239
Managing virtual storage for subprograms ...................... 241
Implementing the ARCSPVS API .................................... 241
Identifying virtual storage areas to be managed ............... 242
Creating the $ARCSPVS parameter block ..................... 243
Using the Define function .......................................... 244
Using the Refresh function ......................................... 246
Using the Delete function .......................................... 247
Using AR/CTL common calls ...................................... 248
Using IMS-compatible calls ....................................... 249
Using ASAM callable interface calls ............................ 251

Overview

The easiest way to obtain AR/CTL services in an application program is with automated methods, which require no program changes. If you are not able to use the automated methods, you can change the application program to issue explicit requests for AR/CTL services. This section provides general information about making changes to application programs written in Assembler language.
Before using the information in this chapter, review the overview information about AR/CTL checkpoint/restart and data services in Chapter 3, “Using AR/CTL checkpoint/restart services.”

**Types of AR/CTL calls**

AR/CTL supports several formats for the calls that request AR/CTL checkpoint/restart services:

- AR/CTL-format
- AR/CTL common format
- IMS-compatible

AR/CTL supports several formats for the calls that request application sequential access method (ASAM) services:

- IMS generalized sequential access method (GSAM) replacement
- ASAM callable interface

**Sample Assembler language programs**

BMC Software provides executable samples of application programs that use AR/CTL calls. For a list of these samples, see Appendix B, “Locating samples.”

**EXEC parameters**

The following sections provide information that is related to EXEC statement parameters.

**Length of parameter list**

Normally, MVS allows a 100-byte PARM string for an EXEC PGM statement. However, ARC/CTL uses some of these bytes during processing. Therefore, 61 bytes is the maximum allowable length of the PARM string on the EXEC statement for an application program that uses AR/CTL.
Original operating system parameters

According to standard MVS linkage conventions, register 1 points to the original operating system parameter list. However, AR/CTL uses this register for other purposes and provides access to a copy of the original parameter list through MVS name token services.

Member ASMORIGP of the sample library contains model code (written in Assembler language) that you can adapt and include in an application program if the program needs access to a copy of the original parameter list. This sample shows how to fetch the operating system parameter list. The address of the list is located at offset +8 (byte 9) of the TOKEN field.

This sample also shows a quick way for the application program to detect whether AR/CTL is active. The sample sets the return code to zero if AR/CTL is active and to a nonzero value if AR/CTL is not active or a problem is detected.

Assemble and link issues

An application program can use any explicit AR/CTL call (AR/CTL-format checkpoint/restart calls, AR/CTL common calls, ASAM callable interface calls) with dynamic linking or with static linking.

Dynamic linking

If the program uses dynamic linking, the AR/CTL libraries are automatically included at execution time. No change is required for your current link-edit process.

Static linking

If the program includes the AR/CTL language interface module (ARCNAALI), you must include the AR/CTL execution library in the SYSLIB concatenation of the link-edit JCL. The order of the libraries in the concatenation is not important.

Using AR/CTL-format checkpoint/restart calls

This section describes the how to code AR/CTL-format checkpoint/restart calls and structures in an Assembler language program. For more information, see “Using AR/CTL-format checkpoint/restart calls” on page 177.
Using AR/CTL-format checkpoint/restart calls

AR/CTL calls
The following shows how to code the AR/CTL checkpoint/restart calls:

```
CALL ARCFunc,(parmlist...),VL
```

Call parameters
The following parameters are valid on the AR/CTL checkpoint/restart call:

**ARCFunc**
Required. The name of the AR/CTL module to handle the call.

**parmlist**
Required. A list of parameters that the application program passes to AR/CTL with the call. The parameters to include in the list depend on the call. Each parameter is a fullword-aligned, 4-byte address.

**VL**
Required. The literal that marks the end of the parameter list.

Restart ID area
The following shows how to code the restart ID area:

```
XRSTAREA DS OCL12
XRSTID  DS CL8
XRSTPAD DS CL4
```

Checkpoint ARB
The following shows how to code the checkpoint ARB:

```
ARCARB   DS OCL32   ARB RECORD
ATYPE    DC CL8'ARCARB'   ARB TYPE IS CONSTANT
AFUNCODE DS CL8' '   FUNCTION CODE
           DS CL8   RESERVED
ASAMSTAT DS CL2' '   STATUS CODE
           DS CL2   RESERVED
           DS F    RESERVED
```
Managing virtual storage for subprograms

AR/CTL provides the Subprogram Virtual Storage (ARCSPVS) application program interface (API) for managing defined areas of virtual storage for any application program, including a subprogram, in a restartable environment.

If an application subprogram already uses the subprogram support that is provided with the QUICKSTART for MVS product, no program changes are required for the use of AR/CTL support for subprograms. For more information, see “QUICKSTART-to-AR/CTL bridge” on page 32.

The following sections explain how to implement the ARCSPVS API in a PL/I application program. For background information, see “Overview of the ARCSPVS API” on page 182.

Implementing the ARCSPVS API

To implement the ARCSPVS API in an application program (including main programs and subprograms), perform the following steps:

1. Identify each virtual storage area to be managed.
   You must label the beginning and the delimiter of the area. For more information, see “Identifying virtual storage areas to be managed” on page 242.

2. In the working storage section of the application program, create the $ARCSPVS parameter block.
   For more information, see “Creating the $ARCSPVS parameter block” on page 243.

3. For each virtual storage area to be managed, use the Define function.
   This step consists of initializing the $ARCSPVS parameter block and specifying DEFINE for the Function parameter (along with the name of the virtual storage area to be defined), issuing a call to ARCSPVS, and passing the initialized parameter block in the call.
   If you prefer not to specify addresses explicitly, you can issue a call to ARCSPAD to obtain the addresses of the beginning and the delimiter of the virtual storage area. For more information, see “Using the Define function” on page 244.

4. If you have specified MANUAL for the Refresh Interval parameter, use the Refresh function to restore the virtual storage area manually.
This step consists of specifying REFRESH for the Function parameter in the $ARCSPVS parameter block (along with the name of the virtual storage area to be refreshed), issuing a call to ARCSPVS, and passing the initialized parameter block in the call. For more information, see “Using the Refresh function” on page 246.

At termination of the subprogram, before the subprogram returns to the main program, you can use the Delete function to remove the virtual storage area, or simply issue the call when the area is no longer needed.

This step consists of specifying DELETE for the Function parameter in the $ARCSPVS parameter block (along with the name of the virtual storage area to be deleted), issuing a call to ARCSPVS, and passing the initialized parameter block in the call. For more information, see “Using the Delete function” on page 247.

Identifying virtual storage areas to be managed

In the working storage section of the application program, identify each virtual storage area that you want AR/CTL to manage as shown in Figure 34.

Figure 34 Virtual storage areas in an application program

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPVT_VS_0</td>
<td>DC C'&lt;SUBPROGRAM VIRTUAL STORAGE 0&gt;' X'FF'</td>
</tr>
<tr>
<td>SPVT_VS_0_BEGIN</td>
<td>EQU SPVT_VS_0,1,C'C'</td>
</tr>
<tr>
<td>SPVT_VS_0_DELIMITER</td>
<td>EQU SPVT_VS_0+L'SPVT_VS_0,1,C'C'</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

A virtual storage area is defined by the beginning address of the area and the delimiter of the area, which is the address of the end of the area and is not included in the area itself. Subtracting the beginning address from the delimiter provides the length of the area. Defined virtual storage areas may not overlap.

The definition of the virtual storage area persists until you explicitly delete the area (by using the Delete function of the ARCSPVS API) or the job step is cold-started. You cannot modify the characteristics of a virtual storage area after you define the area. The name of each virtual storage area must be unique within a job step while the area remains defined.
Creating the $ARCSPVS parameter block

At the end of the working storage section of the application program, create the $ARCSPVS parameter block as shown in Figure 35. For a sample Assembler language DSECT, see member $ARCSPVA of the AR/CTL sample library.

Table 20 describes the required fields in the $ARCSPVS parameter block.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Contents</th>
<th>Format and length</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>the identifier for the parameter block</td>
<td>CL8</td>
<td>$ARCSPVS</td>
</tr>
<tr>
<td>Function</td>
<td>the function to be performed in response to the call to the ARCSPVS API</td>
<td>CL16</td>
<td>DEFINE REFRESH DELETE</td>
</tr>
</tbody>
</table>
Using the Define function

For each virtual storage area to be managed, use the Define function of the ARCPVS API by initializing the parameters in the $ARCSPVS parameter block with the values that you want to use. Then issue a call to ARCPVS and pass the initialized parameter block with the call. See Figure 36.

### Table 20  $ARCSPVS parameter block fields

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Contents</th>
<th>Format and length</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>the name of the virtual storage area to be managed</td>
<td>CL32</td>
<td>When the parameter block is passed to the ARCPVS API, the value cannot be blank or low values.</td>
</tr>
<tr>
<td>Return Code</td>
<td>the return code that the ARCPVS API uses to communicate the results of the function</td>
<td>F</td>
<td>ALL ZEROES 4</td>
</tr>
<tr>
<td></td>
<td>If the function is successful, the API sets the value to 0 before returning control to the application program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If the virtual storage area is refreshed, the API sets the value to 4 before returning control to the application program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Begin</td>
<td>the address of the beginning of the virtual storage area to be managed</td>
<td>F</td>
<td>beginning address</td>
</tr>
<tr>
<td>Delimiter</td>
<td>the address of the delimiter of the virtual storage area to be managed</td>
<td>F</td>
<td>delimiter address</td>
</tr>
<tr>
<td>Refresh Interval</td>
<td>the option that tells AR/CTL when to restore the virtual storage area to its state at the last time that the area was saved</td>
<td>CL16)</td>
<td>DEFINE MANUAL</td>
</tr>
<tr>
<td>Save Interval (UOW)</td>
<td>the parameter that tells AR/CTL when to save the contents of the virtual storage area to the checkpoint data set Currently, AR/CTL always saves the virtual storage area at the unit of work (UOW) boundary, as indicated by the checkpoint call.</td>
<td>CL16</td>
<td>UOW</td>
</tr>
<tr>
<td>Reserved</td>
<td>the portion of the $ARCSPVS parameter block that is reserved for use by AR/CTL  This parameter is required.</td>
<td>412XLI</td>
<td>LOW-VALUES</td>
</tr>
</tbody>
</table>
If you prefer not to specify addresses explicitly, you can issue a call to ARCSPAD to obtain the addresses of the beginning and the delimiter of the virtual storage area.

Figure 36 Define function of the ARCSPVS API

| MVC    | SPVS_ID.=CL8'ARCSPVS'      | * MOVE $ARCSPVS IDENTIFIER |
| MVC    | SPVS_FUNCTION.=CL16'DEFINE' | * MOVE FUNCTION CODE       |
| MVC    | SPVS_NAME.=CL32'ARCSPVS-0'  | * MOVE SUBPROGRAM VIRTUAL  |
|        | X                      | ....STORAGE NAME           |
| XR     | R0,R0                   | * ZERO REGISTER            |
| ST     | R0,SPVS_RETURN          | * ZERO RETURN CODE         |
| LA     | R0,SPVT_VS_0_BEGIN      | * LOAD A-BEGIN SUBPROGRAM  |
|        |                        | X ....VIRTUAL STORAGE      |
| ST     | R0,SPVS@BEGIN           | * STORE A-BEGIN SUBPROGRAM |
|        |                        | X ....VIRTUAL STORAGE      |
| LA     | R0,SPVT_VS_0_DELIMITER  | * LOAD A-DLIMITER SUBPROGRAM|
|        |                        | X ....VIRTUAL STORAGE      |
| ST     | R0,SPVS@DELIMITER       | * STORE A-DLIMITER SUBPROGRAM|
|        |                        | X ....VIRTUAL STORAGE      |
| MVC    | SPVS_REFRESH_INTERVAL.=CL16'DEFINE' | * MOVE REFRESH INTERVAL   |
| MVC    | SPVS_SAVE_INTERVAL.=CL16'UOW'  | * MOVE SAVE INTERVAL      |

* DEFINE SUBPROGRAM VIRTUAL STORAGE

| LA     | R0,$ARCSPVS             | * LOAD A($ARCSPVS)         |
| ST     | R0,SPVT@$ARCSPVS        | * STORE A($ARCSPVS)        |
| OI     | SPVT@$ARCSPVS,X'80'     | * TERMINATE ARCSPVS PARAMETER |
|        |                        | X ....LIST                 |

* LA     | R1,SPVT_ARCSPVS_PL      | * LOAD A(ARCSPVS PARAMETER LIST) |
| L      | R15,SPVT@ARCSPVS        | * LOAD A(ARCSPVS)           |
| BASR   | R14,R15                 | * CALL ARCSPVS              |

For the Define function, set the parameter values as described in Table 21.

Table 21 SARCSPVS parameter block fields for the Define function (part 1 of 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Refresh function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>$ARCSPVS.</td>
</tr>
<tr>
<td>Function</td>
<td>DEFINE.</td>
</tr>
<tr>
<td>Name</td>
<td>the name of the virtual storage area to be managed</td>
</tr>
<tr>
<td></td>
<td>The name must not be blank or null, and must be unique within the job step until it is</td>
</tr>
<tr>
<td></td>
<td>deleted or the job is cold-started.</td>
</tr>
</tbody>
</table>
Using the Refresh function

If you set the REFRESH value for the Refresh Interval parameter of the Define function, use the Refresh function as shown in Figure 37. This function manually refreshes the virtual storage area to its state at the last time that it was saved.

**Figure 37  Refresh function of the ARCSPVS API**

| MVC   | SPVS_ID,=CL8'$ARCSPVS'       * MOVE $ARCSPVS IDENTIFIER |
| MVC   | SPVS_FUNCTION,=CL16'REFRESH' * MOVE FUNCTION CODE       |
| MVC   | SPVS_NAME,=CL32'ARCSPVSA-1'  * MOVE SUBPROGRAM VIRTUAL X |
|       | * ....STORAGE NAME            |
|       | * REFRESH SUBPROGRAM VIRTUAL STORAGE |
|       | * LA R0,$ARCSPVS              * LOAD A($ARCSPVS)          |
|       | ST R0,SPVT@$ARCSPVS          * STORE A($ARCSPVS)          |
|       | QI SPVT@$ARCSPVS,X'80'       * TERMINATE ARCSPVS PARAMETER X |
|       | * ....LIST                    |
|       | * LA R1,SPVT_ARCSPVS_PL      * LOAD A(ARCSPVS PARAMETER LIST) |
|       | L  R15,SPVT@ARCSPVS          * LOAD A(ARCSPVS)          |
|       | BASR R14,R15                 * CALL ARCSPVS              |

---

**Table 21  $ARCSPVS parameter block fields for the Define function (part 2 of 2)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Refresh function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Code</td>
<td>ALL ZEROES</td>
</tr>
<tr>
<td></td>
<td>The application program can check the Return Code parameter after</td>
</tr>
<tr>
<td></td>
<td>the ARCSPVS API returns control to the program. If the value is</td>
</tr>
<tr>
<td></td>
<td>4, the requested operation was successful and the virtual</td>
</tr>
<tr>
<td></td>
<td>storage area was refreshed.</td>
</tr>
<tr>
<td>Begin</td>
<td>beginning address that defines the location of the virtual</td>
</tr>
<tr>
<td></td>
<td>storage area</td>
</tr>
<tr>
<td>Delimiter</td>
<td>address that terminates the virtual storage area (and is not</td>
</tr>
<tr>
<td></td>
<td>included in the area)</td>
</tr>
<tr>
<td>Refresh Interval</td>
<td>DEFINE if you want AR/CTL to restore the area during</td>
</tr>
<tr>
<td></td>
<td>initialization when the program calls the ARCSPVS API with the</td>
</tr>
<tr>
<td></td>
<td>DEFINE function code</td>
</tr>
<tr>
<td></td>
<td>MANUAL if you want AR/CTL to restore the area when the</td>
</tr>
<tr>
<td></td>
<td>program call the ARCSPVS API with the REFRESH function code</td>
</tr>
<tr>
<td>Save Interval</td>
<td>UOW</td>
</tr>
<tr>
<td>Reserved</td>
<td>LOW-VALUES</td>
</tr>
<tr>
<td></td>
<td>This area is required and might be used by AR/CTL.</td>
</tr>
</tbody>
</table>
Using the Delete function

For the Refresh function, set the parameter values as described in Table 22.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Refresh function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>$ARCSPVS</td>
</tr>
<tr>
<td>Function</td>
<td>REFRESH</td>
</tr>
<tr>
<td>Name</td>
<td>must identify a previously defined virtual storage area</td>
</tr>
<tr>
<td>Return Code</td>
<td>ALL ZEROES</td>
</tr>
<tr>
<td>Beginning Address</td>
<td>ignored</td>
</tr>
<tr>
<td>Delimiter</td>
<td>ignored</td>
</tr>
<tr>
<td>Refresh Interval</td>
<td>ignored</td>
</tr>
<tr>
<td>Save Interval</td>
<td>ignored</td>
</tr>
<tr>
<td>Reserved</td>
<td>LOW-VALUES</td>
</tr>
</tbody>
</table>

The application program can check the Return Code parameter after the ARCSPVS API returns control to the program. If the value is 4, the requested operation was successful and the virtual storage area was refreshed.

Beginning Address ignored
Delimiter ignored
Refresh Interval ignored
Save Interval ignored
Reserved LOW-VALUES
This area is required and might be used by AR/CTL.

Using the Delete function

When the application program no longer needs AR/CTL to manage the virtual storage area, you can use the Delete function as shown in Figure 38. This function removes the definition of the virtual storage area, and AR/CTL no longer manages the area.

Figure 38  Delete function of the ARCSPVS API

```
MVC   SPVS_ID,=CL8'$ARCSPVS'    * MOVE $ARCSPVS IDENTIFIER
MVC   SPVS_FUNCTION,=CL16'DELETE'  * MOVE FUNCTION CODE
MVC   SPVS_NAME,=CL32'ARCSPVSA-0'  * MOVE SUBPROGRAM VIRTUAL X
   * ....STORAGE NAME
   * DELETE SUBPROGRAM VIRTUAL STORAGE
   *
   LA    R0,$ARCSPVS             * LOAD A($ARCSPVS)
   ST    R0,$SPVT@$ARCSPVS      * STORE A($ARCSPVS)
   OI    SPVT@$ARCSPVS,X'80'    * TERMINATE ARCSPVS PARAMETER X
   * ....LIST
   *
   LA    R1,SPVT_ARCSPVS_PL     * LOAD A(ARCSPVS PARAMETER LIST)
   L     R15,SPVT@ARCSPVS       * LOAD A(ARCSPVS)
   BASR  R14,R15                * CALL ARCSPVS
```
For the Delete function, set the parameter values as described in Table 23.

**Table 23** $ARCSPVS parameter block fields for the Delete function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Refresh function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>$ARCSPVS</td>
</tr>
<tr>
<td>Function</td>
<td>DELETE</td>
</tr>
<tr>
<td>Name</td>
<td>must identify a previously defined virtual storage area</td>
</tr>
<tr>
<td>Return Code</td>
<td>ALL ZEROES</td>
</tr>
<tr>
<td></td>
<td>The application program can check the Return Code parameter after the ARCSPVS API returns control to the program. If the value is not zero, the requested operation has failed.</td>
</tr>
<tr>
<td>Beginning Address</td>
<td>ignored</td>
</tr>
<tr>
<td>Delimiter</td>
<td>ignored</td>
</tr>
<tr>
<td>Refresh Interval</td>
<td>ignored</td>
</tr>
<tr>
<td>Save Interval</td>
<td>ignored</td>
</tr>
<tr>
<td>Reserved</td>
<td>LOW-VALUES</td>
</tr>
<tr>
<td></td>
<td>This area is required and might be used by AR/CTL.</td>
</tr>
</tbody>
</table>

**Using AR/CTL common calls**

Figure 39 shows fragments of an Assembler language application program that uses AR/CTL common calls. For more information, see “Using AR/CTL common calls” on page 191.

**Figure 39** AR/CTL common call example Assembler language program

```
CALL ASMTARC,(FUNCTION,STATUSCD,FORCEOPT,IOAREALN, X
  IOAREA,AREA0BEG,AREA0END,AREA1BEG,AREA1END, X
  AREA2BEG,AREA2END,AREA3BEG,AREA3END, X
  AREA4BEG,AREA4END,AREA5BEG,AREA5END, X
  AREA6BEG,AREA6END),VL
CLC    STATCD(L'STATCD),=2XL1'40'
BNE    XRST_ERR
FUNCTION DC CL4'XRST'                FUNCTION CODE.
STATUSCD DC XL2'4040'                STATUS CODE.
FORCEOPT DC CL3'NO '                 CHECKPOINT FORCE OPTION.
IOAREALN DC F'12'                    INPUT/OUTPUT AREA LENGTH.
IOAREA DC XL12'404040404040404040404040' INPUT/OUTPUT AREA.
AREA0BEG DS 0X                        FIRST USER AREA BEGIN.
AREA0END DS 0X                        FIRST USER AREA END + 1.
AREA1BEG DS 0X                        SECOND USER AREA BEGIN.
AREA1END DS 0X                        SECOND USER AREA END + 1.
AREA2BEG DS 0X                        THIRD USER AREA BEGIN.
```
Using IMS-compatible calls

This section describes how to code IMS-compatible calls and structures in an Assembler language application program. For more information, see “Using IMS-compatible checkpoint/restart calls” on page 195 and “Using ASAM services through GSAM replacement” on page 130.

Entry and return

Use any name for the entry statement. When AR/CTL passes control to the application program, register 1 contains the address of a variable-length fullword parameter list. Each fullword in the list contains the address of a PCB. The last fullword in the list contains X'80' in the high-order byte. Save the contents of register 1 before overwriting them. Use standard MVS linkage conventions with forward and backward chaining.

Checkpoint I/O area

Define the checkpoint I/O area with the DS statement:

| AREA2END DS 0X         | THIRD USER AREA END + 1.   |
| AREA3BEG DS 0X         | FOURTH USER AREA BEGIN.    |
| AREA3END DS 0X         | FOURTH USER AREA END + 1.  |
| AREA4BEG DS 0X         | FIFTH USER AREA BEGIN.     |
| AREA4END DS 0X         | FIFTH USER AREA END + 1.   |
| AREA5BEG DS 0X         | SIXTH USER AREA BEGIN.     |
| AREA5END DS 0X         | SIXTH USER AREA END + 1.   |
| AREA6BEG DS 0X         | SEVENTH USER AREA BEGIN.   |
| AREA6END DS 0X         | SEVENTH USER AREA END + 1. |

ASAM I/O area

Define the ASAM I/O area with the DS statement:

| ASAMAREA DS OCL52    |
| ASAMFLD1 DS CL4      |
| ASAMFLD2 DS CL16     |
| ASAMFLD3 DS CL32     |
Checkpoint PCB mask
Define only one checkpoint PCB mask per DSECT. Save the address of each PCB or the pointer to the list of PCB addresses. Then load the PCB address register with the PCB address before issuing the AR/CTL call.

<table>
<thead>
<tr>
<th>CHKPPCB</th>
<th>DSECT</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DS</td>
<td>CL10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CHKPSTAT DS CL2 CHECKPOINT STATUS CODE
CHKPSTAT DS CL28 RESERVED

ASAM PCB mask
Save either the address of each PCB or the pointer to the list of PCB addresses. Then load the PCB address register with the PCB address before issuing the AR/CTL call.

<table>
<thead>
<tr>
<th>ASAMPMP</th>
<th>DSECT</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ASAMNAME</td>
<td>DS CL8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASAMSTAT</td>
<td>DS CL2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASAMNAME</td>
<td>DS CL8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASAMSTAT</td>
<td>DS CL28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APROCOPT</td>
<td>DS CL28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALENFLDS</td>
<td>DS CL28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASAMRSA</td>
<td>DS CL8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AURECLEN</td>
<td>DS CL8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Call function codes
Define call function codes as constants with DC statement:

<table>
<thead>
<tr>
<th>XRST</th>
<th>DC CL4'XRST'</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHKP</td>
<td>DC CL4'CHKP'</td>
</tr>
<tr>
<td>GU</td>
<td>DC CL4'GU'</td>
</tr>
<tr>
<td>GN</td>
<td>DC CL4'GN'</td>
</tr>
<tr>
<td>ISRT</td>
<td>DC CL4'ISRT'</td>
</tr>
</tbody>
</table>

AR/CTL calls
The following shows how to code the AR/CTL calls:

CALL ASMTDLI,(function,parmlist...),VL

Call parameters
The following parameters are valid on the AR/CTL call:

ASMTDLI
Required. The name of the AR/CTL module to call for support functions. The module names are identical to those used in IMS for compatibility purposes.
function
  Required. The address of the area that contains the four-byte function code that
  identifies the type of call.

parmlist
  Required. A list of parameters that the application program passes to AR/CTL
  with the call. AR/CTL calls use the same parameter lists required by IMS. The
  parameters to include in the list depend on the call function. If the call requires
  more than one parameter, code them in the order listed.

VL
  Required and valid for Assembler language only. Literal that marks the end of the
  parameter list.

Using ASAM callable interface calls

This section describes how to code ASAM callable interface calls and structures in an
Assembler language application program. For more information, see “Using ASAM
services through the ASAM callable interface” on page 132.

AR/CTL calls

The following shows how to code the AR/CL callable interface calls:

CALL ARCfunc,(parmlist...),VL

Call parameters

The following parameters are valid on the AR/CTL callable interface call:

ARCfunc
  Required. The name of the AR/CTL module to handle the call.

parmlist
  Required. A list of parameters that the application program passes to AR/CTL
  with the call. The parameters to include in the list depend on the call. Each
  parameter is a fullword-aligned, 4-byte address.

VL
  Required. Literal that marks the end of the parameter list.

ASAM I/O area

Code the ASAM I/O area according to the following example. The example area
contains three character type fields (4 bytes, 16 bytes, and 32 bytes, respectively):

<table>
<thead>
<tr>
<th>ASAMAREA</th>
<th>DS</th>
<th>OCL52</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASAMFLD1</td>
<td>DS</td>
<td>CL4</td>
</tr>
</tbody>
</table>
**ASAM ARB**

Code the ARB according to the following example:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCARB</td>
<td>ARB RECORD</td>
<td>DS OCL32</td>
</tr>
<tr>
<td>ATYPE</td>
<td>ARB TYPE IS CONSTANT</td>
<td>DC CL8'ARCARB'</td>
</tr>
<tr>
<td>AFUNCODE</td>
<td>FUNCTION CODE</td>
<td>DS CL8'READ'</td>
</tr>
<tr>
<td>ADDNAME</td>
<td>FCB DDNAME</td>
<td>DS CL8'</td>
</tr>
<tr>
<td>ASAMSTAT</td>
<td>STATUS CODE</td>
<td>DS CL2'</td>
</tr>
<tr>
<td>AURECLEN</td>
<td>UNDEFINED RECORD LENGTH</td>
<td>DS CL2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>
Solving problems

This appendix describes how to handle non-standard situations when APPLICATION RESTART CONTROL (AR/CTL) is active in an environment. This appendix contains the following information:

Overview .................................................. 254
Handling application program situations .................................. 255
  Restarting a non-unique job step ................................... 256
  Cold-starting a job step after an abend ................................ 256
  Restarting at other than the selected checkpoint .................... 257
Handling checkpoint/restart data set problems .......................... 258
  Correcting a checkpoint data set out-of-space condition .......... 259
  Handling a destroyed or damaged checkpoint/restart data set .... 260
Handling ASAM data set situations ....................................... 261
  Correcting an ASAM data set out-of-space condition ............ 261
  Handling a destroyed or damaged ASAM data set ................... 262
Handling system problems .............................................. 262
  Recovering from a system failure .................................. 262
  Handling a cold start of IMS ...................................... 263
  Recovering from a CICS system failure ............................. 263
  Recovering from a disaster ....................................... 263
  Handling a checkpoint-in-doubt condition ......................... 264
Contacting BMC Software Product Support .............................. 265
  Preparing information for Product Support ....................... 265
  Creating a diagnostic tape .................................... 266
  Obtaining traces ........................................... 267
  Obtaining dumps .......................................... 268
Overview

This section provides an overview of non-standard situations you may experience when you operate with AR/CTL. Standard operations for AR/CTL are highly automated; however, certain non-standard situations can require manual intervention. These situations include abnormal termination of non-unique job steps and damage or destruction of essential AR/CTL components.

Table 24 lists some causes of non-standard operational situations and the page numbers of the procedures for handling them.

NOTE

Problems with the BMC Software subsystems and the REGISET are discussed in the APPLICATION RESTART CONTROL Administrator Guide.

Table 24 Non-standard situations

<table>
<thead>
<tr>
<th>Cause</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>application sequential access method (ASAM) data set destroyed or damaged</td>
<td>page 262</td>
</tr>
<tr>
<td>ASAM data set out of space</td>
<td>page 261</td>
</tr>
<tr>
<td>checkpoint in doubt</td>
<td>page 264</td>
</tr>
<tr>
<td>checkpoint/restart data set out of space</td>
<td>page 259</td>
</tr>
<tr>
<td>checkpoint/restart data set destroyed or damaged</td>
<td>page 260</td>
</tr>
<tr>
<td>CICS system failure</td>
<td>page 263</td>
</tr>
<tr>
<td>cold start of an application program job step</td>
<td>page 256</td>
</tr>
<tr>
<td>IMS cold start</td>
<td>page 263</td>
</tr>
<tr>
<td>non-unique job step restart</td>
<td>page 256</td>
</tr>
<tr>
<td>site disaster</td>
<td>page 263</td>
</tr>
<tr>
<td>system failure</td>
<td>page 262</td>
</tr>
</tbody>
</table>

A source of information for problem-solving is the error messages and codes that AR/CTL issues in problem situations. These messages are documented in the BMC Documentation Center and are provided in the Message Viewing utility (option 12 on the Application Enhancement Series primary menu).

If you need help, contact BMC Software Product Support. You may need to provide information about the environment, the application program, and the AR/CTL components in use when the problem occurred. For more information see “Contacting BMC Software Product Support” on page 265.
Handling application program situations

This section describes how to handle non-standard application program situations. These situations include restart of a job step defined to AR/CTL as non-unique, cold start of an application program defined as needing restart by AR/CTL, and restart at other than the checkpoint AR/CTL has selected. AR/CTL automatically handles application program restart in standard situations, which include any application program error that does not require database recovery and a cold start of the application program.

While you correct the problem that caused the abend, determine whether the job step can be restarted or must be cold-started:

- You can restart a job step if all processing before the latest checkpoint is correct.

- You may need to recover the databases and/or sequential data sets and then perform a cold start of a job step if the database updates are incorrect or if the application program updated the database but never issued a checkpoint. Cold-starting a job step without database recovery is valid if the application program made no database updates or the database updates can be repeated without database corruption.

**NOTE**

If the application program abends without issuing a restart call and/or a checkpoint call, AR/CTL cannot restart the job step. If the application program issues database update calls before issuing the restart and/or checkpoint call, and the job step abends before the call, AR/CTL cannot restart the job step. The job step must be handled as a cold start, or the restart must be handled outside of AR/CTL. You can use the Check XRST, Check CHKP, and Check IMS Updates options to prevent job steps from executing without issuing restart and/or checkpoint calls correctly. For more information, see the APPLICATION RESTART CONTROL Reference Manual.
Restarting a non-unique job step

If a job step that is defined as unique abends and requires restart, the usual procedure is to correct the problem that caused the abend and submit the job step for restart. No application program or JCL changes are required. However, if a job step that is defined as non-unique abends and requires restart, you must perform the following steps:

1. Correct the problem that caused the abend.

2. Obtain the restart data set name from the restart control record for the abended job step. The APPLICATION RESTART CONTROL Reference Manual describes how to access this record with action code S. The JES job number may help you identify a non-unique job step; you can display this job number by selecting the restart control record.

3. Include the ARCXRST DD statement, defining the data set name you obtained in the previous step, in the restart JCL.

4. Submit the job step for restart.

**NOTE**

If the job step is erroneously defined as non-unique, perform steps 1 through 4, wait for the restart to complete successfully, and then change the value of the Non-Unique Job Step option. For more information, see the APPLICATION RESTART CONTROL Reference Manual.

Cold-starting a job step after an abend

If a job step abends and requires a cold start (begin over at the beginning), you must perform the following steps before you can submit the job step for cold start. Depending on your environment, other issues may require action.

1. Correct the problem that caused the abend.

2. Ensure that all databases and tables that were updated by the application program are recovered or backed out to the application program start time.

3. Ensure that all VSAM data sets that the application program updated through AR/CTL local or remote VSAM access are recovered or backed out to the application program start time. You can use RECOVERY PLUS for CICS/VSAM (RPCV™) to perform a time stamp recovery from the CICS logs.
4 Handle any ASAM data sets to prepare them for cold start. If the application program allocated them as NEW or MOD, delete them. If the application program allocated them as OLD or SHR, no action is necessary.

5 Delete the restart control record for the job step, and delete the associated restart data set (if it exists). The APPLICATION RESTART CONTROL Reference Manual describes how to access this record with action code D. You can use the COLD keyword to restart the job step if it is using IMS or IMS-compatible structures and calls, as explained in “Using the IMS CHKPID parameter” on page 96.

---

**NOTE**
If you always want to cold-start the job step, use the Always Cold Start option. Under this option, AR/CTL automatically deletes the restart control record and restart data set so that the job step is not restarted. For more information, see the APPLICATION RESTART CONTROL Reference Manual.

---

**Restarting at other than the selected checkpoint**

If you want to restart a job step at other than the checkpoint that AR/CTL has selected for restart, the actions you can take depend on whether an application program uses IMS.

---

**WARNING**
If you restart a job step at other than the checkpoint that AR/CTL has selected for restart, you may need to perform a manual backout to the other checkpoint before you attempt the restart. For example, if the job step processes VSAM data sets, during abnormal job step termination AR/CTL may have performed automatic backout for these data sets. This automatic backout is performed for changes that have been made since the last checkpoint—not the previous checkpoint. If you select the previous checkpoint for restart in this case, errors (such as duplicate keys or updating records that have already been updated) may result after the restart.

---

**Non-IMS program**

If the application program does not use IMS, you can restart the job step from the other checkpoint stored in the restart data set. Use the TOGGLE command on the Restart Data Set Information panel, as described in the APPLICATION RESTART CONTROL Reference Manual. If the other checkpoint is not the one you want to use, you must restart from the beginning after taking appropriate action, as described in “Cold-starting a job step after an abend” on page 256.
 IMS program

If the application program uses IMS, you can use any of the following methods:

- Use the `TOGGLE` command on the Restart Data Set Information panel, as described in the `APPLICATION RESTART CONTROL Reference Manual`.

- Use the Set Manual Restart Mode panel to provide the checkpoint ID and log data set name, as described in the `APPLICATION RESTART CONTROL Reference Manual`. The IMS logs must contain the checkpoint record you want to use for restart.

- In the restart JCL, code the restart checkpoint ID on the EXEC statement as the eighth positional parameter (for a DL/I batch program) or the ninth positional parameter (for a batch message processing or BMP program). Then include the IMSLOGR DD statement that describes the log data sets containing the restart checkpoint record.

- Code the restart checkpoint ID in the `XRST` call in the application program and include the IMSLOGR DD statement in the JCL.

Handling checkpoint/restart data set problems

AR/CTL uses a VSAM linear data set to store the restart and repositioning information for a job step. This data set is called the checkpoint data set when AR/CTL writes checkpoint records to it during job step execution. It is called the restart data set when AR/CTL reads records from it during job step restart. The checkpoint/restart data set is essential to AR/CTL operations. It is possible for this data set to experience an out-of-space condition or destruction through hardware failure or accidental deletion. This section describes these problems and procedures for solving them.

If AR/CTL is unable to allocate the checkpoint data set during normal/cold start, correct the allocation problem and submit the job step as it was originally submitted. Because application program processing does not occur until after AR/CTL allocates the checkpoint data set, job step status remains as it was at the first submission.

If AR/CTL cannot allocate the restart data set during restart, use the procedure described in “Handling a destroyed or damaged checkpoint/restart data set” on page 260.
Correcting a checkpoint data set out-of-space condition

AR/CTL allocates the initial size for the checkpoint data set with the primary allocation quantity you specify plus fifteen times the secondary allocation quantity specified in the ARCCHPK dynamic allocation record. BMC Software recommends that you specify all space as the primary allocation quantity. The APPLICATION RESTART CONTROL Reference Manual provides a formula for estimating the size requirements of the checkpoint data set.

If the checkpoint data set runs out of space, AR/CTL cannot write the current checkpoint record to the data set. This situation seldom occurs because the data set contains only the two most recent checkpoints and AR/CTL reuses the space occupied by previous checkpoints; therefore, the number of checkpoints issued during the job step is not an issue.

A full data set is possible in the following situations:

- Allocation quantities are too small in proportion to the size of the checkpoint records.

- The application program has specified one or more checkpoint areas in the call incorrectly, causing the size of the checkpoint records to continue to increase during job step execution.

To correct the problem, perform the following steps:

1. If an application program problem caused the full data set, correct the problem within the application program.

2. Increase the size of the checkpoint data set:

   - If the data set name of the checkpoint data set contains a qualifier that ensures each data set name is unique, you can increase the primary and secondary allocation quantities for the data set in the ARCCHPK dynamic allocation record. If the volume is specified in the skeleton, make sure that the volume has enough free space. AR/CTL allocates the new checkpoint data set during AR/CTL initialization.

   - If the data set name of the checkpoint data set is the same for each execution of the job step, you can use the access method services REPRO command to reproduce the existing checkpoint data set as a larger data set on a volume that has enough space. Using the same data set name for all executions is not recommended.

3. Submit the job step for restart.
Handling a destroyed or damaged checkpoint/restart data set

The checkpoint/restart data set can be destroyed (possibly from a hardware problem or from being deleted) or damaged (possibly by an I/O error). The procedure to use depends on whether an application program uses IMS and, if it does use IMS, whether the checkpoint records were written to the IMS log.

**NOTE**

If the REGISET and one or more checkpoint/restart data sets are destroyed or damaged, perform the procedure to recover the REGISET first, as described in the APPLICATION RESTART CONTROL Administrator Guide. Then return to this procedure to work with the restart data sets.

### Checkpoint records exist in the IMS environment

If IMS is present and the checkpoint (X’18’) records were written to the IMS log, you can perform a manual IMS restart by including the IMSLOGR DD statement in the JCL and entering the checkpoint ID on the `PARM` parameter of the EXEC statement in the JCL. Or you can perform the following steps:

1. List the restart control records as explained in the APPLICATION RESTART CONTROL Reference Manual.
2. Use action code `M` to select the record to modify.
3. Supply the log data set information:
   - If the log data set is cataloged, enter the data set name of the log.
   - If the log data set is not cataloged, or if you prefer not to provide it on the panel, provide the data set information with the IMSLOGR DD statement in the job step JCL.
4. Supply the checkpoint ID to use for restart. The panel reminds you to verify that the application program has been backed out to this checkpoint.
5. Submit the job step for restart.

### Checkpoint records do not exist or non-IMS environment

If the checkpoint records do not exist or the job step was not executing in an IMS environment, the only action is to cold start the job step. For more information see “Cold-starting a job step after an abend” on page 256.
Handling ASAM data set situations

ASAM data sets are sequential data sets that an application program accesses through the ASAM callable interface or through IMS generalized sequential access method (GSAM) calls. An ASAM data set can run out of space during processing or can be destroyed or damaged. This section discusses these situations and the procedures for handling them.

Correcting an ASAM data set out-of-space condition

An output ASAM data set can run out of space during application program processing under the following circumstances:

- Allocation quantities are too small for the number and size of records that will be written to the data set.
- The volume is almost full or is fragmented so that few (or no) areas of contiguous space are available for secondary extents.
- An application program logic error, such as a loop, caused the program to write extraneous data to the data set.

If a program logic error caused the full ASAM data set and the data in it to be invalid, recover any other databases that were updated during the job step; then, cold-start the job step, as explained in “Cold-starting a job step after an abend” on page 256.

If the allocation quantities are too small or the volume does not have enough space, perform the following steps:

1. Allocate a new ASAM data set with adequate quantities for the primary and secondary allocation quantities on a volume with adequate space.

2. Use the ASAM Copy utility (program name ARCYRC00) to copy the old output ASAM data set to the new data set.

   For more information, see “Copying an ASAM data set” on page 133.

   **WARNING**

   Do not use IEBGENER or any other utility to copy an ASAM data set. Other utilities can destroy the data set structure.

3. Delete the old data set.
4 Rename the new data set to the old data set name.

5 Submit the job step for restart.

Handling a destroyed or damaged ASAM data set

If a critical ASAM data set is destroyed or damaged during processing, recover the databases updated by the application program; then, cold-start the job step, as explained in “Cold-starting a job step after an abend” on page 256.

Handling system problems

AR/CTL handles most system failures automatically. This section discusses system problems and disaster recovery.

Recovering from a system failure

AR/CTL marks a job step that is executing as ACTIVE in the restart control record. While the status is ACTIVE, AR/CTL does not allow the job step to be restarted; this action prevents problems if a job step is accidentally resubmitted while it is still executing. If a system failure or database management system (DBMS) failure occurs during execution of a job step controlled by AR/CTL, AR/CTL might not receive control to mark the job step as ABENDED.

To allow AR/CTL to restart the job step, reset the status to ABENDED on the panel that lists restart control records. For more information, see the APPLICATION RESTART CONTROL Reference Manual.

To automate resetting the status at IPL, you can use the Restart Control Record Status Reset utility (program name AESURABN) as described in the APPLICATION RESTART CONTROL Reference Manual.

After the system is restored, the DBMS is restarted, and backout is performed (if necessary), submit the unique job steps for restart. No application program or JCL changes are necessary. For non-unique job steps, follow the procedure for a non-unique job step, as described in “Restarting a non-unique job step” on page 256.
Handling a cold start of IMS

AR/CTL allows automated restart of application programs across a cold start of an IMS system. After the system is restored, IMS is restarted, and backout is performed (if necessary), submit the unique job steps for restart. No application program or JCL changes are necessary. For non-unique job steps, follow the procedure described in “Restarting a non-unique job step” on page 256.

Recovering from a CICS system failure

Through the remote VSAM access services of AR/CTL, a batch application program can use a VSAM data set owned by a CICS region. If CICS fails while sharing a VSAM data set with a batch program, CICS considers the batch program to be an in-flight transaction. At emergency restart, CICS performs dynamic transaction backout for all changes made by the batch program since the last checkpoint (syncpoint) or since the beginning of the job step, whichever is later.

If emergency restart fails, you may be able to recover the data set with RPCV. If you run an RPCV UPDATE BACKOUT, be sure to include in-flight transactions. If you run an RPCV forward file recovery, be sure to exclude in-flight transactions.

WARNING

A data integrity exposure may result if you set the Access Level option to 2 in the data set option (FCB) member for a shared VSAM data set and a batch job updates that data set while it requires CICS recovery. The CICS recovery may overlay the updates made by the batch job. You can prevent this exposure by setting the Access Level option to 3.

Recovering from a disaster

It is beyond the scope of AR/CTL to restart application programs executing at the time of a disaster or when a system snapshot is taken for disaster recovery. If your site experiences a disaster, such as a fire or a storm, the procedure to perform depends on whether an off-site backup of the REGSET is available. For more information, see the APPLICATION RESTART CONTROL Administrator Guide.
Handling a checkpoint-in-doubt condition

One of the two checkpoints in the restart data set is always valid for restart. Under certain circumstances, AR/CTL may be writing the checkpoint to the restart data set when an abend or system failure occurs. At restart, if the checkpoint does not have a COMPLETED status, AR/CTL issues message BMC74301E to request manual intervention.

Before you can restart the job step, you must handle the checkpoint-in-doubt condition. The procedure depends on the DBMSs involved in application program execution. Part of the procedure is to issue the TOGGLE command on the Restart Data Set Information panel (which you access by selecting the restart control record with action code C). For more information about using the TOGGLE command, see the APPLICATION RESTART CONTROL Reference Manual.

DB2-only program

For a DB2-only program, perform the following steps:

1. Determine whether the DB2 commit operation has completed by issuing a DISPLAY THREAD (*) TYPE(INDOUBT) command.
   - If no in-doubt threads are found, the commit operation has completed. No action is necessary.
   - If in-doubt threads are found, the commit operation has not completed. For each in-doubt thread, issue a DB2 RECOVER INDOUBT(connection-name) ACTION (ABORT) ID( ) command.

2. In AR/CTL, issue the TOGGLE command as necessary to select the checkpoint with the checkpoint ID you obtained from the message. You may need to issue the command twice.

IMS-only and IMS/DB2 program

For an IMS-only program or an IMS/DB2 program, perform the following steps:

1. Obtain the checkpoint ID of the most recent IMS checkpoint. The latest DFS0540I message on the JES message log contains this checkpoint ID.

2. In AR/CTL, issue the TOGGLE command as necessary to select the checkpoint with the checkpoint ID you obtained from the message. You may need to issue the command twice.
NOTE
If the program is executing in an IMS batch DLI region, make sure that the IMS databases are backed out to the checkpoint identified in the last DFS0540I message. The BATCH CONTROL FACILITY (BCF) component of AR/CTL can automate this task.

BMP-only program

In addition to checking the latest DFS0540I message, check the DFS0682I message. Perform the following steps:

1. Obtain the checkpoint ID of the most recent checkpoint. The latest DFS0540I message on the JES message log may contain this checkpoint ID.

2. Also check the DFS0682I message, which is issued by the backout function and can be used to indicate the true status of an in-doubt checkpoint. The checkpoint ID that was involved in the backout is identified in this message.

NOTE
The DFS0682I message is applicable to a BMP only, and is not present in the SYSLOG of the individual BMP. Instead, it can be found in the JES message log of the IMS control region.

Contacting BMC Software Product Support

This section describes how to contact BMC Software for product support. A BMC Software technical support analyst for AR/CTL is always on call to answer questions and solve problems. You can contact a technical support analyst as described on the back of the title page of this manual.

Preparing information for Product Support

You can help the BMC Software technical support analyst find a faster resolution to your problem by having the following information available when you call:

- version, release, and maintenance level of the BMC Software Primary Subsystem (BMCP) and the BMC Consolidated Subsystem (BCSS)

- version, release, and maintenance level of AR/CTL

You can find this information on the Operating Environment Display panel, as described in the APPLICATION RESTART CONTROL Administrator Guide.
Creating a diagnostic tape

The BMC Software technical support analyst may ask you to send BMC Software documentation for use in problem diagnosis. The fastest way to get documentation to BMC Software is by way of email or FTP. Otherwise you may send a cartridge tape.

Table 25 shows the utilities you can use to copy the data to the tape.

### Table 25  Utilities for creating diagnostic tape

<table>
<thead>
<tr>
<th>Data</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>general partitioned data set</td>
<td>IEBCOPY or DFDSS</td>
</tr>
<tr>
<td>system or snap dump</td>
<td>IEBGENER or DFDSS</td>
</tr>
<tr>
<td>checkpoint/restart data set (VSAM linear data set)</td>
<td>IDCAMS REPRO function or DFDSS</td>
</tr>
<tr>
<td>REGISET</td>
<td>IDCAMS REPRO function or DFDSS</td>
</tr>
<tr>
<td>ASAM data set</td>
<td>ASAM Copy utility (provided with AR/CTL)</td>
</tr>
<tr>
<td>VSAM data set</td>
<td>IDCAMS REPRO function or DFDSS</td>
</tr>
</tbody>
</table>
Obtaining traces

Enhanced call tracing can be a valuable tool for solving application program problems. For more information, see Chapter 5, “Using AR/CTL operational services.”

The following DD statements may be useful if you include them in the application program job step JCL:

**ASMTRACE DD**

If a problem occurs during AR/CTL processing and an application program is using an ASAM data set or a GSAM data set that AR/CTL is replacing with ASAM), the BMC Software technical support analyst may ask you to include the ASMTRACE DD statement to define the ASAM trace data set. The following shows how the data set should be allocated:

```
//ASMTRACE DD DSN=data.set.name,DISP=(NEW,CATLG,CATLG),LRECL=4096,
// RECFM=FB,BLKSIZE=16384
```

**RMTTRACE DD**

If a problem occurs during AR/CTL processing and the application program is using a data set through remote VSAM access services, the BMC Software technical support analyst may ask you to include the RMTTRACE DD statement to define the remote VSAM access trace data set. The following shows how the data set should be allocated:

```
//RMTTRACE DD DSN=data.set.name,DISP=(NEW,CATLG,CATLG),LRECL=4096,
// RECFM=FB,BLKSIZE=16384
```

The block size can be any even multiple of 4096.
VSMTRACE DD

If a problem occurs during AR/CTL processing and the application program is using a data set through local VSAM access services, the BMC Software technical support analyst may ask you to include the VSMTRACE DD statement to define the local VSAM access trace data set. The following shows how the data set should be allocated:

```
//VSMTRACE DD DSN=data.set.name,DISP=(NEW,CATLG,CATLG),LRECL=4096,
   // RECFM=FB,BLKSIZE=16384
```

The block size can be any even multiple of 4096.

**Printing diagnostic trace data sets**

You can use the IDCAMS PRINT command to print ASMTRACE, RMTTRACE, and VSMTRACE data sets. The following example shows statements to print the ASMTRACE data set:

```
//PASMTRC EXEC PGM=IDCAMS,COND=EVEN
//SYSPRINT DD SYSOUT=*  
//ASMTRACE DD DSN=data.set.name,DISP=(OLD,KEEP,KEEP)
//SYSPRINT DD SYSOUT=*  
//ASMTRACE DD DSN=data.set.name,DISP=(OLD,KEEP,KEEP)
//SYSPRINT DD SYSOUT=*  
PRINT INFILE(ASMTRACE) DUMP
```

**Obtaining dumps**

Application program snap dumps can be a valuable tool for solving application program problems. For more information, see “Using on-demand snap dumps” on page 149.

The SYSUDUMP DD statement may be useful if you include it in the application program job step JCL. If you include the standard SYSUDUMP DD statement in the JCL and a problem occurs during the execution of the job step, AR/CTL writes diagnostic information to the SYSUDUMP data set. Normally the SYSUDUMP data set is defined as a SYSOUT=* data set.

The SYSUDUMP DD statement should be present for most abend conditions. If it was not present, contact BMC Software before rerunning the job. A BMC Software technical support analyst will determine whether a SYSUDUMP data set really is necessary.
Abbreviated dumps do not always provide the type of information needed for problem resolution. Please ensure that the dump is complete. If you have the Abend-AID product installed, you can use one of the following DD statements to produce a complete dump:

```
//ABNLIGNR DD DUMMY
//ABNLDUMP DD DUMMY
```

The first statement turns Abend-AID off for this job step and allows a SYSUDUMP to be produced. The second statement produces an Abend-AID dump, followed by a SYSUDUMP.

If a system abend, such as S0C4 occurs, Abend-AID may write two dumps. The first is an abbreviated dump, and the second is a normal system dump. Please send both to BMC Software for problem diagnosis.

## System dump options

BMC Software requires specific dump options to be in effect so that sufficient data is available in the dump output.

To display the dump options currently in effect, enter the following command:

```
D DUMP,OPTIONS
```

Dump options may be modified with the following Change Dump operator command, CHNGDUMP (CD)

### Recommended dump options

To display the dump options currently in effect, type the following:

```
D DUMP,OPTIONS
```

The following operator commands change dump options:

- **SYSUDUMP**

  ```
  CD,SET,SYSUDUMP,ADD,SDATA=(TRT,CB,ENQ,DM,IO,ERR,SUM),
  PDATA=(SA,REGS,LPA,JPA,PSW,SPLS,SUBTASKS)
  ```

- **SYSMDUMP**
System dump options

- **SYSABEND**

  ```
  CD, SET, SYSABEND, ADD, SDATA=(TRT, CB, ENQ, DM, IO, ERR, SUM),
  PDATA=(SA, REGS, LPA, JPA, PSW, SPLS)
  ```

- **SDUMP**

  ```
  SDUMP ADD(ALLPSA, NUC, SOA, LSQA, RGN, LPA, TRT, CSA, SOA,
  DUMDUMP, Q=NO), BUFFERS=00000000K,
  MAXSPACE=00001500M, MSGTIME=99999
  ```

**NOTE**

These changes are system-wide.
This appendix provides information about sample programs, JCL, macros and definitions, and user exit routines included on the distribution tape with APPLICATION RESTART CONTROL (AR/CTL). This appendix contains the following information:

Overview ................................................................. 271
Program samples ...................................................... 272
JCL samples .............................................................. 272
Macro and definition samples .................................... 273
User exit routines ...................................................... 274

Overview

BMC Software distributes samples with AR/CTL products. These samples include programs, JCL, macros and definitions, and user exit routines. You can use these samples as models for your own code. The tables in this section list members that contain samples mentioned in this manual.

The following libraries on the distribution tape contain samples:

- HLQ.ARCSAMP
- HLQ.ARCCUST

The variable HLQ is a high-level qualifier that should be easily identified by all installation users in your facility. Because the person who installed AR/CTL products at your site could have chosen any name, the guide refers to these libraries generically as sample libraries and install libraries.

The tables do not list all members in the sample and install libraries. Some members may have been added since this manual was last published. The members are listed in alphabetic order by member name.
Program samples

This section lists program samples (Table 26), including install verification programs (IVPs) and application programs that demonstrate AR/CTL usage in the supported environments and programming languages.

Table 26  Sample programs

<table>
<thead>
<tr>
<th>Member</th>
<th>BMC library</th>
<th>Environment</th>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCSVSCP</td>
<td>ARC.SAMP</td>
<td>any</td>
<td>COBOL</td>
<td>sample program that uses the Subprogram Virtual Storage (ARCSPVS) application program interface (API)</td>
</tr>
<tr>
<td>ASMORIGP</td>
<td>ARC.SAMP</td>
<td>any</td>
<td>Assembler</td>
<td>code sample to load the original OS parameter</td>
</tr>
<tr>
<td>CBLDB2F</td>
<td>ARC.SAMP</td>
<td>DB2</td>
<td>COBOL</td>
<td>DB2 fetch application program</td>
</tr>
<tr>
<td>CBLDB2L</td>
<td>ARC.SAMP</td>
<td>DB2</td>
<td>COBOL</td>
<td>DB2 load application program</td>
</tr>
<tr>
<td>CBLIMS1</td>
<td>ARC.SAMP</td>
<td>IMS</td>
<td>COBOL</td>
<td>IMS application program</td>
</tr>
<tr>
<td>CBLIMS2</td>
<td>ARC.SAMP</td>
<td>IMS</td>
<td>COBOL</td>
<td>IMS application program</td>
</tr>
<tr>
<td>CBLV1</td>
<td>ARC.SAMP</td>
<td>VSAM</td>
<td>COBOL</td>
<td>VSAM application program</td>
</tr>
<tr>
<td>IVP000</td>
<td>ARC.SAMP</td>
<td>QSAM</td>
<td>Assembler</td>
<td>program to populate the QSAM input file for IVP001 and IVP002</td>
</tr>
<tr>
<td>IVP001</td>
<td>ARC.SAMP</td>
<td>QSAM</td>
<td>Assembler</td>
<td>non-IMS IVP</td>
</tr>
<tr>
<td>IVP002</td>
<td>ARC.SAMP</td>
<td>QSAM</td>
<td>Assembler</td>
<td>IMS IVP (without the BATCH CONTROL FACILITY component of AR/CTL)</td>
</tr>
<tr>
<td>PLIDB2F</td>
<td>ARC.SAMP</td>
<td>DB2</td>
<td>PL/I</td>
<td>DB2 fetch application program</td>
</tr>
<tr>
<td>PLIDB2L</td>
<td>ARC.SAMP</td>
<td>DB2</td>
<td>PL/I</td>
<td>DB2 load application program</td>
</tr>
<tr>
<td>PLIIMS1</td>
<td>ARC.SAMP</td>
<td>IMS</td>
<td>PL/I</td>
<td>IMS application program</td>
</tr>
<tr>
<td>PLIV1</td>
<td>ARC.SAMP</td>
<td>VSAM</td>
<td>PL/I</td>
<td>VSAM application program</td>
</tr>
</tbody>
</table>

JCL samples

This section lists JCL samples (Table 27), including jobs that execute the sample application programs.
Table 27  Sample JCL

<table>
<thead>
<tr>
<th>Member</th>
<th>BMC library</th>
<th>Environment</th>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#ARCHIST</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>JCL</td>
<td>allocate AR/CTL history data set</td>
</tr>
<tr>
<td>#ARCLSTA</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>JCL</td>
<td>executes program AESURACT to list restart control records with status ABENDED or ACTIVE</td>
</tr>
<tr>
<td>#ARCRABN</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>JCL</td>
<td>executes program AESURABN to reset status of restart control records from ACTIVE to ABENDED</td>
</tr>
<tr>
<td>#ARCTR0</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>JCL</td>
<td>executes Trace Format utility (program ARCTR00)</td>
</tr>
<tr>
<td>#ARCURLS</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>JCL</td>
<td>reset the current shift identifier</td>
</tr>
<tr>
<td>#DB2LNK</td>
<td>ARC.SAMP</td>
<td>DB2</td>
<td>JCL</td>
<td>example to link-edit AR/CTL with DB2 application program</td>
</tr>
<tr>
<td>#SDCJOB1</td>
<td>ARC.SAMP</td>
<td>DB2 and COBOL</td>
<td>JCL</td>
<td>executes programs CBLDB2F AND CBLDB2L</td>
</tr>
<tr>
<td>#SDPJOB1</td>
<td>ARC.SAMP</td>
<td>DB2 and PL/I</td>
<td>JCL</td>
<td>executes programs PLIDB2F AND PLIDB2L</td>
</tr>
<tr>
<td>#SICJOB1</td>
<td>ARC.SAMP</td>
<td>IMS and COBOL</td>
<td>JCL</td>
<td>executes program CBLIMS1</td>
</tr>
<tr>
<td>#SICJOB2</td>
<td>ARC.SAMP</td>
<td>IMS and COBOL</td>
<td>JCL</td>
<td>executes program CBLIMS2</td>
</tr>
<tr>
<td>#SIPJOB1</td>
<td>ARC.SAMP</td>
<td>IMS and PL/I</td>
<td>JCL</td>
<td>executes program PLIIMS1</td>
</tr>
<tr>
<td>#SVCJOBn</td>
<td>ARC.SAMP</td>
<td>VSAM and COBOL</td>
<td>JCL</td>
<td>executes program CBLV1</td>
</tr>
<tr>
<td>#SVPJOBn</td>
<td>ARC.SAMP</td>
<td>VSAM and PL/I</td>
<td>JCL</td>
<td>executes program PLIV1</td>
</tr>
</tbody>
</table>

Macro and definition samples

This section lists macro and definition samples (Table 28).

Table 28  Sample macros and definitions

<table>
<thead>
<tr>
<th>Member</th>
<th>BMC library</th>
<th>Environment</th>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ARCASME</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>macro</td>
<td>parameter list for Data Services exit</td>
</tr>
<tr>
<td>$ARCITRD</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>macro</td>
<td>macro to map the IMS trace detail record (includes AR/CTL calls)</td>
</tr>
<tr>
<td>$ARCITRH</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>macro</td>
<td>macro to map the IMS trace header record (includes AR/CTL calls)</td>
</tr>
<tr>
<td>$ARCITRT</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>macro</td>
<td>macro to map the IMS trace trailer record (includes AR/CTL calls)</td>
</tr>
<tr>
<td>$ARC2TRD</td>
<td>ARC.SAMP</td>
<td>DB2</td>
<td>macro</td>
<td>macro to map the DB2 trace detail record</td>
</tr>
</tbody>
</table>
This section lists user exit routine samples (Table 29), which can be used in all AR/CTL modes and programming environments.

### Table 28  Sample macros and definitions

<table>
<thead>
<tr>
<th>Member</th>
<th>BMC library</th>
<th>Environment</th>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ARC2TRH</td>
<td>ARC.SAMP</td>
<td>DB2</td>
<td>macro</td>
<td>macro to map the DB2 trace header record</td>
</tr>
<tr>
<td>$ARC2TRT</td>
<td>ARC.SAMP</td>
<td>DB2</td>
<td>macro</td>
<td>macro to map the DB2 trace trailer record</td>
</tr>
<tr>
<td>$ARCSPVA</td>
<td>ARC.SAMP</td>
<td>any</td>
<td>Assembler</td>
<td>Assembler language DSECT to describe the ARCSPVS API parameter block</td>
</tr>
<tr>
<td>$ARCUABN</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>macro</td>
<td>DSECT to map active job step member keys with abended status</td>
</tr>
<tr>
<td>$ARCUACT</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>macro</td>
<td>DSECT to map active job step member keys with active status</td>
</tr>
<tr>
<td>$ARCUEXT</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>macro</td>
<td>communication control block for XRST and CHKP exits</td>
</tr>
<tr>
<td>$ARCUPUE</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>macro</td>
<td>parameter list for checkpoint pacing exit</td>
</tr>
<tr>
<td>ARCSPVSB</td>
<td>ARC.SAMP</td>
<td>any</td>
<td>COBOL</td>
<td>COBOL copybook for the ARCSPVS API parameter block</td>
</tr>
<tr>
<td>ARCSPVSP</td>
<td>ARC.SAMP</td>
<td>any</td>
<td>PL/I</td>
<td>PL/I definition of the ARCSPVS API parameter block</td>
</tr>
</tbody>
</table>

### Table 29  Sample user exit routines

<table>
<thead>
<tr>
<th>Member</th>
<th>BMC library</th>
<th>Environment</th>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCA5MUX</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>Assembler</td>
<td>sample ASAM exit</td>
</tr>
<tr>
<td>ARCCREXT</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>COBOL</td>
<td>sample Cursor Repositioning exit</td>
</tr>
<tr>
<td>ARCIPEUX</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>Assembler</td>
<td>sample Checkpoint Pacing user exit</td>
</tr>
<tr>
<td>ARCUJCL0</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>Assembler</td>
<td>sample Submit Restart exit</td>
</tr>
<tr>
<td>ARCSREX</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>Assembler</td>
<td>sample Checkpoint exit</td>
</tr>
<tr>
<td>ARCXRSTX</td>
<td>ARC.SAMP</td>
<td>all</td>
<td>Assembler</td>
<td>sample Restart exit</td>
</tr>
</tbody>
</table>
To implement AR/CTL for an application program, BMC Software recommends that you use a program registration record or an automatic registration record. However, if you do not want to use one of these records, you can change the job step JCL to execute program ARCCTRL instead of the application program.

Figure 40 shows the EXEC statement and the AES$ssid DD statement for executing program ARCCTRL.

Code the following parameters on the EXEC statement:

**profilename**
Required. Specify the name of the application program to be executed under AR/CTL.

**asbname**
Required. Specify the name of the application specification block (ASB) to use for this application program. You can code AUTO$ASB to use the default ASB.

**arcid**
Optional. You can specify an override AR/CTL identifier. If you omit it, AR/CTL uses the value of the Default AR/CTL Identifier (ARCID) option as it is set in the global options module. Retain the comma if you omit this parameter.
**ssm**
Required for DB2 applications. Specify a value to be appended to the ARC ID to form the subsystem name member. The member name is loaded from the data set that is identified with the ARCSSMLB DD statement. The member contains information such as which DB2 to connect to and is similar to the information that is supplied with the DDITV02 DD statement for IMS jobs.

**planname**
Required for DB2 applications if the DB2 plan name is different from the program name. Specify the DB2 plan name.

**/parms/parms**
Optional. You can specify application and run-time parameters.

If the program that is being executed is ARCCTRL, the AES$ssid DD statement works in the same way as a program registration record. It ensures that the job runs only under the subsystem ID of the BCSS that corresponds to the ssid value in the AES$ssid ddname.
Index

Symbols
#ARCTR0 sample 155
#ARCURSH sample 84
#DB2LNK sample 220
$ARCACTU sample 101
$ARCSPVS parameter block 186, 225, 243
/F command 147, 149

Numerics
0Cx-type system abends 80
-927 abend 52

A
abend
system 0Cx-type 80
ABEND macro 61
abend, -927 52
ACCEPT processing 176
Active Jobsteps Display panel 262
AES
communication with BMC Software subsystems 24
enhanced processing 24
AES Status Check utility 68
AES$ssid DD statement 72, 92
exclusion with 91
AESURABN utility 262
alternate index repositioning 121
Always Cold Start option 257
Application Enhancement Solutions primary menu 42
application program
samples 272
testing after AR/CTL implementation 91
APPLICATION RESTART CONTROL Active option 89
applications starting with DSN/DFS 50
APPLID table 127
AR/CTL
components 39
considerations 49
detecting active status 173, 220, 239
excluding job steps from participation 88
introduction to products 22
problems 254, 265
sample library 271
supported IMS region types 54
tasks 64
AR/CTL Status Check utility 100
ARB
ASAM 214, 215
coding for checkpoint call 181
use in ASAM call 212
use in checkpoint call 180
ARC DD statement 93
ARC/CTL exclusions 50
ARCACTIV program 100
ARCASAM call 132, 212
ARCHKP
call 179
ARCCCTRL
executing application program under 275
module not found 68
use with AES$ssid 92
ARCDLTRC
DD statement 151
ARCDYNPR data set 266
ARCECHK call 179
ARCL000 module 52
ARCPFNT data set 266
ARCPRT DD statement 94
ARCSPVS API
COBOL 182
PL/I 122, 241
ARCSSMLB DD statement 94, 99
ARCSTATS DD statement 95
ARCSYSIN DD statement 95
ARCTLPS2 plan 51
#ARCTR0 sample 155
ARCTRC0 program 155
#ARCURSH sample 84
ARCRST
call 178, 191
DD statement 256
ARCRST DD statement 94
ARCYRC0 program 133, 261
ARV DD statement 93
ARVDBLOG DD statement 96, 122
ARVFCTAB DD statement 96
ASAM
ARB 212, 214
call summary 202
callable interface 132, 212
function codes 215
I/O area 209, 213
PCB mask 209, 210
staged output 120, 130, 133
testing 91
through GSAM replacement 130, 202
trace data set 267
ASAM Copy utility 133, 261
ASAM data set
allocation 261
closing 208
copying for backup 133
copying for diagnostic purposes 266
destroyed or damaged 262
getting a unique record 206
getting the next record 204
inserting a record 207
moving 133
opening 204
out-of-space conditions 261
preparing for cold start 257
problems 261
ROLD function code 216
ASAM data set allocation parameters, data services 118
ASMORIGP sample 173, 220, 239
ASMTRACE DD statement 267
Assembler language
application programs 237
assemble and link issues 239
AUTO$ASB 46, 72
use with specific FCBs 73
automatic checkpoints
Fast Path consideration 55
automatic registration record 77
Automatic Restart Preparation utility 262

B
backout of VSAM changes 33, 121
batch LSR support 122
BCF
overview 25
required for application reattach 50
use with Application Reattach option 146
BCSS
AES$ssid DD statement 92
overview 43
BMC Primary Subsystem (BMCP) 43
BMC Software Database Utilities products 60
BMC Software Product Support 265
BMC Software, contacting 2
BMP application restarted as DLI 148
BMP-Only Program 265
Bypass Execution DName option 89
bypass use of AR/CTL data services 73

C
CAF programs 52
calls
AR/CTL-common 191
ARCASAM 132, 212
ARCCHKP 179
ARCECHK 179
ARCXRST 178, 191
ASAM 132, 212
checkpoint/restart in AR/CTL format 177
CHKP 199
CLSE 208
GN 204
GU 206
IMS-compatible checkpoint/restart 195
ISRT 207
OPEN 204
restart 178, 191, 197
summary 177
testing 91
XRST 197
cancel command 61
CASE tools 59
checkpoint
ARCCHKP call 179
ARCECHK call 179
CHKP call 199
DB2 commits 110
forced 110
forcing 182
I/O area 201
in doubt 264
message suppression 97
PCB mask 201
testing 91
testing frequency of 91
checkpoint data set
allocation 259
copying for diagnostic purposes 266
destroyed or damaged 260
overview 47
problems 258
size 259
checkpoint ID
automatic in ARCCHKP and ARCECHK calls 179
CHKP call 199
in checkpoint I/O area 201
overriding 96
checkpoint pacing 84
Fast Path consideration 54
checkpoint/restart
AR/CTL-common call 191
AR/CTL-format calls 177
ARB 181
ARCCHKP call 179
ARCECHK call 179
Index 279

ARCXRST call 178, 191
   call summary 177
   changes before restart 109
   CHKP call 199
   considerations 107
   DB2 commits 110
   implementation 171
   IMS-compatible calls 195
   methods 104
   processing 105
   program code to execute at restart 109
   requirements in programs 105
   XRST call 197
checkpoint/restart services with data services 116
   CHKP AREA END constant 177
   CHKP call
      checkpoint I/O area 201
      coding 199
   CHKPID parameter 96
CICS
   requirements 56
   requirements for remote VSAM access 126
   resource usage 127
   system failure 263
   CICS APPLID table 127
CLSE
   call 208
   function in ARB 215
COBOL
   ACCEPT processing 176
   application program changes 171
   compile and link issues 174
   considerations 176
   EXTEND option 176
   external data 176
   GOBACK statement 176
   STOP RUN statement 176
   supported versions 172
cold start
   COLD-type keywords 97
   of IMS 263
   commit processing 106
   common AR/CTL calls 191
   concatenated data sets 57
   CONCURRENT REORG for IMS 140
   connection ID table
      overview 126
   considerations
      checkpoint/restart 107
      COBOL 176
      PL/I 221
   considerations of AR/CTL 49
   control statements
      adding to application JCL 95
      errors written to ARCPRTINT DD statement 94
      processing options 95
   conventions, documentation 19
COPYDD keyword 157
   current shift identifier record 83
   cursor repositioning record 79
   CURSOR WITH HOLD option of DB2 106
   customer support 3
D
DASD compression 59
DATA PACKER® 59
   data services 115
   ASAM callable interface 132, 212
   ASAM data set allocation parameters 118
   ASAM through GSAM replacement 130, 202
   checkpoint/restart services with data services 116
   GDSs 118
   local VSAM access 121
   output sequential files 116
   preallocating data sets 117
   remote VSAM access 123
   sequential file interception 119
data sets
   concatenated 57
   multi-volume 132
   temporary 57, 60
   types supported by ASAM 132
   uncataloged 60
   database recovery 256
   Database Utilities products 60
DB2
   AR/CTL plan 51
   attach methods 71
   call attach facility 52
   commits translated to checkpoints 110
   connection information 94, 98
   connection types 51
   CURSOR WITH HOLD option 106
   data sharing groups 53
   external subsystem module table 99
   plan name different from program name 100
   requirements for AR/CTL support 50
   resource translation table 100
   services of AR/CTL 51
   subsystem member 99
DB2 CURSOR WITH HOLD 106
DB2 Trace Detail Information report 166
DB2 Trace Header Information report 162
DB2 Trace Trailer Information report 167
#DB2LNK sample 220
DDITV02 data set 99
DFS/DSN, applications starting with 50
DFSxxxI message suppression 97
diagnostics
   contacting BMC Software 265
   creating a tape for BMC Software 266
   disaster recovery 263
Display Active Jobstep panel 260
DL/I batch application
and BCF 25
reattach 146
restarted as BMP 148
DLIDATA keyword 159
DSN/DFS, applications starting with 50
DSNALI module 52
DSNELI module 52
DSNHLI module 52
DSNMIN10 module 99
dump options
SDUMP 270
SYSABEND 270
SYSDUMP 269
system 269
dump, snap 149
dynamic allocation record 82
dynamic backout log 96
dynamic backout of VSAM changes 33, 121

early termination support 145
use in testing 146
using 147
electronic documentation 17
Enhanced Call Trace option 152
enhanced call tracing
native read/write calls not traced 154
overview 150
starting and stopping 153
structures 150
trace records 154
trace reports 160
using 149
environment registration 75
ESDS
ASAM support 132
LOAD mode 124
update through a PATH 123
exclusion
AES$ssid DD statement 91
considerations 89
methods 88
program exclusion record 78
exclusions, ARC/CTL 50
Execution Summary report 136
EXTEND option 176
external data 176
external subsystem module table 99

Fast Path databases 54

FBS record format 120
FCBs
specific 73
use with AUTO$ASB 73
Force AR/CTL Execution option 88
FORCE function in ARB 182
forced checkpoints 110
FORMAT keyword 159
function code field in ARB 215
FUNCTION keyword 135, 157

GDGs, data services 118
general data services, considerations 116
GN call 204
GOBACK statement 176
GSAM
replacement forced with sequential interception 130, 202
replacement implementation 130, 202
RSA emulation 211
testing 91
GSAM replacement
multi-volume support 131
GU call
coding 206
RSA parameter 211

Hardware requirements 49
Help, online 17
history data set 46
HLQ.ARCCUST library 271
HLQ.ARCSAMP library 271

IKJEFT01 program 52, 71
implementation
ASAM callable interface 132, 212
checkpoint/restart 171
GSAM replacement 130, 202
local VSAM access services 121
of AR/CTL in application programs 48
overview 65
remote VSAM access services 123
sequential file interception 119
IMS
AES$ssid DD statement use 92
automatic checkpoint use 54
checkpoint records written to logs 260
CHKPID parameter 96
cold start 263
early termination 148
final checkpoint forced 54
requirements for AR/CTL support 53
restart at other than the selected checkpoint 258
restart on different IMS system 76
stop region command 61
terminating applications early 147
IMS environment registration record 75
IMS program exclusion record 78
IMS program registration record 76
IMS Trace Detail Information report
  AIB PCB variation 164
  ASAM, GSAM, or DB PCB variation 164
  I/O PCB variation 163
IMS Trace Header Information report 161
IMS Trace Trailer Information report 167
IMS-compatible calls
  checkpoint/restart 195
  GSAM call summary 202
IMS-compatible mode
  considerations 55
  linking 175
IMSLOGR DD statement 258
in doubt checkpoint 264
INQUERE SQLCA statement 177
INDD keyword 135, 157
INP operand of OPEN function 214
install library 271
interception of program exception conditions 80
ISPF interface overview 41
ISRT call
  coding 207
  RSA emulation 211
J
JCL statements
  ARC DD 93
  ARCPRT DD 94
  ARCSSMLB DD 94
  ARCTATS DD 95
  ARCSYSIN DD 95
  ARCRXST DD 94
  ARV DD 93
  ARVDBLOG DD 96
  ASAM copy utility 134
  IMSLOGR DD 258
  RCMDRLOG DD 96
  samples 272
  Trace Format utility 155
job scheduling package
  use with AR/CTL 59
K
keywords
  COLD-type 97
  COPYDD 157
  DLIDATA 159
  FORMAT 159
  FUNCTION 135, 157
  INDD 135, 157
  NOMSGS-type 97
  OUTDD 135, 157
  PAGE 157
  PCBNUMB 158
  STARTAFT 158
  STOPAFT 158
  SUBSYS 158
  TYPE 158
  WIDTH 158
KSDS
  support 132
  update through nonunique PATH 124
L
Language Environment
  ABTERMENC(ABEND) 176
  RPTOPTS(ON) 176
  TERMTHDACT(UADUMP) 176
language interface module 52
local VSAM access
  implementation 121
  trace data set 268
locks, release of 127
logging of VSAM changes 33, 121
LSR support 122
M
messages
  for problem determination 254
  suppressing 97
modify command, MVS 147, 149
multi-volume data sets 132
multi-volume GSAM data set 131
MVS system requirements 49
N
name token services 173, 220, 239
NOMSGS-type keywords 97
non-IMS environment registration record 75
non-IMS program exclusion record 78
non-IMS program registration record 76, 77
nonunique job step
APPLICATION RESTART CONTROL User Guide

PL/I
application programs 217
coding considerations 219
compile and link issues 220
te-time considerations 221
supported versions 218
plan ARCTLP$2 51
preallocating data sets, data services 117
print data sets 132
problems
ASAM data sets 261
checkpoint/restart data set 258
CICS system failure 263
cold start 256
nonunique job step restart 256
overview 254
restart at other than the selected checkpoint 257
system 262
with AR/CTL 265
processing options record 78
PROCOPT code in ASAM PCB mask 210
product support 3, 265
program exception handling 80
program exclusion 78
program registration 76
program registration method 68
program samples 272
publications, related 17

QSAM
FBS or VBS record format 120
open in update mode 120
questions about AR/CTL 265
QUICKSTART-to-AR/CTL bridge
introduction 32
QUICKSTART-to-AR/CTL bridge
subprogram support 182, 222, 241

RCMDBLOG DD statement 96, 122
READ function in ARB 215
reattach class record 80
RECOVERY MANAGER for IMS 140
recovery of a database 256
REGISET
overview 44
records 45
shared 60
REGISET Record Copy utility 92
registration
automatic 77
environment 69, 75

P
pacing class record 84
PAGE keyword 157
panels
Active Jobsteps Display 262
Application Enhancement Solutions primary menu 42
APPLICATION RESTART CONTROL primary menu 42
Display Active Jobstep 260
Operating Environment Display 265
overview 41
parameters
length of EXEC PARM string 173, 219, 238
original operating system 173, 220, 239
PARM string length 173, 219, 238
PCBNUMB keyword 158
PDS repositioning 57
performance
effect of RSA emulation 211
file sharing 58

Q
QSAM
FBS or VBS record format 120
open in update mode 120
questions about AR/CTL 265
QUICKSTART-to-AR/CTL bridge
introduction 32
QUICKSTART-to-AR/CTL bridge
subprogram support 182, 222, 241

R
RCMDBLOG DD statement 96, 122
READ function in ARB 215
reattach class record 80
RECOVERY MANAGER for IMS 140
recovery of a database 256
REGISET
overview 44
records 45
shared 60
REGISET Record Copy utility 92
registration
automatic 77
environment 69, 75

ARCXRST DD statement 94
restart 256

OCI
early termination support 147
enhanced call tracing 152
SNAP command 149
STOP command 147
STOP IMMEDIATE command 147
online Help 17
OPEN
call 204
function in ARB 214, 215
issuing restart request before 120, 121, 123
TYPE=J 120
OPEN I-O call 120
Operating Environment Display panel 265
operating system requirements 49
operations with AR/CTL 254
options
Always Cold Start 257
APPLICATION RESTART CONTROL Active 89
Bypass Execution DDname 89
Enhanced Call Trace 152
Force AR/CTL Execution 88
OUTDD keyword 135, 157
output data sets
restart and staging 120, 130, 133
RSA emulation 211
output sequential files, data services 116
OUT-type operand of OPEN function 214

P
pacing class record 84
PAGE keyword 157
panels
Active Jobsteps Display 262
Application Enhancement Solutions primary menu 42
APPLICATION RESTART CONTROL primary menu 42
Display Active Jobstep 260
Operating Environment Display 265
overview 41
parameters
length of EXEC PARM string 173, 219, 238
original operating system 173, 220, 239
PARM string length 173, 219, 238
PCBNUMB keyword 158
PDS repositioning 57
performance
effect of RSA emulation 211
file sharing 58
program 69, 76
related publications 17
remote VSAM access
  CICS APPLID table 127
  CICs requirements 126
  connection ID table 126
  ESDS in LOAD mode 124
  ESDS updates through PATH 123
  implementation 123
  KSDS update through nonunique PATH 124
  SYSPLEX support 34
trace data set 267
REORGL PLUS for DB2 Online Feature 143
reports
  ARCTSTATS DD statement 95
  DB2 Trace Detail Information 166
  DB2 Trace Header Information 162
  DB2 Trace Trailer Information 167
  Execution Summary 136
  IMS Trace Detail Information (AIB PCB variation) 164
  IMS Trace Detail Information (ASAM, GSAM, or DB PCB variation) 164
  IMS Trace Detail Information (I/O PCB variation) 163
  IMS Trace Header Information 161
  IMS Trace Trailer Information 167
  Trace Summary 168
repositioning of data sets
  PDS 57
  testing 91
  VSAM restrictions 121, 123
resource translation table 100
restart
  ARCXRST call 178, 191
  ASAM 133
  at other than the selected checkpoint 257
  on different IMS system 76
  program changes 109
  program code to execute at restart 109
  RPL changes 121
  staged output 120, 130, 133
  testing 91
  XRST call 197
restart control record 87
  deleting before cold start 257
  resolving checkpoint in doubt 264
  status 262
Restart Control Record Status Reset utility 262
restart data set
  allocation 259
  ARCXRST DD statement 94
  copying for diagnostic purposes 266
  destroyed or damaged 260
  problems 258
  restart of nonunique job step 256
  size 259
restart ID area 179, 193
restart services with data services 116
RMTTRACE DD statement 267
ROLD function in ARB 216
rollback processing 58
RSA
  bypassing 211
  coding 211
  field in ASAM PCB mask 210
  GN call 205
  GU call 206
  ISRT call 207
S
samples
  #ARCTRNC0 155
  #ARCURSH 84
  #DB2LNK 220
  $ARCACTU 101
  ASMOHIGP 173, 220, 239
  summary of 271
  trace record DSECTS 154
SDUMP, dump options 270
sequential file interception 119
snap dump 149
SQL return code class record 81
SQL return code interception 81
SQL statement types traced 166
SQLCA statement 177
staged output
  ISRT processing 207
  processing at restart 120, 130, 133
  ROLD processing 216
STARTAFT keyword 158
statistics record 87
Status Check utility
  AES 68
  AR/CTL 100
status codes
  if checkpoint bypassed 181, 193
  in ARB 215
  in ASAM ARB 215
  in ASAM PCB mask 210
  in checkpoint PCB mask 202
  in IMS calls 209
  in IMS-compatible calls 209
STEPLIB and uncataloged or temporary data sets 60
STOP command 147
STOP IMMEDIATE command 147
STOP RUN statement 176
STOPAFT keyword 158
stream-oriented I/O 59
Subprogram Virtual Storage API
  COBOL 182
  PL/I 222, 241
subprograms, support for 182, 222, 241
SUBSYS keyword 158
subsystem
   BCSS 43
   BMCP 43
subsystem member creation 99
support 265
support, customer 3
syntax statement conventions 20
SYSI literal 99
SYSABEND, dump options 270
SYSDUMP, dump options 269
SYSPLEX support for remote VSAM access services 34
system dump 268
system dump options 269
system log overflow 127
system problems 262
SYSUDUMP DD statement 268

T
Table 1-1 64
tape data sets 132
technical support 3
temporary data set in STEPLIB 60
temporary data sets 57
termination processing 176
testing
   AR/CTL with application 91
   early termination support 146
TOGGLE command
   resolving checkpoint in doubt 264
   restart at other than the selected checkpoint 257
trace data set
   allocation 151
   copying 157
   printing 157
   record types 154
   size 152
Trace Format utility
   control statement keywords 157
   JCL requirements 155
trace reports
   contents 160
   dump format 168
Trace Summary report 168
tracing ASAM data sets 267
tracing VSAM data sets 267, 268
TYPE keyword 158
TYPE=J OPEN requests 120

U
uncataloged data set in STEPLIB 60
update mode 120
user areas, testing 91
utilities

AES Status Check 68
AR/CTL Status Check 100
ASAM Copy 261
ASAM Copy (ARCYRC00) 133
Automatic Restart Preparation 262
REGISET Record Copy 92
Restart Control Record Status Reset 262
Trace Format (ARCTRC00) 155

V
VBS record format 120
VSAM
   batch application programs 121, 123
   batch LSR support 122
   copying data set for diagnostic purposes 266
   data sets supported by ASAM 132
   dynamic backout 33, 121, 146
   local VSAM access services 121
   logging 33, 121
   remote VSAM access services 123
   trace data set 267, 268
VSAM dynamic backout log 96
VSAM file control table data set
   DD statement 96
   overview 48
VSAM recovery record 87
VSMTRACE DD statement 268

W
WAL 47
WIDTH keyword 158
WITH HOLD, cursor 106
working storage 177
WRITE function in ARB 215
write-ahead log 47

X
XRST call
   checkpoint I/O area 201
   usage 197

U
uncataloged data set in STEPLIB 60
update mode 120
user areas, testing 91
utilities