

IBM Workload Automation



Driving IBM Workload Scheduler for z/OS

Version 9.3 SPE (Revised March 2018)

IBM Workload Automation



Driving IBM Workload Scheduler for z/OS

Version 9.3 SPE (Revised March 2018)

Note

Before using this information and the product it supports, read the information in "Notices" on page 365.

This edition applies to version 9, release 3, modification level 0 of IBM Workload Scheduler for z/OS (program number 5698-T08) and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 1999, 2016.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

© Copyright HCL Technologies Limited 2016, 2018.

Contents

Figures vii

Tables ix

About this publication xiii

Who should read this publication. xiii
Accessibility xiv
Technical training xiv
Support information xiv
Conventions used in this publication xiv
How to read syntax diagrams xv

Part 1. Programming interfaces 1

Chapter 1. The program interface (PIF) . 3

Program interface samples. 3
Related tools 3
 Batch command interface tool 3
 IBM Workload Scheduler for z/OS control
 language. 4
Communicating with EQQYCOM 4
Required data sets 5
Optional data set 5
Error messages 6
Parameter overview 6
 Action code. 7
 Resource code 7
 Data area 8
 Argument names and values 8
 Communication block 9
 Return code 10
Sequence of requests 10
Data area description and format 10
 Header format 10
 Data record format 11
Date considerations. 12
 Internal date representation 12
 Date arguments in PIF applications 12
 Updating application description run cycles with
 PIF 13
Security considerations 13
Running user-written programs compiled for older
scheduler versions 14
Overview of request types 14
DELETE request. 16
 Action code 16
 Resource code 16
 Data area 17
 Arguments 17
 Communication block address 24
 Return code 24
EXECUTE request 24
 Action code 25
 Resource code 25

 Data area 25
 Arguments 25
 Communication block address 25
 Return code 25
INIT request 25
 Action code 25
 Resource code 25
 Data area 26
 Arguments 26
 Communication block address 27
 Return code 27
INSERT request 28
 Action code 29
 Resource code 29
 Data area 30
 Arguments 30
 Communication block address 38
 Return code 38
LIST request 38
 Action code 38
 Resource code 38
 Data area 39
 Arguments 40
 Communication block address 49
 Return code 49
MODIFY request 50
 Action code 50
 Resource code 50
 Data area 50
 Arguments 50
 Communication block address 59
 Return code 59
OPTIONS request 59
 Action code 59
 Resource code 59
 Data area 60
 Arguments 60
 Communication block address 63
 Return code 63
REPLACE request 63
 Action code 63
 Resource code 64
 Data area 64
 Arguments 64
 Communication block address 65
 Return code 65
RESET request 65
 Action code 65
 Resource code 65
 Data area 65
 Arguments 65
 Communication block address 65
 Return code 66
SELECT request 66
 Action code 66
 Resource code 66

| | |
|--|----|
| Data area | 68 |
| Arguments | 68 |
| Communication block address | 77 |
| Return code | 77 |
| SETSTAT request | 78 |
| Action code | 78 |
| Resource code | 78 |
| Data area | 78 |
| Arguments | 78 |
| Communication block address | 78 |
| Return code | 79 |
| TERM request | 79 |
| Action code | 79 |
| Resource code | 79 |
| Data area | 79 |
| Arguments | 79 |
| Communication block address | 79 |
| Return code | 79 |
| JCL preparation using PIF | 80 |
| Substituting variables | 80 |
| Simulating variable substitution | 81 |

Chapter 2. The Application Programming Interface (API) 83

| | |
|--|----|
| Communicating with IBM Workload Scheduler for z/OS | 83 |
| CPI-C support provided by IBM Workload Scheduler for z/OS | 84 |
| API buffer layouts | 85 |
| APP - Fixed section | 86 |
| APPOBJ - Object section | 88 |
| APPSEL - Selection section | 90 |
| APPVAL - Selection value section | 91 |
| APPFLD - Field section | 92 |
| APPDAT - Data section | 93 |
| Specifying object names | 93 |
| Selecting object instances | 94 |
| Specifying key types | 94 |
| Specifying selection criteria | 95 |
| Broadcasting events | 95 |
| Selecting object fields to update or retrieve | 96 |
| Return codes and reason codes generated by IBM Workload Scheduler for z/OS | 96 |
| Return codes and reason codes generated in the fixed section (APP) | 96 |
| Return codes and reason codes generated in the object section (APPOBJ) | 97 |
| Security | 98 |
| APPC and RACF | 98 |
| IBM Workload Scheduler for z/OS and RACF | 98 |

Part 2. Programming tools 101

Chapter 3. Batch command interface tool. 103

| | |
|--|-----|
| Online tools | 103 |
| The batch command interface | 103 |
| Input to batch command interface | 103 |
| BCIT output | 106 |
| Instructions | 107 |

| | |
|----------------------------|-----|
| COPY | 108 |
| DELETE | 110 |
| EXPORT | 120 |
| GROUPDEF support | 120 |
| IMPORT | 121 |
| INSERT | 123 |
| LIST | 133 |
| LISTSTAT | 142 |
| MODIFY | 144 |
| OPTIONS | 150 |
| REPLACE | 151 |
| SELECT | 151 |
| SETSTAT | 160 |

Chapter 4. Control Language (OCL) 163

| | |
|--|-----|
| What you can do using OCL | 163 |
| Advantages of OCL | 164 |
| Summary of OCL instructions | 164 |
| Customizing OCL | 167 |
| Specifying the initialization parameters | 169 |
| Example 1 | 169 |
| Example 2 | 169 |
| Example 3 | 169 |
| Example 4 | 170 |
| Obtaining access authorization | 170 |
| Logging executed instructions | 171 |
| Specifying OCL instructions | 171 |
| Specifying input arrival dates and times | 172 |
| Description of OCL instructions | 175 |
| INIT | 228 |
| JSUACT | 229 |
| KILLJOB | 230 |
| KILLREC | 232 |
| LABEL | 234 |
| MODCOND | 234 |
| MODOP | 236 |
| NOP | 241 |
| OPSTAT | 242 |
| PROMPTN | 245 |
| PROMPTY | 247 |
| RELEASE | 249 |
| RELOP | 251 |
| RELSUCC | 253 |
| SET | 255 |
| Concatenation operators | 255 |
| Built-In Functions | 255 |
| SETUPD | 257 |
| SRSTAT | 258 |
| UNNOP | 261 |
| UPD | 262 |
| WSSSTAT | 263 |
| WTO | 265 |
| Requirements | 265 |
| Sample job and procedure | 265 |
| EQYRJC sample job | 265 |
| EQYRPRC sample procedure | 266 |
| Messages | 267 |
| Message format | 267 |

Part 3. Appendixes 269

Appendix A. Program interface record format

| | | | |
|--|-----|---|-----|
| format | 271 | Current plan workstation (resource codes CPWS, CPWSCOM) | 307 |
| TOD fields | 271 | CPWS - Common segment | 307 |
| Application description (resource codes AD, ADCOM) | 272 | CPIVL - Current plan workstation open interval segment | 309 |
| ADAPD - Application dependency segment | 272 | CPOPT - workstation description record segment | 310 |
| ADCIV - Interval definition for conditional external predecessor segment | 273 | Current plan virtual workstation destination (resource codes CPWSV, CPWSVCOM) | 311 |
| ADCOM - Common segment | 274 | CPWSV - Common segment | 311 |
| ADDEP - Dependency segment | 275 | CPVIVL - Current plan virtual workstation destination open interval segment | 313 |
| ADCNC - Condition segment | 276 | Current plan special resource (resource codes CSR, CSRCOM) | 313 |
| ADCNS - Condition dependency segment | 276 | CSRCOM - Current plan resource common segment | 314 |
| ADEXT - Extended name segment | 277 | CSRIVL - Current plan special resource interval segment | 316 |
| ADKEY - Key segment | 277 | CSRIWS - Current plan resource interval "connected" workstation | 316 |
| ADOP - Operation segment | 277 | CSRDWS - Current plan resource default "connected" workstation | 317 |
| ADRE - Remote job information segment | 279 | ETT - Event triggered tracking criteria segment | 317 |
| ADRUN - Run cycle segment | 280 | Dates generated by run cycle rules (resource code GENDAYS) | 318 |
| ADSAI - Operation system automation information segment | 282 | JCL setup variables (resource codes JCLPREP, JCLPREPA) | 318 |
| ADSR - Special resource segment | 283 | JSVC - Common segment | 319 |
| ADUSF - User field segment | 284 | JSVV - Variable definition segment | 319 |
| ADVDD - Variable Duration and Deadline | 284 | JCL variable table (resource codes JCLV, JCLVCOM) | 319 |
| ADXIV - Interval definition for external predecessor segment | 284 | JCLVC - Common segment | 320 |
| All workstations closed (resource code AWSCL) | 285 | JCLVV - Variable definition segment | 320 |
| AWSCL - All workstations closed interval segment | 285 | JCLVD - Dependency segment | 321 |
| Calendar (resource codes CL, CLCOM) | 286 | Job control language (resource codes JS, JSCOM) | 321 |
| CLCOM - Common segment | 286 | JS - Job control language segment | 322 |
| CLSD - Specific date segment | 287 | Job log (resource code JLCOM) | 323 |
| CLWD - Weekday segment | 287 | JLCOM - Common segment | 323 |
| Current plan condition (resource codes CPCOND, CPCONDCO) | 288 | Long-term plan occurrence (resource codes LTOC, LTOCCOM) | 323 |
| CPCOND - Condition segment | 288 | LTOC - Common segment | 324 |
| CPSIMP - Condition dependency segment | 288 | LTOP - Operation segment | 325 |
| Current plan occurrence (resource code CPOC, CPOCCOM) | 289 | LTCPRE - Conditional predecessor segment | 326 |
| CPOC - Current plan occurrence segment | 289 | LTCSUC - Conditional successor segment | 326 |
| CPOCPRE - Occurrence predecessor segment | 291 | LTPRE - Predecessor segment | 326 |
| CPOCSUC - Occurrence successor segment | 292 | LTSUC - Successor segment | 327 |
| Current plan operation (resource codes CPOP, CPOPCOM) | 293 | LTEXT - External run cycle group for variable duration and deadline | 327 |
| CPCPR - Conditional predecessor segment | 293 | Operator instruction (resource codes OI, OICOM) | 327 |
| CPCSU - Conditional successor segment | 294 | OI - Operator instruction segment | 328 |
| CPEXT - Operation extended name segment | 294 | Period (resource codes PR, PRCOM) | 328 |
| CPOP - Common segment | 294 | PR - Period segment | 329 |
| CPOPSRU - Special resource usage segment | 299 | Run cycle group (resource codes RG, RGCOM) | 330 |
| CPPRE - Predecessor segment | 300 | RGCOM - Common segment | 330 |
| CPREND - Distributed remote job info segment | 301 | RGRUN - Run cycle segment | 331 |
| CPRENTZ - z/OS remote job info segment | 302 | Special resource (resource codes SR, SRCOM) | 332 |
| CPSAI - Operation system automation information segment | 303 | Workstation description (resource codes WS, WSCOM) | 335 |
| CPSUC - Successor segment | 303 | WSCOM - Common segment | 336 |
| CPSR - Special resource segment | 304 | WSEST - Destination segment | 337 |
| CPREC - Operation recovery segment | 304 | WSIVL - Open interval segment | 338 |
| Current plan status (resource code CPST) | 305 | | |
| CPST - Common segment | 305 | | |
| Current plan operation user field (resource codes CPUSRF, CPUSRFEM) | 306 | | |
| CPUSRF - Operation user field segment | 307 | | |

| | |
|---|-----|
| WSSD - Specific date segment | 338 |
| WSWD - Weekday segment | 338 |
| WSAM - Workstation access method segment | 339 |
| WSOPT - workstation description record segment | 340 |
| Virtual workstation destination description (resource codes WSV, WSVCOM) | 340 |
| WSVCOM - Common segment | 341 |
| WSVIVL - Open interval segment | 342 |
| WSVSD - Specific date segment | 342 |
| WSVWD - Weekday segment | 343 |

Appendix B. API object fields 345

| | |
|---|-----|
| Current plan status object | 345 |
| Current plan operation object | 346 |
| Current plan special resource object | 351 |
| Current plan workstation object | 352 |
| Current plan open interval object. | 354 |
| Current plan operation event object | 354 |
| Current plan OPINFO event object | 356 |
| Current plan special resource event object. | 357 |

| | |
|---|-----|
| Current plan backup event object. | 358 |
| Current plan workstation event object | 359 |

**Appendix C. Sample library
(SEQQSAMP) 361**

| | |
|--|-----|
| IBM Workload Scheduler for z/OS Application Programming Interface | 361 |
| API buffer examples | 361 |
| IBM Workload Scheduler for z/OS program interface | 362 |
| JS data set maintenance | 362 |
| JCL variable substitution | 362 |
| Current plan and LTP actions | 363 |
| Other PIF samples. | 363 |

Notices 365

| | |
|--|-----|
| Trademarks | 367 |
| Terms and conditions for product documentation | 367 |

Index 369

Figures

| | | | |
|--|----|--|-----|
| 1. Program interface parameters | 7 | 4. Example of arguments for processing a list | 69 |
| 2. Program interface arguments in TSO command notation | 9 | 5. Example of a send buffer layout for a GET request | 86 |
| 3. Program interface data area example | 11 | 6. Application flow example | 169 |

Tables

| | | | |
|--|----|---|----|
| 1. Comparison of Date Representations | 13 | 53. List CPCONDCO Arguments | 43 |
| 2. Access Authority for Program Interface Requests | 13 | 54. List CPOC Arguments | 43 |
| 3. Program Interface Resources and the Corresponding IBM Workload Scheduler for z/OS Fixed Resources Used for Checking Authorization | 13 | 55. List CPOPCOM Arguments | 44 |
| 4. Records Using a Common Segment | 15 | 56. List CPOPSRU Arguments | 45 |
| 5. Delete AD Arguments | 17 | 57. List CPWSCOM Arguments | 45 |
| 6. Delete AWSCL Arguments | 18 | 58. List CPWSVCOM Arguments | 46 |
| 7. Delete CL Arguments | 18 | 59. List CSRCOM Arguments | 46 |
| 8. Delete CPCOND Arguments | 18 | 60. List ETT Arguments. | 46 |
| 9. Delete CPOC Arguments | 18 | 61. List GENDAYS Arguments | 46 |
| 10. Delete CPOCPRE Arguments | 18 | 62. List JCLVCOM Arguments | 47 |
| 11. Delete CPOCSUC Arguments | 18 | 63. List JLCOM Arguments | 47 |
| 12. Delete CPOP Arguments | 19 | 64. List JSCOM Arguments | 48 |
| 13. Delete CPPRE Arguments | 19 | 65. List LTOCCOM Arguments | 48 |
| 14. Delete CPSIMP Arguments | 19 | 66. List OICOM Arguments | 48 |
| 15. Delete CPSR Arguments | 20 | 67. List PRCOM Arguments | 48 |
| 16. Delete CPSUC Arguments. | 20 | 68. List RGCOM, RGKEY Arguments | 48 |
| 17. Delete CPUSRF Arguments | 21 | 69. List SRCOM Arguments | 49 |
| 18. Delete ETT Arguments. | 21 | 70. List WSCOM Arguments | 49 |
| 19. Delete IVL Arguments | 21 | 71. List WSVCOM Arguments | 49 |
| 20. Delete JCLV Arguments | 21 | 72. Modify CPCOND Arguments | 51 |
| 21. Delete JL Arguments | 21 | 73. Modify CPEXT Arguments | 51 |
| 22. Delete JS, JSCOM Arguments | 22 | 74. Modify CPOC Arguments. | 51 |
| 23. Delete LTOC Arguments | 22 | 75. Modify CPOP Arguments | 52 |
| 24. Delete LTCPRE Arguments | 22 | 76. Modify CPREND Arguments. | 54 |
| 25. Delete LTPRE Arguments | 22 | 77. Modify CPRENTZ Arguments. | 54 |
| 26. Delete OI Arguments | 23 | 78. Modify CPSAI Arguments | 55 |
| 27. Delete PR Arguments | 23 | 79. Modify CPUSRF Arguments | 55 |
| 28. Delete RG Arguments | 23 | 80. Modify CPWS Arguments. | 56 |
| 29. Delete SR Arguments | 23 | 81. Modify CPWSV Arguments | 56 |
| 30. Delete VIVL Arguments | 24 | 82. Modify CSR Arguments | 57 |
| 31. Delete WS Arguments | 24 | 83. Modify IVL Arguments | 58 |
| 32. Delete WSV Arguments | 24 | 84. Modify LTOC Arguments | 58 |
| 33. Insert CPOC Arguments | 31 | 85. Modify VIVL Arguments | 58 |
| 34. Insert CPOCPRE Arguments | 31 | 86. Replace AD Arguments | 64 |
| 35. Insert CPOCSUC Arguments. | 32 | 87. Select AD, ADCOM Arguments | 70 |
| 36. Insert CPCOND Arguments | 32 | 88. Select AWSCL Arguments. | 70 |
| 37. Insert CPOP Arguments | 32 | 89. Select CL, CLCOM Arguments | 70 |
| 38. Insert CPPRE Arguments | 33 | 90. Select CPCOND, CPCONDCO Arguments | 71 |
| 39. Insert CPSAI Arguments | 34 | 91. Select CPOC Arguments | 71 |
| 40. Insert CPSIMP Arguments | 34 | 92. Select CPOP, CPOPCOM Arguments | 71 |
| 41. Insert CPSR Arguments | 35 | 93. Select CPUSRF Arguments | 73 |
| 42. Insert CPSUC Arguments | 35 | 94. Select CPWS, CPWSCOM Arguments | 73 |
| 43. Insert CPUSRF Arguments | 36 | 95. Select CPWSV, CPWSVCOM Arguments | 73 |
| 44. Insert IVL Arguments | 36 | 96. Select CSR, CSRCOM Arguments | 74 |
| 45. Insert JCLPREP Arguments | 36 | 97. Select ETT Arguments | 74 |
| 46. Insert JCLV Arguments. | 36 | 98. Select JCLPREP Arguments | 74 |
| 47. Insert LTOC Arguments | 37 | 99. Select JCLPREPA Arguments | 74 |
| 48. Insert LTPRE Arguments | 37 | 100. Select JCLV, JCLVCOM Arguments. | 75 |
| 49. Insert VIVL Arguments | 37 | 101. Select JLCOM Arguments | 75 |
| 50. List ADCOM and ADKEY Arguments. | 42 | 102. Select JS, JSCOM Arguments | 75 |
| 51. List AWSCL Arguments | 42 | 103. Select LTOC, LTOCCOM Arguments | 75 |
| 52. List CLCOM Arguments | 43 | 104. Select OI, OICOM Arguments | 76 |
| | | 105. Select PR, PRCOM Arguments | 76 |
| | | 106. Select RG, RGCOM Arguments | 76 |
| | | 107. Select SR, SRCOM Arguments | 76 |
| | | 108. Select WS, WSCOM Arguments | 77 |

| | | | |
|---|-----|---|-----|
| 109. Select WSV, WSVCOM Arguments | 77 | 163. Clock value setting at different time interval | 271 |
| 110. Setstat CPSIMP Argument | 78 | 164. ADAPD Control Block | 272 |
| 111. Contents of a Send Buffer | 85 | 165. ADCIV Control Block | 274 |
| 112. App-Fixed Section | 86 | 166. ADCOM Control Block | 274 |
| 113. APPOBJ-Object Section | 88 | 167. ADDEP Control Block | 275 |
| 114. APPSEL-Selection Section | 91 | 168. ADEXT Control Block | 277 |
| 115. APPVAL-Selection Value Section | 92 | 169. ADOP Control Block | 278 |
| 116. APPFLD-Field Section | 92 | 170. ADRE Control Block | 279 |
| 117. APPDAT-Data Section | 93 | 171. ADRUN Control Block | 281 |
| 118. API Object Names | 93 | 172. Run Cycle Offsets | 281 |
| 119. Operators That You Can Specify in the APPSEL Section | 95 | 173. Rule Definition | 282 |
| 120. Subresource Protection for Requests through the API | 99 | 174. ADSAI Control Block | 282 |
| 121. Positional parameters that can be passed with the Batch Command Interface | 104 | 175. ADSR Control Block | 283 |
| 122. Access Authorizations | 170 | 176. ADUSF Control Block | 284 |
| 123. Input Arrival Date and Time Keywords | 172 | 177. ADVDD Control Block | 284 |
| 124. How OCL Uses the IADATE, IATIME, and IA Keywords | 173 | 178. ADXIV Control Block | 285 |
| 125. IBM Workload Scheduler for z/OS-supplied Variables | 175 | 179. AWSCL Control Block | 285 |
| 126. Keywords Used in the Add Instruction | 176 | 180. CLCOM Control Block | 286 |
| 127. Keywords used in the Addcond Instruction | 177 | 181. CLSD Control Block | 287 |
| 128. Keywords used in the Addop Instruction | 181 | 182. CLWD Control Block | 288 |
| 129. Keywords used in the Addpred Instruction | 185 | 183. CPOC Control Block | 289 |
| 130. Keywords used in the Addres Instruction | 187 | 184. CPOCPRE Control Block | 291 |
| 131. Keywords used in the Addsimp Instruction | 189 | 185. CPOCSUC Control Block | 292 |
| 132. Keywords used in the Chgextname Instructions | 192 | 186. CPEXT Control Block | 294 |
| 133. Keywords used in the Chgjob Instructions | 194 | 187. CPOP Control Block | 295 |
| 134. Keywords used in the Chgopsai Instructions | 195 | 188. CPOPSRU Control Block | 300 |
| 135. Keywords used in the Chkappl Instruction | 198 | 189. CPPRE Control Block | 301 |
| 136. Chkdate Instruction Variables | 201 | 190. CPREND Control Block | 302 |
| 137. Keywords used in the Compl Instruction | 209 | 191. CPRENTZ Control Block | 302 |
| 138. Keywords used in the Del Instruction | 211 | 192. CPSAI Control Block | 303 |
| 139. Keywords used in the Delcond Instruction | 213 | 193. CPSUC Control Block | 303 |
| 140. Keywords used in the Delpred Instruction | 214 | 194. CPSR Control Block | 304 |
| 141. Keywords used in the Delres Instruction | 216 | 195. CPREC Control Block | 304 |
| 142. Keywords used in the Delsimp Instruction | 218 | 196. CPST Control Block | 306 |
| 143. Keywords used in the Force Instruction | 221 | 197. CPUSRF Control Block | 307 |
| 144. Keywords used in the Hold Instruction | 224 | 198. CPUSRFELEM Control Block | 307 |
| 145. Keywords used in the Init Instruction | 228 | 199. CPWS Control Block | 308 |
| 146. Keywords used in the JSUACT Instruction | 229 | 200. CPIVL Control Block | 309 |
| 147. Keywords used in the Killjob Instruction | 230 | 201. CPOPT Control Block | 310 |
| 148. Keywords used in the Killrec Instruction | 232 | 202. CPWSV Control Block | 311 |
| 149. Keywords used in the Modcond Instruction | 234 | 203. CPVIVL Control Block | 313 |
| 150. Keywords used in the Modop Instructions | 236 | 204. CSRCOM Control Block | 314 |
| 151. Operations Details that can be modified | 239 | 205. CSRIVL Control Block | 316 |
| 152. Keywords used in the Nop Instruction | 241 | 206. CSRIWS Control Block | 316 |
| 153. Keywords used in the Opstat Instruction | 242 | 207. CSRDWS Control Block | 317 |
| 154. Keywords used in the PROMPTN Instruction | 245 | 208. ETT Control Block | 317 |
| 155. Keywords used in the PROMPTY Instruction | 247 | 209. GNDAY Control Block | 318 |
| 156. Keyword used in the Release Instruction | 249 | 210. JSVC Control Block | 319 |
| 157. Keywords used in the Relop Instruction | 252 | 211. JSVV Control Block | 319 |
| 158. Keywords used in the Relsucc Instruction | 253 | 212. JCLVC Control Block | 320 |
| 159. Keywords used in the Srstat Instruction | 258 | 213. JCLVV Control Block | 320 |
| 160. Keywords used in the Unnop Instruction | 261 | 214. JCLVD Control Block | 321 |
| 161. Keywords used in the Wsstat Instruction | 263 | 215. JS Control Block | 322 |
| 162. Clock value setting at the start of different years | 271 | 216. JLCOM Control Block | 323 |
| | | 217. LTOC Control Block | 324 |
| | | 218. LTOP Control Block | 325 |
| | | 219. LTPRE Control Block | 326 |
| | | 220. LTSUC Control Block | 327 |
| | | 221. LTEXT Control Block | 327 |
| | | 222. OI Control Block | 328 |
| | | 223. PR Control Block | 329 |

| | | | |
|--|-----|--|-----|
| 224. Period Origin Dates | 329 | 240. WSVCOM Control Block | 341 |
| 225. Period Interval End Dates | 330 | 241. WSVIVL Control Block | 342 |
| 226. RGCOP Control Block | 330 | 242. WSVSD Control Block | 342 |
| 227. RGRUN Control Block | 331 | 243. WSWD Control Block | 343 |
| 228. Rule Definition | 332 | 244. CP_STATUS Object Fields | 345 |
| 229. SRCOM Control Block | 333 | 245. CP_OPERATION Object Fields | 346 |
| 230. SRIVL Segment | 334 | 246. CP_RESOURCE Object Fields | 351 |
| 231. SRIWS Segment | 335 | 247. CP_WORK_STATION Object Fields | 352 |
| 232. SRDWS Segment | 335 | 248. CP_OPEN_INTERVAL Object Fields | 354 |
| 233. WSCOM Control Block | 336 | 249. CP_OPER_EVENT Object Fields | 354 |
| 234. WSDEST Control Block | 337 | 250. CP_OPINFO_EVENT Object Fields | 356 |
| 235. WSIVL Control Block | 338 | 251. CP_SR_EVENT Object Fields | 357 |
| 236. WSSD Control Block | 338 | 252. BACKUP_EVENT Object Fields | 358 |
| 237. WSWD Control Block | 339 | 253. CP_WS_EVENT Object Fields | 359 |
| 238. WSAM Control Block | 339 | 254. SEQQSAMP Library Members for Programming Interfaces and the API. | 361 |
| 239. WSOPT Control Block | 340 | | |

About this publication

Developer's Guide: Driving IBM Workload Scheduler for z/OS shows you how to use the programming interfaces to IBM Workload Scheduler for z/OS to help you plan, schedule, and monitor work in the production department of your computer installation.

Your workload can run on various platforms, but you control it from a central z/OS® system that runs the IBM Workload Scheduler for z/OS controller.

This guide is part of a set of guides that allows you to program many aspects of working with the products in the IBM Workload Automation family. These guides comprise:

- *IBM Workload Automation: Developer's Guide: Driving IBM Workload Scheduler for z/OS*
- *IBM Workload Automation: Developer's Guide: Extending IBM Workload Automation*
- *IBM Workload Automation: Developer's Guide: Workload Automation Programming Language*

Note: If you control your z/OS controller using Dynamic Workload Console, information about the programming interfaces you can use with the Dynamic Workload Console are available in both of the other Developer's Guides in the set.

The term *scheduler*, when used in this publication, refers to IBM Workload Scheduler for z/OS. The term DB2®, when used in this publication, refers to DATABASE 2 and DB2 Universal Database™.

The term z/OS is used in this publication to mean z/OS and OS/390® operating systems. Where the term OS/390 appears, the related information applies only to OS/390 operating systems.

Who should read this publication

This publication is for users who write application programs that request services from IBM Workload Automation.

This publication documents the programming interface (PIF) and the application programming interface (API). To use PIF you must know job control language (JCL) and have a good working knowledge of a programming language, for example, assembler or PL/I. You can use programming languages that support z/OS and OS/390 linkage conventions and that can load and delete an assembler program.

To use the API, you require a knowledge of Advanced Program-to-Program Communication (APPC). You must be able to write application transaction programs (ATPs) that use the services of APPC. Because the API is implemented using a subset of CPI-C (Common Programming Interface for Communications) verbs, you must be able to write ATPs that use CPI-C.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully.

With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

For full information, see the Accessibility Appendix in the *IBM Workload Scheduler User's Guide and Reference*.

Technical training

Cloud & Smarter Infrastructure provides technical training.

For Cloud & Smarter Infrastructure technical training information, see:
<http://www.ibm.com/software/tivoli/education>

Support information

IBM provides several ways for you to obtain support when you encounter a problem.

If you have a problem with your IBM software, you want to resolve it quickly. IBM provides the following ways for you to obtain the support you need:

- Searching knowledge bases: You can search across a large collection of known problems and workarounds, Technotes, and other information.
- Obtaining fixes: You can locate the latest fixes that are already available for your product.
- Contacting IBM Software Support: If you still cannot solve your problem, and you need to work with someone from IBM, you can use a variety of ways to contact IBM Software Support.

For more information about these three ways of resolving problems, see the appendix about support information in *IBM Workload Scheduler: Troubleshooting Guide*.

Conventions used in this publication

Conventions used in this publication.

The publication uses several typeface conventions for special terms and actions. Technical changes to the text are indicated by a vertical line to the left of the change. These conventions have the following meanings:

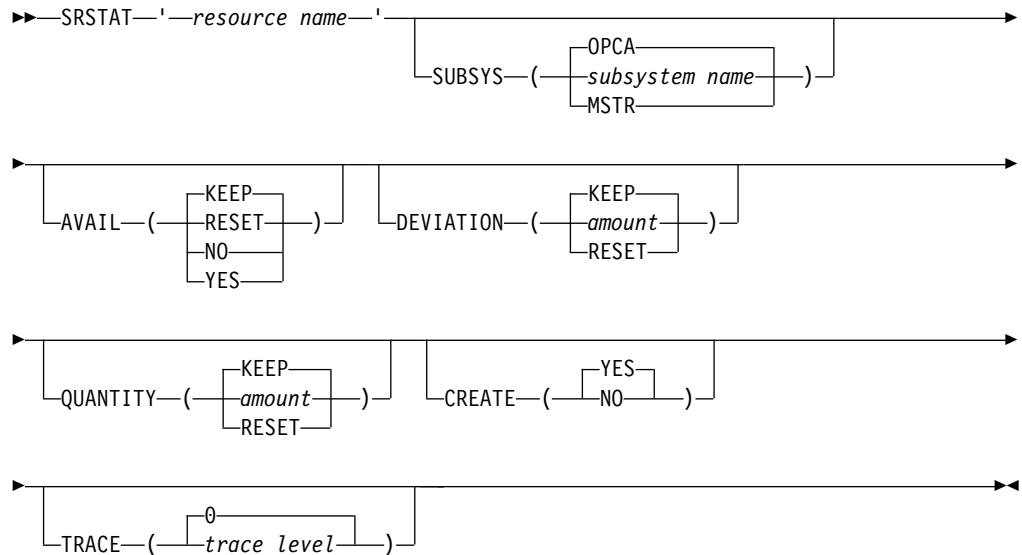
| Information type | Style convention | Example |
|---|---------------------|---------------|
| Commands | All capital letters | CREATE |
| References in the text to fields on panels | All capital letters | QUANTITY |
| File and directory names, input you should type in panel fields | Monospace | MYAPPLICATION |

| Information type | Style convention | Example |
|--|------------------|--------------------|
| First time new term introduced, publication titles | Italics | <i>Application</i> |

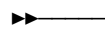
How to read syntax diagrams

Syntax diagrams help to show syntax in a graphical way.

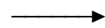
Throughout this publication, syntax is described in diagrams like the one shown here, which describes the SRSTAT TSO command:



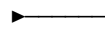
The symbols have these meanings:



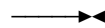
The statement begins here.



The statement is continued on the next line.



The statement is continued from a previous line.

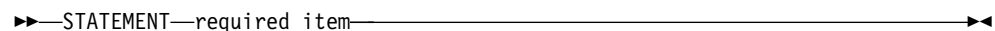


The statement ends here.

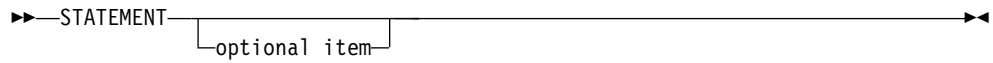
Read the syntax diagrams from left to right and from top to bottom, following the path of the line.

These are the conventions used in the diagrams:

- Required items appear on the horizontal line (main path):



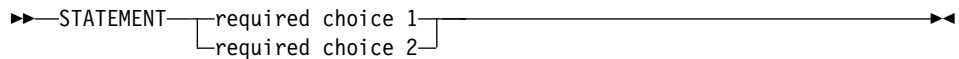
- Optional items appear below the main path:



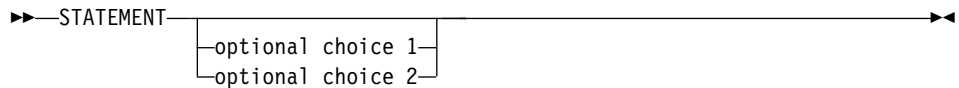
- An arrow returning to the left above the item indicates an item that you can repeat. If a separator is required between items, it is shown on the repeat arrow.



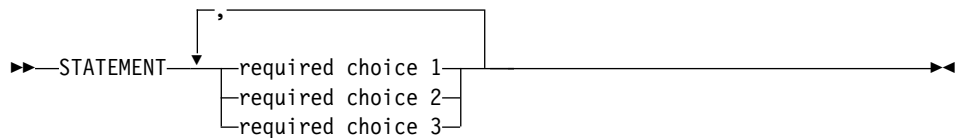
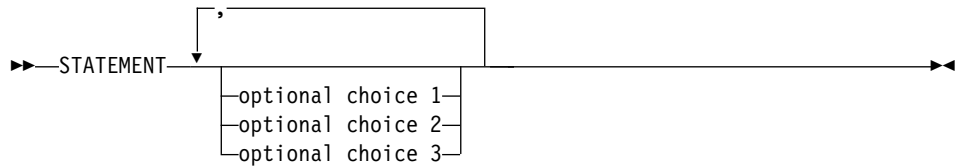
- If you can choose from two or more items, they appear vertically in a stack.
 - If you must choose one of the items, one item of the stack appears on the main path:



- If choosing one of the items is optional, the entire stack appears below the main path:



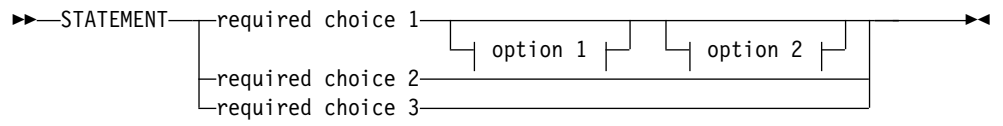
- A repeat arrow above a stack indicates that you can make more than one choice from the stacked items:



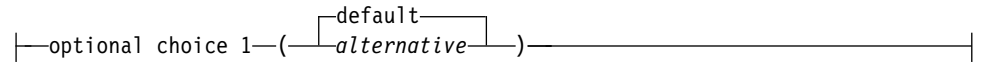
- Parameters that are above the main line are default parameters:



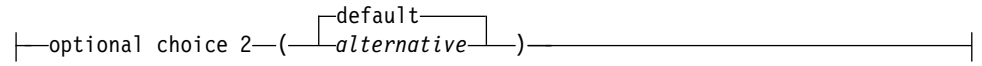
- Keywords appear in uppercase (for example, STATEMENT).
- Parentheses and commas must be entered as part of the command syntax, as shown.
- For complex commands, the item attributes might not fit on one horizontal line. If that line cannot be split, the attributes appear at the bottom of the syntax diagram:



option 1



option 2



Part 1. Programming interfaces

Chapter 1. The program interface (PIF)

About this task

This chapter describes the IBM Workload Scheduler for z/OS program interface (PIF), which lets a user-written program issue various requests to the IBM Workload Scheduler for z/OS subsystem. For example, you can automate actions that you perform through the IBM Workload Scheduler for z/OS dialogs.

The program interface supports these basic requests:

Database requests

- Read and update information from the application description and operator instruction databases.
- Read information from the workstation description and calendar databases.

LTP requests

Read and update occurrences in the LTP data set.

Current plan requests

Read and update this information in the current plan data set:

- Occurrences
- Operations
- Workstations

The program interface is supported using an IBM Workload Scheduler for z/OS-supplied communication subroutine, EQQYCOM.

Program interface samples

The IBM Workload Scheduler for z/OS sample library shipped with the IBM Workload Scheduler for z/OS programs contains many sample programs that use the program interface function. These programs will execute successfully with a few minor changes to suit your installation. You can continue to run them as they are, or use them as a model to create your own programs. See Appendix C, “Sample library (SEQQSAMP),” on page 361 for a description of the PIF sample members provided in the SEQQSAMP library.

Related tools

IBM Workload Scheduler for z/OS is delivered with some tools that take advantage of the PIF. These tools are provided as additional aids in the process of automating workload management with IBM Workload Scheduler for z/OS.

Batch command interface tool

A Batch Command Interface tool is supplied to perform some of the actions supplied by the PIF interface by means of a batch command interface. See “Batch command interface tool” for more information about how to use this tool. Refer to the EQQYCBAT sample for running this tool.

IBM Workload Scheduler for z/OS control language

About this task

The IBM Workload Scheduler for z/OS Control Language (OCL) is a REXX program, `EQQOCL`, delivered in the IBM Workload Scheduler for z/OS `SEQQMISC` library, that allows some of the IBM Workload Scheduler for z/OS data to be handled from a simple REXX-like interface. You use the tool by running `EQQOCL` as a compiled REXX program in a batch TSO environment. You can choose to run it in a separate job or as a step in your production JCL.

The tool is described in Chapter 4, “Control Language (OCL),” on page 163. The following members of IBM Workload Scheduler for z/OS sample libraries are also supplied to help you use the tool:

EQQYRJCL

A sample JCL to run the tool

EQQYRPRC

A sample procedure to invoke `EQQOCL`

EQQYRMSG

Contains the messages issued by the `EQQOCL` tool

EQQYRPRM

A sample input parameter member

The OCL tool requires the IBM® Compiler Libraries for REXX/370 Version 1.3.0, Program Number 5695-014. The Compiler Libraries must be installed on the system, and the TSO Compiler Programming Table, `IRXCMPTM`, must be customized and the Compiler Libraries modules must be made available to TSO. Refer to the Compiler Libraries for REXX/370 manuals for more information.

The OCL tool calls the `EQQSTOR` program, available in the IBM Workload Scheduler for z/OS sample library member `EQQRXSTG`. If the `WTO` function is used, OCL calls the `IPOWTO` program, available in the scheduler sample library member `EQQOCWTO`. If the `UPD` function is used, OCL calls the `EQQPIFT` program available in IBM Workload Scheduler for z/OS sample library member `EQQPFIJV`. This sample program requires the PL/I compiler and runtime libraries. The `EQQSTOR` program, and optionally `IPOWTO` and `EQQPIFT` programs, must be link-edited and made available to OCL at runtime.

Communicating with **EQQYCOM**

Requests to IBM Workload Scheduler for z/OS to perform particular actions are calls to `EQQYCOM`, using normal z/OS linkage conventions.

You must create a program that calls `EQQYCOM` and provide it with the necessary instructions, such as a parameter list, to enable IBM Workload Scheduler for z/OS to perform the required action. With each call to `EQQYCOM`, you can make one IBM Workload Scheduler for z/OS request.

`EQQYCOM` can be linked with the modules from which it is called, or it can be created as a separate load module and control passed to it using the link macro. If you create `EQQYCOM` as a separate load module and frequent calls are required, you should, for performance reasons, consider placing `EQQYCOM` in the link-pack area. All modules in the same job-step must be in an APF-authorized library. The first module loaded at the start of the job-step must also be link-edited with the APF-authorized attribute. In the TSO or TSO-batch environment, you need not have the PIF program authorized.

Details of your request to IBM Workload Scheduler for z/OS are a parameter list that you pass to EQQYCOM. Before passing control to EQQYCOM, you must load the address of your parameter list into general purpose register 1.

Note: If you want to run a PIF program from an IBM Workload Scheduler for z/OS dialog, ensure that your PIF program is invoked as a separate task. Otherwise, your dialog session will end when the PIF program has completed. For example, you can run a REXX exec that runs your PIF program using the ATTACH command.

Calling EQQYCOM from exits that are taken by the controller address space is not supported and will cause unpredictable results if attempted.

Required data sets

When you use the program interface, allocate the data sets identified by these ddnames to each address space where your program runs:

EQQMLIB

IBM Workload Scheduler for z/OS message library.

EQQMLOG

Data Set for messages from the program interface.

An extra message log is required for each additional INIT request made before a TERM request, or when PIF is invoked by a program or started task that already uses the EQQMLOG data set allocated to the caller address space. For detailed information, see "INIT request" on page 25.

Optional data set

EQQYPARM

Parameter file for specifying the INIT initialization statement. EQQYPARM must reference a sequential data set or a member of a partitioned data set whose logical record length is 80 bytes.

The //EQQYPARM DD * notation followed by INIT statements is not allowed and might cause a system X'0C4' abend.

Note:

1. It is important that you also allocate the IBM Workload Scheduler for z/OS diagnostic data set, EQQDUMP. Debugging information is written to this data set, for example, IBM Workload Scheduler for z/OS control blocks and traces.
2. If you plan to run PIF applications many times per day from a long-running non-TSO address space (for example, NetView[®]), to prevent a storage shortage do not specify the EQQYPARM ddname. Instead, specify the parameters either in the PIF application or in the controller INTFOPTS initialization statement. When you run a PIF application by specifying the EQQYPARM ddname, a TSO environment must be established each time and some of the resources remain allocated until the task ends. This might lead to a storage shortage, if the commands are issued many times.

Error messages

When an error occurs in a request, messages are always written to the message log data set allocated to the caller address space. The data set is either EQQMLOG or that specified in the MLOGDDN argument of the INIT request. In certain cases, messages are also written to the EQQMLOG data set allocated to the IBM Workload Scheduler for z/OS subsystem to which your requests are directed.

Errors related to the request itself (for example, a spelling error in a parameter argument) result in a message written only to the message log allocated to the caller address space.

Errors related to the IBM Workload Scheduler for z/OS subsystem (for example, an error detected by IBM Workload Scheduler for z/OS data validation) result in a brief message to the caller message log. A more detailed message about the error is written to the EQQMLOG allocated to the IBM Workload Scheduler for z/OS subsystem.

Parameter overview

The parameter list contains the necessary information for one request. Figure 1 on page 7 illustrates the basic structure of the parameter list and the addressing linkage to it.

The parameter list must always consist of seven fullwords, representing the seven parameter types outlined here. Not all parameters are required for some requests, in which case you must set the parameter value to hexadecimal zeros. A character-type parameter value that contains blanks also indicates that the parameter is omitted. The parameter list itself must not contain zeros.

Figure 1 on page 7 describes the parameter values that are referenced by the parameter address list.

An overview of the parameters follows. More detailed descriptions of the required parameters are given with the description of each request type.

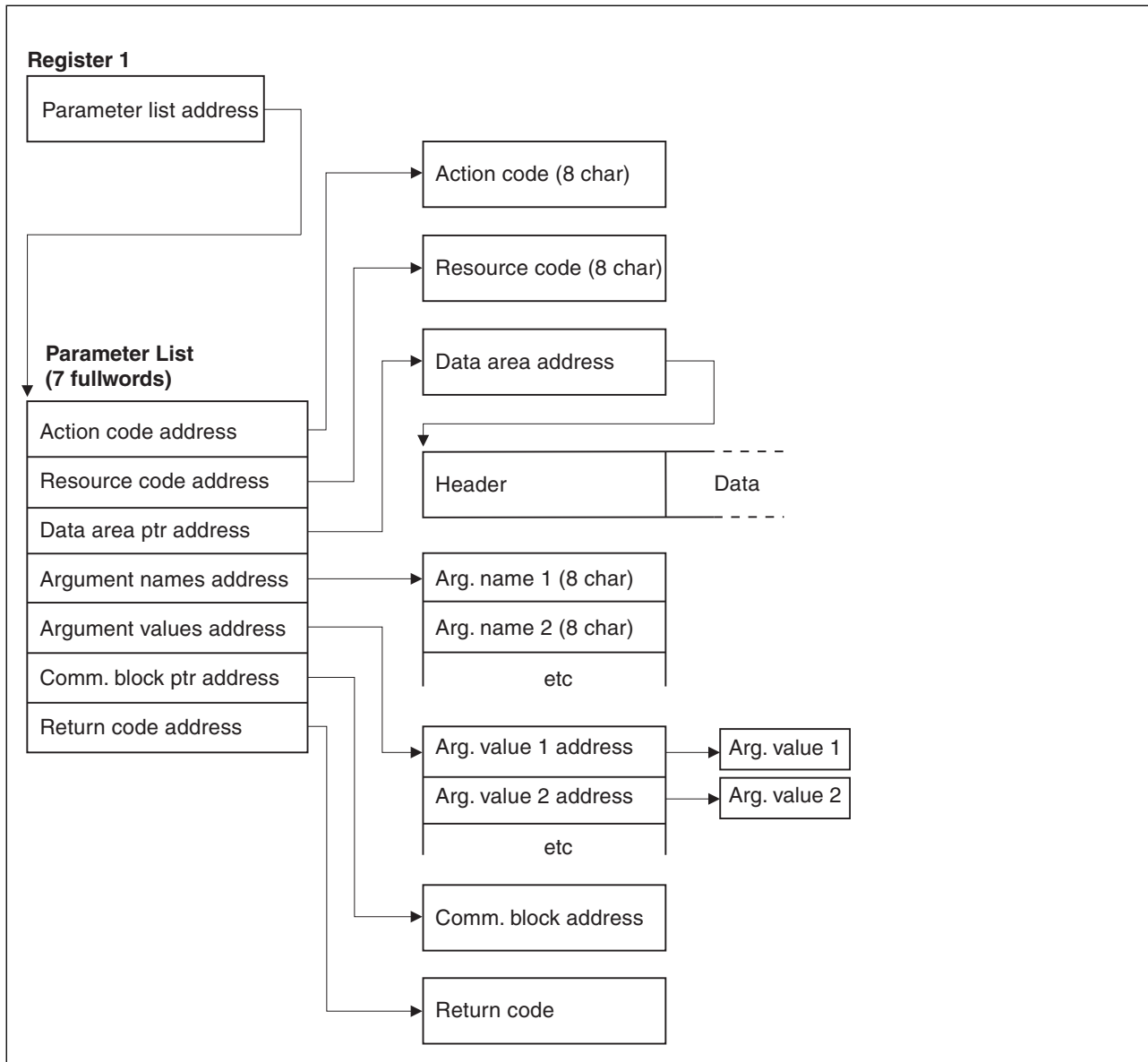


Figure 1. Program interface parameters

Action code

The first fullword in the parameter list is the address of the action code.

The action code describes the action to be performed. For example, to update a record in one of the IBM Workload Scheduler for z/OS databases, you use the REPLACE action code.

Resource code

The second fullword in the parameter list is the address of the resource code.

The resource code describes the IBM Workload Scheduler for z/OS resource that the request is directed to. For example, to replace an application description in the AD database, you use the AD resource code.

Data area

The third fullword in the parameter list is the address of a fullword that contains the address of a data area.

A data area consists of the actual data involved in the request. If you are retrieving information from a database, EQQYCOM places the record in this area and provides its address in the fullword whose address is in the parameter list.

Note: EQQYCOM might use the same piece of data area storage for successive data retrieval requests, overwriting the storage area used for the previous request each time. Therefore, your program must copy the information to its own storage area if it must be kept during later retrieval requests.

If you are writing information to a database, your program must build its own data area and provide its address in the fullword whose address is in the parameter list.

Attention: When the data area is not used, the data area address in the parameter list must be set to hexadecimal zero; failure to do so might cause unpredictable results. Some programming languages might require special coding to achieve this task; for example, in PL/I programs, use the SYSNULL built-in function.

The data area consists of a header, which describes the structure of the data record, and the data itself. “Data area description and format” on page 10 describes this in more detail.

Argument names and values

The fourth and fifth fullwords in the parameter list are the addresses of the argument name list and argument value address list.

The arguments provide specific information about your request. An argument can consist of an *argument name* alone, or an argument name and a matching *argument value*. Some requests require only one or more argument names, and some require argument names and values. If argument values are required, they are always associated one-for-one with the argument names.

Arguments can be compared to operands of a TSO command, where the argument name corresponds to the parameter keyword, and the argument value corresponds to the parameter value. For example:

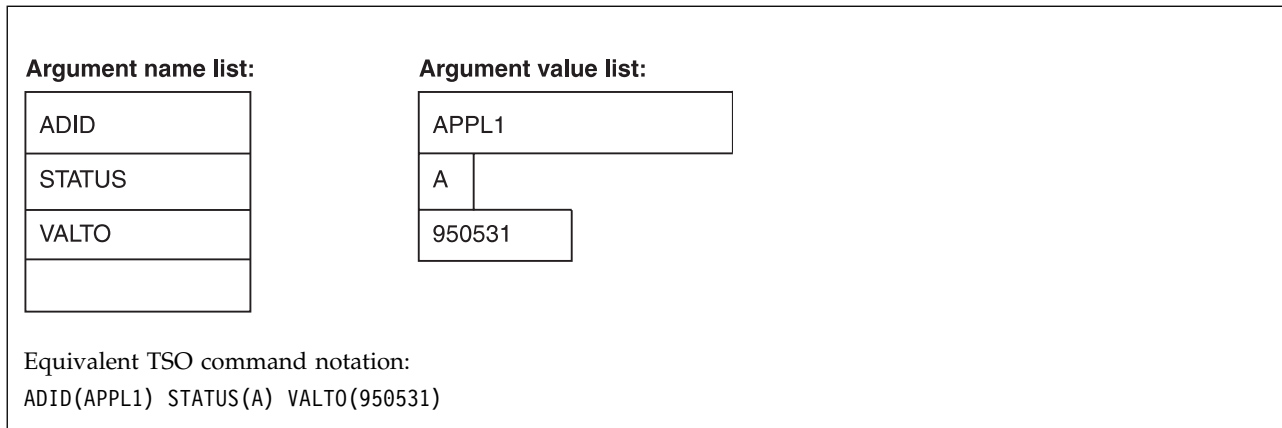


Figure 2. Program interface arguments in TSO command notation

The parameter list contains two addresses for the arguments, one pointing to the argument name list and one pointing to the argument value address list.

The argument name list is an array of 8-byte character fields. Each field contains an argument name, is left justified. Blanks must appear to the right of the argument name if it is shorter than 8 characters. The list is terminated by an all-blank field.

The argument value address list contains a list of addresses that point to the argument values. For a character argument value, the length of the field should be the same as that shown in the argument table. But, when a character argument is used as a selection argument, only the characters up to the first blank or comparison operator are used. Date and time data types are processed in the same way as character argument values. A numeric argument value must always be a fullword.

The retrieval of a record from the application description database is an example of how arguments are used. Here, the arguments identify the particular record required. The argument names identify the names of fields in the record, and the argument values identify the values of those fields for the record you want to retrieve (see Figure 2).

Sometimes, there might be a reason to specify the same argument more than once. For example, to get a list of active operations, you can specify argument name STATUS and C ≠ for the value, plus argument name STATUS and D ≠ for the value. You can specify an argument multiple times; up to 32 arguments can be defined in the argument name list.

Communication block

The sixth fullword in the parameter list is the address of a fullword containing the address of the communication block.

The first request to EQQYCOM must be an INIT request, which establishes a *communication session* between EQQYCOM and your program. During INIT request processing, EQQYCOM builds a *communication block* representing the session and returns its address in the fullword whose address is in the parameter list. The communication block address provided must remain unmodified during each subsequent call to EQQYCOM until the end of the session, so that EQQYCOM can identify the session that requests are coming from.

Return code

After each request, EQQYCOM provides a return code indicating if the request was successful or not.

The seventh fullword in the parameter list is the address of a fullword containing the return code. This return code is also placed in register 15.

Sequence of requests

About this task

Each communication session must always start with an INIT request and end with a TERM request. There can be several requests between them.

When modifying the current plan, requests must be made as follows:

1. With a series of requests, an *MCP block* is built containing all the necessary information required for one modification of the current plan.
2. With an EXECUTE request, information in the MCP block is used to actually update the current plan data set.

Also, when modifying the current plan, you can make a series of requests that refer to the same occurrence. The first request identifies the occurrence, and following requests can modify data related to that occurrence without needing to specifically identify it each time. The program interface remembers what the *current occurrence* is. Similarly, the program interface remembers the *current operation* and, once identified, a series of requests can be made that refer to it.

Other requests can be made in any sequence except where specifically noted. For example, you can produce a *list* of records with one request, which you can follow with one or more requests that *select* records from the list.

Data area description and format

About this task

Requests to EQQYCOM often involve either reading one or more records from an IBM Workload Scheduler for z/OS database or data set, or writing them. In both cases, the record is placed in a data area and its address provided in a fullword whose address is in the parameter list. When you are retrieving information, EQQYCOM places the required record in a data area and provides the address of this area. When you are writing information to an IBM Workload Scheduler for z/OS database or data set, your program must build its own data area and provide its address. Note that EQQYCOM might use the same piece of storage for data areas in successive data retrieval requests, overwriting the data area used for the previous request each time.

The data area consists of two parts:

- The header
- The data record

Header format

The header describes the segments in the record and their actual location within the record. The length and format of each segment type is fixed. For a description of the segments, see Appendix A, "Program interface record format," on page 271.

Note: For records retrieved with the SELECT request, the header always has a length that is a multiple of 32, with any unused header entries set to 00x. For records created for the INSERT and REPLACE requests, it is not necessary to set the header length to a multiple of 32, but if you do, you can use direct byte for byte comparison of input and output records.

The header consists of one or more header entries, each entry describing one segment in the data record. Each header entry is 16 bytes and consists of:

Segment name (8 characters)

A character field containing the name of a segment. If this field is blank, this is the last header entry in the header.

Offset to segment (1 fullword)

Offset to the start of this segment within the record from the start of the header. If this data area is from a LIST or SELECT request and it is the last header entry (segment name is blanks), this field contains more information about the request. This is further described under the detailed descriptions of the requests later in this chapter.

Reserved (4 bytes)

Reserved for use by IBM Workload Scheduler for z/OS.

The header is terminated by a header entry with a blank segment name. Figure 3 shows an example of a data area using an application description.

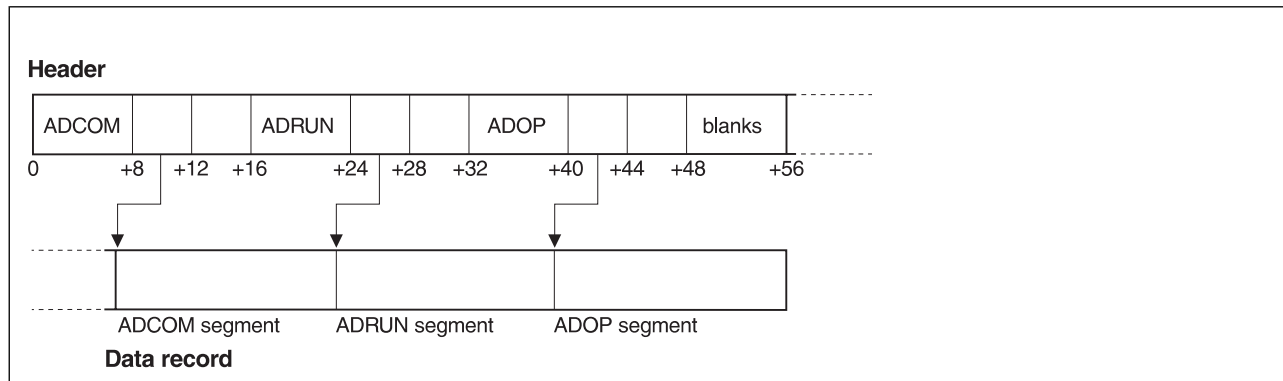


Figure 3. Program interface data area example

Data record format

Each data record handled by the program interface function consists of a *subset* of the complete IBM Workload Scheduler for z/OS record. Each record consists of the same fields that are available in the ISPF dialogs, in the same format. Yes/No fields are single character fields, which contain either Y or N. Integer values are fullword fields.

The amount of information in an IBM Workload Scheduler for z/OS record can vary enormously. For example, an application description can contain one run cycle and one operation, or it can contain many run cycles and many operations. The size of each record and its format can vary greatly. Because of this, the program interface function uses a *header* for each record. The header contains information about the record.

Each record consists of one or more *segments* representing different information in that record. For example, an application description consisting of one run cycle and

three operations is described by a record consisting of one run cycle segment and three operation segments. Also, one *common* segment always exists, which contains basic information, such as the application name, owner, and validity date. The common segment is always the first segment of the data record. Other segments can appear in any order except that segments that are logically related appear together. For example, in an application description record, the operation segments (ADOP) can appear in any order, but the dependency (ADDEP) and special resource segments (ADSR) always follow immediately after the ADOP to which they belong.

Date considerations

IBM Workload Scheduler for z/OS can handle dates up to 31 December 2071. This high date is the default for application description valid-to and run cycle out-of-effect dates when you use the IBM Workload Scheduler for z/OS dialogs.

Internal date representation

Internally, IBM Workload Scheduler for z/OS works with a two-digit year format, so dates are represented as 00 to 99. In order to handle dates before and after 2000, IBM Workload Scheduler for z/OS has chosen 72 as the base year. This means that, internally, 1972 is represented as 00, 1995 as 23, and 2071 as 99.

This internal date does not affect IBM Workload Scheduler for z/OS dialog and report users. They always see the real date. However, PIF requests often involve reading or writing records in an IBM Workload Scheduler for z/OS database. These records contain dates in the internal two-digit format with base year 72. You use the PIFCWB and PIFHD parameters of the INTFOPTS statement or the CWBASE and HIGHDATE parameters of the INIT statement to define how you want these dates to be presented to PIF applications.

PIFCWB and CWBASE values determine what base year IBM Workload Scheduler for z/OS uses when presenting dates to PIF applications. If you specify 00, dates are presented as the last two digits of the real date. For example, 1995 is presented as 95 and 2001 as 01. Note, however, that the PIFCWB and CWBASE parameters affect all dates *except* the default out-of-effect and valid-to dates. These dates are presented to PIF application as the value specified in the PIFHD and HIGHDATE parameters.

Refer to *IBM Workload Scheduler for z/OS: Customization and Tuning* for more details on these statements.

Date arguments in PIF applications

You might have PIF applications developed before year 2000 that use 991231 as the value of the VALTO argument to indicate the default valid-to date of the last version of the AD. The real default date is 31 December 2071. However, by using the PIFHD parameter of the INTFOPTS statement or the HIGHDATE parameter of the INIT statement to define the high date as 991231, you can use these existing PIF applications without updating them.

A good way to avoid specifying a specific date for default valid-to dates is to define the PIFHD keyword of the INTFOPTS statement or the HIGHDATE keyword of the INIT statement as a six-character string. IBM Workload Scheduler for z/OS will always interpret this as representing the default valid-to date.

Table 1 on page 13 gives an overview of the different date representations.

Table 1. Comparison of Date Representations

| Real date | Internal date | PIF date (base 00, high date 711231) | PIF date (base 72, high date 991231) |
|------------|---------------|--------------------------------------|--------------------------------------|
| 1994/06/15 | 220615 | 940615 | 220615 |
| 2004/06/15 | 320615 | 040615 | 320615 |
| 2071/12/31 | 991231 | 711231 | 991231 |

Updating application description run cycles with PIF

When you use the ISPF dialogs to update or create application descriptions, you specify a run cycle out-of-effect date. Then IBM Workload Scheduler for z/OS calculates the run cycle valid-to date by subtracting one day from the out-of-effect date. However, when you use PIF to update an AD you do not specify the out-of-effect date, you specify the valid-to date. Then IBM Workload Scheduler for z/OS calculates the out-of-effect date by adding one day. If you specify the valid-to date as the default high date, adding one day would make the date higher than the highest allowed date. Therefore, when you specify the valid-to date in a PIF application as the default high date, IBM Workload Scheduler for z/OS takes the IBM Workload Scheduler for z/OS high date as the out-of-effect date.

Security considerations

You need authorization to use many of the program interface requests. If you do not have authority for the request you need, give the relevant access type and RACF® resource code to your IBM Workload Scheduler for z/OS administrator. Table 2 describes the access authority you need:

Table 2. Access Authority for Program Interface Requests

| Program interface request | Access type required |
|---|----------------------|
| INIT OPTIONS RESET TERM | None |
| LIST SELECT | Read |
| DELETE EXECUTE INSERT MODIFY REPLACE SETSTAT | Update |

You need authorization to access these IBM Workload Scheduler for z/OS fixed resources:

Table 3. Program Interface Resources and the Corresponding IBM Workload Scheduler for z/OS Fixed Resources Used for Checking Authorization

| Program interface resource | IBM Workload Scheduler for z/OS fixed resource |
|-------------------------------|--|
| ADCOM, AD, ADEXT, ADKEY, ADRE | AD |

Table 3. Program Interface Resources and the Corresponding IBM Workload Scheduler for z/OS Fixed Resources Used for Checking Authorization (continued)

| Program interface resource | IBM Workload Scheduler for z/OS fixed resource |
|--|--|
| AW_SCL | W_SCL |
| CL, CLCOM | CL |
| CPEXT, CPST, CPOC, CPOCCOM, CPOP, CPOPCOM, CPWS, CPWSV, CPWSCOM, CPWSVCOM, IVL, VIVL, MCPBLK | CP |
| CSR, CSRCOM, CPOPSRU | SR |
| ETT | ETT |
| JCLV, JCLVCOM | JV |
| JS, JSCOM, JCLPREP, JCLPREPA, JL, JLCOM | JS |
| LTOC, LTOCCOM | LT |
| OI, OICOM | OI |
| PR, PRCOM | PR |
| RG, RGCOM | RG |
| SR, SRCOM | RD |
| WS, WSCOM, WSV, WSVCOM | WS |

For example, to list the intervals during which all workstations are closed, resource AW_SCL, you need READ access to the W_SCL fixed resource.

Running user-written programs compiled for older scheduler versions

Before you try to run a program compiled for a previous version of IBM Workload Scheduler for z/OS, the program OBJ must be compiled, or at least link-edited, for the current IBM Workload Scheduler for z/OS version.

Overview of request types

The requests that you can make to the program interface are summarized here. The requests are described in detail in the following sections and are arranged in alphabetical order.

DELETE

Deletes data items.

EXECUTE

Performs an actual update of the current plan.

INIT Initializes the communication session between your program and the IBM Workload Scheduler for z/OS subsystem.

INSERT

Inserts new data items or additional information into existing data items.

LIST Retrieves a list of data items of a specified type using generic search arguments.

MODIFY

Modifies data fields in the LTP or current plan, or identifies CP or LTP data items for further modification.

OPTIONS

Specifies options to be used when performing PIF requests. You can use these options to automatically resolve external dependencies when adding LTP or CP occurrences, improve the time taken to retrieve information about operations, request the address of the area where the message ID is returned, and to prevent messages being written to the message log.

REPLACE

Replaces an existing application description or operator instruction.

RESET

Cancels a series of modify current plan requests if performed before the EXECUTE request.

SELECT

Retrieves a single data item in detail.

SETSTAT

Modifies the status of a condition dependency. You can use it to change the condition status from undecided to true or false, if the original status is undecided because of missing step-end information.

TERM Terminates the communication session between your program and the IBM Workload Scheduler for z/OS subsystem.

Table 4. Records Using a Common Segment

| Arg names | Length |
|-------------|--------|
| ADCOM | 192 |
| AWSC | 80 |
| CLCOM | 96 |
| CPOCCOM | 428 |
| CPOPCOM (*) | 380 |
| CPOPSRU (*) | 96 |
| CSRCOM (*) | 240 |
| CPWSCOM (*) | 128 |
| CPWSVCOM(*) | 129 |
| ETT | 128 |
| JCLVCOM | 96 |
| JLCOM | 64 |
| JSCOM | 96 |
| LTOCCOM | 157 |
| OICOM | 96 |
| PRCOM | 96 |
| RGCOM | 160 |
| SRCOM | 204 |
| WSCOM | 128 |
| WSVCOM | 128 |

(*): You cannot specify this argument name to delete the entire record.

DELETE request

The DELETE request deletes a record or record segment. If you delete a record the arguments identify the particular record to be deleted. If you want to delete only some information in an occurrence (for example, one of the operations in an occurrence), you must first use a MODIFY request to identify the occurrence before you use the DELETE request for the operation. Similarly, if you want to delete a special resource specification or a current plan condition for an operation, you must use a MODIFY request to identify the occurrence and then use a MODIFY request to identify the operation, before using a DELETE for the special resource.

To delete an interval of a current plan workstation you must precede the DELETE IVL with a MODIFY CPWS to identify the workstation.

To delete the extended name of an operation you must use the MODIFY request. See "MODIFY CPEXT" on page 144 for details.

The DELETE request can be used to modify information in the current plan. All requests that cause a modification of the current plan require a later EXECUTE request for the modification to actually take effect.

Action code

DELETE

Resource code

The resource code specifies which record type or record segment you want to delete. You can specify these values:

AD Application description record
AWSCL All workstations closed record
CL Calendar record
CPCOND Current plan condition
CPOC Current plan occurrence record
CPOP Current plan operation record
CPPRE Current plan predecessor segment
CPSIMP Current plan condition dependency
CPSR Current plan special resource segment
CPSUC Current plan successor segment
CPUSRF Current plan user field segment
ETT Event triggered tracking criteria record
IVL Current plan workstation interval segment
JCLV JCL variable table record
JL JS file JOBLOG record
JS Job control language record
LTOC LTP occurrence record
LTCPRE LTP conditional predecessor segment
LTPRE LTP predecessor segment
OI Operator instruction record.

| | |
|------|---|
| PR | Period record |
| RG | Run cycle group record |
| SR | Special resource record |
| VIVL | Current plan virtual workstation destination interval segment |
| WS | Workstation description record |
| WSV | Virtual workstation destination record |

Data area

Not used.

Arguments

The arguments identify the particular record you want to delete. Two ways you can do this are:

- Specify field names of the record as argument names and specify the addresses of field values, to identify the particular record you want to delete. The values can be:
 - Character values. A blank character terminates the field.
 - Numeric values, which must occupy a fullword.

You must specify sufficient arguments to *uniquely* identify a record. You can use a comparison operator after the argument values. The default, *equals* (=), is assumed if you do not.

- Specify the record type as an argument name and the address of the previously retrieved common segment as the argument value address, if you have already retrieved the common segment of a record but you then want to delete the entire record. Table 4 on page 15 describes the record types that you can specify as argument names.

Note: The values of PIF arguments as dates depend on the PIF base year, which is defined by the PIFCWB keyword on the INTFOPTS statement, or the CWBASE keyword of the INIT statement. The value of the VALTO argument for default high date depends on the PIFHD keyword of the INTFOPTS statement or the HIGHDATE keyword of the INIT statement. See *IBM Workload Scheduler for z/OS: Customization and Tuning* for more details on these statements.

Delete AD arguments

Table 5. Delete AD Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|---|
| ADID | 16 | Char | Application description ID |
| GROUP | 8 | Char | Authority group name |
| GROUPDEF | 16 | Char | Group definition ID |
| OWNER | 16 | Char | Owner ID |
| PRIORITY | 4 | Integer | Priority |
| STATUS | 1 | Char | Status: P=Pending A=Active |
| TYPE | 1 | Char | Application type: A=Application G=Group Default is A |
| VALFROM | 6 | Char YYMMDD | Valid-from date |
| VALTO | 6 | Char YYMMDD | Valid-to date |

Note: IBM Workload Scheduler for z/OS assumes application type A, if you do not specify the TYPE argument name.

Delete AWSCL arguments

Table 6. Delete AWSCL Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|-------------|
| DATE | 6 | Char YYMMDD | Date |

Delete CL arguments

Table 7. Delete CL Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------|
| CALENDAR | 16 | Char | Calendar ID |

Delete CPCOND arguments

Note: Always identify an operation with a MODIFY CPOP request before a DELETE CPCOND request.

Table 8. Delete CPCOND Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| CONDID | 4 | Integer | Condition ID. Valid values are from 1 to 999. |

Delete CPOC arguments

Table 9. Delete CPOC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|-----------------------------|
| ADID | 16 | Char | Application description ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |

Delete CPOCPRE arguments

Table 10. Delete CPOCPRE Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|---|
| PREADID | 16 | Char | Predecessor application ID |
| PREIA | 10 | Char YYMMDDHHMM | Predecessor input arrival date and time |
| * | 1 | Char | Reserved |
| PREOPNO | 4 | Integer | Predecessor operation number |

Delete CPOCSUC arguments

Table 11. Delete CPOCSUC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|---------------------------------------|
| SUCADID | 16 | Char | Successor application ID |
| SUCIA | 10 | Char YYMMDDHHMM | Successor input arrival date and time |
| SUCOPNO | 4 | Integer | Successor operation number |

Delete CPOP arguments

Table 12. Delete CPOP Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|------------------|
| OPNO | 4 | Integer | Operation number |

Delete CPPRE arguments

Table 13. Delete CPPRE Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|---|
| PREADID | 16 | Char | Predecessor application ID |
| PREIA | 10 | Char YYMMDDHHMM | Predecessor input arrival date and time |
| PREMAND | 1 | Char | The predecessor is mandatory. The value can be Y or N (default). Specify Y if the predecessor is mandatory. |
| PREOPNO | 4 | Integer | Predecessor operation number |

Note: When deleting an internal predecessor, only specify PREOPNO. Specify all arguments to delete an external mandatory predecessor. Omit PREMAND if the predecessor is not mandatory.

Delete CPSIMP arguments

Note: Always identify an occurrence, an operation and a condition with:

- An INSERT or MODIFY CPOC request
- An INSERT or MODIFY CPOP request
- An INSERT or MODIFY CPCOND request

before a DELETE CPSIMP request.

Table 14. Delete CPSIMP Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| PREADID | 16 | Char | Predecessor application name |
| PREIA | 10 | Char | Predecessor application input arrival date and time |
| PREOPNO | 4 | Integer | Predecessor operation number |
| PROCSTEP | 8 | Char | Use it to define a step level dependency. If the step is not in a procedure, this parameter identifies the job step name, otherwise it identifies the step name in the JCL procedure. It must correspond to the name of an EXEC PGM= statement. |

Table 14. Delete CPSIMP Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| STEPNAME | 8 | Char | Use it in conjunction with PROCSTEP when defining a step level dependency, only if the step is in a procedure, to identify the name of a step that invokes an in-stream or cataloged procedure. It must correspond to the name of an EXEC PROC= statement. |
| TYPE | 2 | Char | Condition type: RC = To check the predecessor return code ST = To check the predecessor status |
| LOG | 2 | Char | Logical operator: GE = Greater than or equal to. Valid only for RC condition type. GT = Greater than. Valid only for RC condition type. LE = Less than or equal to. Valid only for RC condition type. LT = Less than. Valid only for RC condition type. EQ = Equal to. NE = Not equal to. Use it to specify conditions on final statuses only. RG = Range. |
| VALRC | 4 | Char | Return code value. |
| VALRC2 | 4 | Char | Return code value, as second boundary in a range expressed by the RG logical operator. |
| VALST | 1 | Char | Status, valid only for ST type |

Delete CPSR arguments

Table 15. Delete CPSR Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-----------------------|
| RESNAME | 44 | Char | Special resource name |

Delete CPSUC arguments

Table 16. Delete CPSUC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|--------------------|---------------------------------------|
| SUCADID | 16 | Char | Successor application ID |
| SUCIA | 10 | Char YYMMDDHHMM | Successor input arrival date and time |
| SUCOPNO | 4 | Integer | Successor operation number |

Note: When deleting an internal successor, only specify SUCOPNO. All three arguments must be specified to delete an external successor.

Delete CPUSRF arguments

Table 17. Delete CPUSRF Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-----------------|
| UFNAME | 16 | Char | User field name |

Note: When deleting a user field, only specify UFNAME. The corresponding user field value is also deleted.

Delete ETT arguments

Table 18. Delete ETT Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| ADID | 16 | Char | Associated application ID |
| ETTNAME | 44 | Char | Name of trigger |
| ETTYPE | 1 | Char | Type of trigger, 2 -> job 3 -> special resource |

Delete IVL arguments

An interval can have information originating from the workstation description, indicator CPIVLDP in segment CPIVL is set to Y, or else to N. If an interval is changed or created via the dialog or the program interface, the indicator CPIVLMOD in CPIVL is set to Y, or else to N.

DELETE IVL only affects modifications. Intervals with CPIVLDP=Y remain after a DELETE, the interval is reset to the daily planning values and CPIVLMOD is set to N. Intervals with CPIVLDP=N are fully deleted.

Table 19. Delete IVL Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|------------------------------|
| FROM | 10 | Char YYMMDDHHMM | Interval start date and time |

Delete JCLV arguments

Table 20. Delete JCLV Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-----------------------|
| JCLVTAB | 16 | Char | JCL variable table ID |

Delete JL arguments

Table 21. Delete JL Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|--------------------|-----------------------------|
| ADID | 16 | Char | Application ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| JOBNAME | 8 | Char | z/OS Job name |
| OPNO | 4 | Integer | Operation number |
| WSNAME | 4 | Char | Workstation name |

Delete JS arguments

Table 22. Delete JS, JSCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|--------------------|-----------------------------|
| ADID | 16 | Char | Application ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| JOBNAME | 8 | Char | z/OS Job name |
| OPNO | 4 | Integer | Operation number |
| WSNAME | 4 | Char | Workstation name |

Delete LTOC arguments

Table 23. Delete LTOC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|----------------------------|
| ADID | 16 | Char | Application description ID |
| IAD | 6 | Char YYMMDD | Input arrival date |
| IAT | 4 | Char HHMM | Input arrival time |

Delete LTCPRE arguments

Table 24. Delete LTCPRE Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|--|
| ADID | 16 | Char | Application description ID |
| IAD | 6 | Char YYMMDD | Input arrival date |
| IAT | 4 | Char HHMM | Input arrival time |
| PREADID | 16 | Char | Conditional predecessor application ID |
| PREIAD | 6 | Char YYMMDD | Conditional predecessor input arrival date |
| PREIAT | 4 | Char HHMM | Conditional predecessor input arrival time |

Delete LTPRE arguments

Table 25. Delete LTPRE Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|--------------------------------|
| ADID | 16 | Char | Application description ID |
| IAD | 6 | Char YYMMDD | Input arrival date |
| IAT | 4 | Char HHMM | Input arrival time |
| PREADID | 16 | Char | Predecessor application ID |
| PREIAD | 6 | Char YYMMDD | Predecessor input arrival date |
| PREIAT | 4 | Char HHMM | Predecessor input arrival time |

Note: DELETE LTPRE is used only to delete external predecessors. No support is provided in the long-term plan for internal dependencies.

Delete OI arguments

Table 26. Delete OI Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|------------------|
| ADID | 16 | Char | Application ID |
| OPNO | 4 | Integer | Operation number |

Note: To delete both the operator instruction and any associated temporary instructions, issue a LIST OICOM request followed by this loop:

1. A request with the OICOM segment as the argument
2. A SELECT OICOM with argument NEXT.

Continue the loop until SELECT OICOM NEXT gives a return code greater than 0.

Delete PR arguments

Table 27. Delete PR Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------|
| PERIOD | 8 | Char | Period name |
| PRTYPE | 1 | Char | Period type |

Delete RG arguments

Table 28. Delete RG Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--------------------------------|
| RGID | 8 | Char | Run cycle group ID |
| RGOWNER | 16 | Char | Run cycle group owner |
| RGCALEND | 16 | Char | Run cycle group calendar |
| RGVARTAB | 16 | Char | Run cycle group variable table |
| RUNNAME | 8 | Char | Run cycle name |
| RUNCAL | 16 | Char | Run cycle calendar |
| RUNVTAB | 16 | Char | Run cycle variable table |
| RUNSETID | 8 | Char | Run cycle subset ID |

Delete SR arguments

Table 29. Delete SR Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---------------------------|
| RESGROUP | 8 | Char | Special resource group ID |
| RESHIPER | 1 | Char | DLF resource indicator |
| RESNAME | 44 | Char | Special resource name |

Delete VIVL arguments

If an interval contains information originating from the Virtual Workstation Destination description, the indicator CPVIVLDP in segment CPVIVL is set to Y, otherwise it is set to N. If an interval is changed or created using the dialog or the program interface, the indicator CPVIVLMOD in segment CPVIVL is set to Y, otherwise it is set to N.

DELETE VIVL only affects modifications. Intervals with CPVIVLDP=Y remain after a DELETE, the interval is reset to the daily planning values and CPVIVLMOD is set to N. Intervals with CPVIVLDP=N are fully deleted.

Table 30. Delete VIVL Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|------------------------------|
| FROM | 10 | Char YYMMDDHHMM | Interval start date and time |

Delete WS arguments

Table 31. Delete WS Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| WSNAME | 4 | Char | Workstation name |
| WSREP | 1 | Char | Workstation reporting attribute |
| WSRETYPE | 1 | Char | Remote engine type: D = distributed, Z = z/OS or blank |
| WSTWS | 1 | Char | Y=fault tolerant workstation N=non-fault tolerant workstation |
| WSTYPE | 1 | Char | Workstation type |
| WSWAIT | 1 | Char | WAIT workstation, Y or N |

Delete WSV arguments

Table 32. Delete WSV Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---------------------------------|
| WSNAME | 4 | Char | Virtual workstation name |
| WSDEST | 8 | Char | Virtual workstation destination |

Communication block address

This is the address returned by INIT request processing, which must remain unmodified for all following requests.

Return code

When EQQYCOM returns control, this fullword shows the outcome of the request:

- 0 The request was successful.
- 4 The record; AD, AWSCL, CL, ETT, JCLV, JS, OI, PR, SR, WS, or WSV is currently being updated by another user. The record is not deleted.
- 8 The request was unsuccessful. An error message has been written to the message log data set.

EXECUTE request

The EXECUTE request causes an update of the current plan after one or more modify, insert, or delete current plan requests are completed.

If you are changing more than one current plan occurrence or current plan workstation before an EXECUTE request, you must complete all changes to one

occurrence or workstation before changing another. If you do not complete all changes to one occurrence or workstation a message is issued and all modifications since the last EXECUTE request are reset.

For changes to current plan resources, CSR, no EXECUTE is required.

Action code

EXECUTE

Resource code

MCPBLK

Data area

Not used.

Arguments

Not used.

Communication block address

This is the address returned by INIT request processing, which should remain unmodified for all following requests.

Return code

When EQQYCOM returns control, this fullword shows the outcome of the request:

- 0 The request was successful.
- 8 The request was unsuccessful. An error message has been written to the message log data set.

INIT request

The INIT request identifies the IBM Workload Scheduler for z/OS subsystem required and initializes the communication session between this subsystem and your program. It must always be the first request. The INIT request builds a communication block. EQQYCOM returns its address to your program.

Through the parameter file EQQYPARM, the user might override the subsystem name specified in the INIT request, and set an LU name, a host name and a port number for TCP/IP communication, a TRACE level, and the DATINT flag.

The parameter file can be a sequential file, or a PDS allocated as:

```
//EQQYPARM DD DISP=SHR,DSN=OPCESA.SYS1.CNTL(YPARM)
```

Action code

INIT

Resource code

The name of an active IBM Workload Scheduler for z/OS subsystem to which all following requests are directed.

Data area

Not used.

Arguments

You can specify arguments to:

- Determine if a recovery environment is established. The recovery environment consists of a SPIE exit routine and an ESTAE recovery routine, which, in case of error, dumps certain storage areas and terminates execution. You can specify argument name ESTAEI, ESTAER, LUNAME, or NOESTAE. Argument values are not required.
- Identify the message log to which that messages are written.
- Communicate the level of user program. Use the argument USRLEV (introduced by APAR PI57310 for compatibility reasons) to enable the dialog between the old user-written PIF programs and the new PIF delivered through the Small Programming Enhancements (SPE), which has caused changes in the segment layouts.

Argument name=ACCOID

The parameter that determines if the OI database is to be accessed when a LIST or SELECT request on CP operations is issued.

Argument value=*accoid*

A 1-byte character field for the accoid: valid values are Y or N. Y means that the OI database is read (this is the default). N means that the OI database is not read.

Argument name=DUBPROC

The parameter with which the BPX1SDD routine is invoked for the program interface TCP/IP session.

Argument value=*dubproc*

A 1-byte character field for the dubproc: valid values are Y or N. Y means that BPX1SDD is invoked by using DUBPROCESS parameter. N means that BPX1SDD is invoked by using DUBPROCESSDEFER parameter. The default is N.

Argument name=ESTAEI

The recovery environment is established at the INIT request. It remains in effect until the TERM request. This is the default.

Argument name=ESTAER

The recovery environment is established and terminated for each individual request. This might be needed if, for example, your program has a recovery environment dependent on the setting of a certain register, as in PLI.

Argument name=LUNAME

This argument allows the user to specify a server or controller LU name for the program interface session.

Argument value=*luname*

A 17-byte field for the LU name address, ending by a blank if shorter than 17 bytes.

Argument name=MLOGDDN

This argument identifies a message log that messages are written to, rather than the default message log, EQQMLOG.

Each INIT request requires its own message log. If you make more than one INIT request before a TERM request, or if PIF is invoked by a program or started task that is already using EQQMLOG, specify MLOGDDN for each additional INIT request. If MLOGDDN is not specified, and EQQMLOG is already in use, message EQQZ038E is written to the SYSLOG and the INIT request fails.

Argument value=ddname

An 8-character field, left justified, which identifies the ddname of the data set that messages are written to.

Argument name=NOESTAE

No recovery environment is established.

Argument name=REMHOST

The server host name for the program interface TCP/IP session. REMHOST and LUNAME are mutually exclusive.

Argument value=server hostname

A 52-byte field for the host name address, ending by a blank if shorter than 52 bytes.

Argument name=REMPORT

The server port number for the program interface TCP/IP session. REMHOST and LUNAME are mutually exclusive.

Argument value=server port number

A 4-byte integer field for the port number: valid values are from 0 to 65535.

Argument name=USRLEV

Introduced by APAR PI57310 for compatibility reasons, this argument communicates to EQQYCOM the level of the user program. If not specified, the programming interface assumes that the program invoking the PIF is an old version, and no recompile is done to see segments changes. In this situation, PIF uses old layouts to communicate with old user program.

Argument value=*n*

Identifies the level of the Small Programming Enhancement (SPE) applied. The valid values are:

- 9 Identifies the enablement of SPE changes done through the APAR PI57310.
- 10 Identifies the enablement of SPE changes done through the APARs PI62520 and PI62521.

Communication block address

When EQQYCOM returns control to your program, this contains the address of the communication block representing this program interface session. Ensure that this address remains unmodified during all following calls to EQQYCOM. The initial value of this field is not important, because it will be overlaid with the communication block address by EQQYCOM.

Return code

When EQQYCOM returns control, this fullword indicates the outcome of the request:

- 0 The request was successful. A program interface session has been successfully started. The address of the communication block has been placed in the parameter list.
- 8 The request was unsuccessful. Check the message log, SYSLOG, and EQQDUMP data sets for error information.

INSERT request

The INSERT request writes a new record or record segment to an IBM Workload Scheduler for z/OS database or data set. This can be done in several ways:

- To insert new application descriptions, operator instructions, JCL records, new all workstations closed, calendar, ETT, period, special resource or workstation record, your program must provide the complete record to be inserted in the data area. Arguments are not used.
 - To insert a new application occurrence into the current plan, you can:
 - Provide a complete *application description* record in the data area. This is then converted by IBM Workload Scheduler for z/OS into a current plan occurrence. Here, the arguments can be used to provide the input arrival and deadline date and time.
- Or
- Select an existing application description from the database to be added as an occurrence into the current plan. Here, the arguments are used to identify the existing application description from which the occurrence will be created. The arguments can also specify occurrence-related information such as input arrival time and deadline time. The data area is not used.

When inserting an application occurrence into the long-term plan, the name of the application description must be supplied through the argument parameters. You cannot supply an application description through the data area. The data area pointer address must be set to zero before your program call.

When inserting a new occurrence using either of the previous methods, the input arrival date and time and deadline date and time can be provided in the arguments. If the input arrival is not provided when inserting a current plan occurrence, the current date and time is used (that is, the date and time at which the occurrence is inserted). However, if an occurrence already exists with this application ID and input arrival date and time, the next available minute in which no occurrence of this application exists will be used. You must supply an input arrival date and time if you are inserting an occurrence in the LTP.

If arguments are not provided for the deadline, these defaults are observed by IBM Workload Scheduler for z/OS:

- If the occurrence is being added to the current plan and the input arrival is provided, the deadline from the first run cycle is used if a run cycle exists. If there are no run cycles or the input arrival is not provided, the deadline is set to the input arrival time plus 8 hours.
- When the occurrence is being added to the long-term plan, the deadline is set to the input arrival plus 8 hours.

By default, external dependencies of the occurrence are not resolved when it is added to the LTP or current plan. If resolution of external dependencies is required, the OPTIONS request must be used to specify this.

- To insert the extended name of an operation you must use the MODIFY request. See “MODIFY CPEXT” on page 144 for details.
- To insert new information into an existing LTP or current plan occurrence, you use the arguments to provide all the necessary information. For example, you

can insert a new operation into an existing current plan occurrence. But the actual occurrence to which the information is to be added must have been identified by a previous MODIFY or INSERT request. Similarly, you can insert new information for an existing current plan operation, provided that the operation has been identified. This means you must first use a MODIFY request to identify the occurrence and then use a MODIFY request to identify the operation, before inserting a predecessor (CPPRE), successor (CPSUC), or special resource (CPSR).

When identified, the program interface maintains a *current occurrence* and *current operation*.

If you want to insert a new interval into a current plan workstation you must first identify the workstation with a MODIFY CPWS request.

The arguments are used to specify all required information. The data area is not used.

The INSERT request can be used to modify information in the current plan. All requests that cause a modification of the current plan require a later EXECUTE request for the modification to take effect.

Action code

INSERT

Resource code

The resource code specifies which record type or record segment you want to insert. You can specify these values:

AD Application description record

AWSC

All workstations closed record

CL Calendar record

CPCOND

Current plan condition

CPOC Current plan occurrence record

CPOP Current plan operation record

CPPRE

Current plan predecessor segment

CPSAI

System automation information for the current plan operation

CPSIMP

Current plan condition dependency

CPSR Current plan special resource segment

CPSUC

Current plan successor segment

CPUSRF

Current plan user field segment

ETT Event triggered tracking criteria record

IVL Current plan workstation interval

JCLPREP

Promptable setup variables for the current operation

JCLV JCL variable table record

JS Job control language record

LTOC LTP occurrence record

LTPRE

LTP predecessor segment

OI Operator instruction record

| | |
|------|---|
| PR | Period record |
| RG | Run cycle group record |
| SR | Special resource record |
| VIVL | Current plan virtual workstation destination interval segment |
| WS | Workstation record |
| WSV | Virtual workstation destination record |

Data area

The data area is used in these situations:

- If you are inserting new application descriptions, operator instructions, JCL records, new all workstations closed, calendar, ETT, period, special resource, or workstation records, provide the complete record in the data area.
- If you are inserting new current-plan occurrences, specify the application ID as an argument and specify that no data area is available. If it is necessary to supply the application description via the data area, omit the application ID argument and give the application description via the data area.

Arguments

The arguments are used in these situations:

- If you are inserting a new current-plan occurrence of an existing application description, use the arguments to identify the application rather than having to provide the complete record yourself. The arguments tell IBM Workload Scheduler for z/OS which application is required, and it handles the insertion of this application as an occurrence record in the LTP or CP. The arguments can also provide additional information, such as input arrival time, deadline, and priority. If you use the arguments, set the data area pointer address to zero before you issue your call.

If you are inserting a new LTP or current plan occurrence, use the arguments to identify the application.

- If you are inserting a new current-plan occurrence and providing the application description information in the data area, the arguments can specify occurrence information, such as input arrival time, deadline, and priority. These arguments override the values in the application description.
- If you are inserting information for an existing LTP or current plan occurrence or operation, use the arguments to provide all the information to be inserted.

Note:

1. No arguments can be provided for AD, OI, JS, AWSCL, CL, ETT, PR, SR, WS, and WSV resource codes, because the complete record must be in the data area.
2. When inserting calendar records (CL) the standard day segment (that is, ='STANDARD') must appear as the second segment in the input field right after the CLCOM segment. Its corresponding interval segment must be immediately after.
3. When inserting special resource (SR), DAY=8 represents the *STANDARD* day.
4. The format of the duration used in the data area in Insert AD/WS will be defined by the DURSEC option, described in the paragraph "OPTIONS request" on page 59.

Note: The values of PIF arguments as dates depend on the PIF base year, which is defined by the PIFCWB keyword on the INTFOPTS statement, or the CWBASE keyword of the INIT statement. The value of the VALTO argument for default high date depends on the PIFHD keyword of the INTFOPTS statement or the

HIGHDATE keyword of the INIT statement. See *IBM Workload Scheduler for z/OS: Customization and Tuning* for more details on these statements.

Note: If the argument DURATION is used with the argument EDUR, an error message occurs.

Insert CPOC arguments

When you are inserting a current plan occurrence, the ADID argument is required unless you are providing the entire application description in the data area. The ADID argument identifies an existing application description, an occurrence of which is to be inserted into the current plan. All remaining arguments are optional and provide more information about the occurrence.

Table 33. Insert CPOC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|-----------------------------|
| ADID | 16 | Char | Application description ID |
| DEADLINE | 10 | Char YYMMDDHHMM | Deadline date and time |
| DESC | 24 | Char | Descriptive text |
| ERRCODE | 4 | Char | Error code |
| GROUP | 8 | Char | Authority group |
| GROUPDEF | 16 | Char | Group definition ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| JCLVTAB | 16 | Char | JCL variable table |
| ODESC | 24 | Char | Descriptive text of owner |
| OWNER | 16 | Char | Owner ID |
| PRIORITY | 4 | Integer | Priority |

Notes:

1. A DEADLINE argument is accepted also when no IA argument is specified. If the IBM Workload Scheduler selected IA is later than the DEADLINE argument value, the argument value is ignored. The default, IA plus 8 hours, is used instead.
2. If you specify 24.00 as the IA time, it is converted to 00.00 of the following day. In fact, the valid input arrival times are 00.00 through 23.59.
3. If you specify as deadline 00.00, it is converted to 24.00 of the previous day. In fact, the valid deadline times are 00.01 through 24.00.

Insert CPOCPRE arguments

Table 34. Insert CPOCPRE Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|-----------------------------|
| PREADID | 16 | Char | Application ID |
| PREIA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| PREOPNO | 4 | Integer | Operation number |
| TRPTTIME | 4 | Integer HHMM | Tansport time |

Insert CPOCSUC arguments

Table 35. Insert CPOCSUC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|-----------------------------|
| SUCADID | 16 | Char | Application ID |
| SUCIA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| SUCOPNO | 4 | Integer | Operation number |

Insert CPCOND arguments

Note: Always identify an operation with an INSERT or MODIFY CPOP request before an INSERT CPCOND request.

Table 36. Insert CPCOND Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| CONDID | 4 | Integer | Condition ID. Valid values are from 1 to 999. |
| COUNT | 4 | Integer | Condition counter. Use it to define the rule type: 0 = All the condition dependencies, in the corresponding INSERT CPSIMP list, must be true $n > 0$ = At least n out of the specified condition dependencies must be true The default is 0. |
| DESC | 16 | Char | Descriptive text |

Insert CPOP arguments

Table 37. Insert CPOP Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| AEC | 1 | Char | Automatic error completion |
| AJR | 1 | Char | Automatic job hold/release |
| ASUB | 1 | Char | Automatic job submission |
| CLATE | 1 | Char | Cancel if late |
| CLNTYPE | 1 | Char | Data Set cleanup type |
| CONDRJOB | 1 | Char | Conditional recovery job |
| CSCRIPT | 1 | Char | Uses centralized script |
| DEADWTO | 1 | Char | Issue deadline WTO |
| DESC | 24 | Char | Descriptive text |
| DURATION | 4 | Integer | Estimated duration in 100th of a second |
| EDUR | 4 | Char HHMM | Estimated duration |
| EXPJCL | 1 | Char | Expanded JCL option |
| FORM | 8 | Char | Form number or blanks |
| HRC | 4 | Integer | Highest successful return code |

Table 37. Insert CPOP Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|--------------------|--|
| JCLASS | 1 | Char | Job class |
| JOBCRT | 1 | Char | Critical job. P=Critical path target W=Eligible for WLM assistance N=Not eligible for WLM assistance |
| JOBNAME | 8 | Char | Job name |
| JOBPOL | 1 | Char | Workload monitor late job policy. ' ' (blank) = default L = Long duration D = Deadline S = Latest start time C = Conditional mode |
| MONITOR | 1 | Char | Y=Operation monitored by an external product N=Operation not monitored by an external product |
| OPDL | 10 | Char YYMMDDHHMM | Operation deadline date and time or blank |
| OPIA | 10 | Char YYMMDDHHMM | Operation input arrival date and time or blank |
| OPNO | 4 | Integer | Operation number |
| PSUSE | 4 | Integer | Parallel servers required |
| R1USE | 4 | Integer | Resource 1 required |
| R2USE | 4 | Integer | Resource 2 required |
| RERUT | 1 | Char | Reroutable operation |
| RESTA | 1 | Char | Restartable operation |
| STATUS | 1 | Char | Operation status |
| TIMEDEP | 1 | Char | Time-dependent job |
| USERDATA | 16 | Char | Information stored in operation user data |
| USRSYS | 1 | Char | User sysout support |
| WSNAME | 4 | Char | Workstation name |
| WLMSCLS | 8 | Char | WLM service class |

Insert CPPRE arguments

Table 38. Insert CPPRE Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|-----------------------------|
| PREADID | 16 | Char | Application ID |
| PREIA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| PREOPNO | 4 | Integer | Operation number |
| TRPTTIME | 4 | Integer HHMM | Transport time |

Note: When CPPRE is needed to insert an internal dependency, only PREOPNO and TRPTTIME arguments are valid.

Insert CPSAI arguments

Table 39. Insert CPSAI Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| AUTFUNC | 8 | Char | System Automation automated function (for operation). It must be an alphanumeric value, uppercase format. The first character cannot be numeric. |
| COMMETXT | 255 | Char | System Automation command text. It must be set and cannot be blank. |
| COMPINFO | 64 | Integer | System Automation completion information |
| SECELEM | 8 | Char | System Automation security element |

Note:

1. The occurrence and operation to which the system automation information refers are identified, respectively, by the INSERT CPOC and INSERT CPOP sequences
2. You can use the insert CPSAI only if the operation runs on an automation workstation.

Insert CPSIMP arguments

Note: Always identify an occurrence, an operation and a condition with:

- An INSERT or MODIFY CPOC request
- An INSERT or MODIFY CPOP request
- An INSERT or MODIFY CPCOND request

before an INSERT CPSIMP request.

Table 40. Insert CPSIMP Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| PREADID | 16 | Char | Predecessor application name |
| PREIA | 10 | Char | Predecessor application input arrival date and time |
| PREOPNO | 4 | Integer | Predecessor operation number |
| PROCSTEP | 8 | Char | Use it to define a step level dependency. If the step is not in a procedure, this parameter identifies the job step name, otherwise it identifies the step name in the JCL procedure. It must correspond to a step specifying the EXEC PGM= statement. |
| STEPNAME | 8 | Char | Use it in conjunction with PROCSTEP when defining a step level dependency, only if the step is in a procedure, to identify the procedure invocation step name. |

Table 40. Insert CPSIMP Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| TYPE | 2 | Char | Condition type: RC = To check the predecessor return code ST = To check the predecessor status |
| LOG | 2 | Char | Logical operator: GE = Greater than or equal to. Valid only for RC condition type. GT = Greater than. Valid only for RC condition type. LE = Less than or equal to. Valid only for RC condition type. LT = Less than. Valid only for RC condition type. EQ = Equal to. NE = Not equal to. Use it to specify conditions on final statuses only. RG = Range. |
| VALRC | 4 | Char | Return code value. |
| VALRC2 | 4 | Char | Return code value, as second boundary in a range expressed by the RG logical operator. |
| VALST | 1 | Char | Status, valid only for ST type |

Note: To create an internal dependency, do not specify either PREADID or PREIA.

Insert CPSR arguments

Table 41. Insert CPSR Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| ONCOMPL | 1 | Char | Action on complete Y N R |
| ONERROR | 1 | Char | Keep on error Y N |
| QUANTITY | 4 | Integer | Quantity required. Specify 0 to allocate the total quantity of the special resource. The value 0 is the same as blank in the dialogs. |
| RESNAME | 44 | Char | Special resource name |
| RESUSAGE | 1 | Char | Special resource usage S X |

Insert CPSUC arguments

Table 42. Insert CPSUC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|-----------------------------|
| SUCADID | 16 | Char | Application ID |
| SUCIA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| SUCOPNO | 4 | Integer | Operation number |

Note: When CPSUC is needed to insert an internal dependency, only the SUCOPNO argument is valid.

Insert CPUSRF arguments

Note: Always identify an operation with an INSERT or MODIFY CPOP request before an INSERT CPUSRF request.

Table 43. Insert CPUSRF Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|------------------|
| UFNAME | 16 | Char | User field name |
| UFVALUE | 54 | Char | User field value |

Insert IVL arguments

An interval can have information originating from the workstation description, indicator CPIVLDP in segment CPIVL is set to Y, or otherwise to N. If an interval is changed via the dialog or the program interface then the indicator CPIVLMOD is Y, or otherwise N

INSERT IVL can insert an interval spanning existing intervals with CPIVLMOD=N. The inserted interval will be converted to several intervals as required by daily planning. Other requests following the INSERT must take this possible split into account; each request is handled fully before the next request.

Table 44. Insert IVL Arguments

| Arg names | Len | Data type | Description |
|-----------|-----|-----------------|----------------------------|
| FROM | 10 | Char YYMMDDHHMM | Interval start date/time |
| PSCAP | 4 | Integer | Parallel server capacity |
| R1CAP | 4 | Integer | Resource 1 capacity |
| R2CAP | 4 | Integer | Resource 2 capacity |
| TO | 10 | Char YYMMDDHHMM | Interval end date and time |

Insert JCLPREP arguments

Table 45. Insert JCLPREP Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|-----------------------------|
| ADID | 16 | Char | Application ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| OPNO | 4 | Integer | Operation number |

Note: "JCL preparation using PIF" on page 80 describes JCL preparation using the program interface.

Insert JCLV arguments

Table 46. Insert JCLV Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-----------------------|
| JCLVTAB | 16 | Char | JCL variable table ID |

Insert LTOC arguments

Table 47. Insert LTOC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|--------------------|------------------------|
| ADID | 16 | Char | Application ID |
| DEADLINE | 10 | Char YYMMDDHHMM | Deadline date and time |
| ERRCODE | 4 | Char | Error code |
| GROUPDEF | 16 | Char | Group definition ID |
| IAD | 6 | Char YYMMDD | Run date |
| IAT | 4 | Char HHMM | Input arrival time |
| JCLVTAB | 16 | Char | JCL variable table |
| PRIORITY | 4 | Integer | Priority |

Insert LTPRE arguments

Table 48. Insert LTPRE Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|--------------------|
| ADID | 16 | Char | Application ID |
| IAD | 6 | Char YYMMDD | Run date |
| IAT | 4 | Char HHMM | Input arrival time |
| PREADID | 16 | Char | Application ID |
| PREIAD | 6 | Char YYMMDD | Run date |
| PREIAT | 4 | Char HHMM | Input arrival time |

Note: INSERT LTPRE is used only to insert external predecessors. No support is provided in the long-term plan for internal dependencies.

Insert VIVL arguments

If an interval contains information originating from the workstation description, the indicator CPVIVLDP in segment CPVIVL is set to Y, otherwise it is set to N. If an interval is changed using the dialog or the program interface then the indicator CPVIVLMOD in segment CPVIVL is set to Y, otherwise it is set to N.

INSERT VIVL can insert an interval spanning existing intervals with CPVIVLMOD=N. The inserted interval will be converted to several intervals as required by daily planning. Other requests following the INSERT must take this possible split into account; each request is completed before the next request.

Table 49. Insert VIVL Arguments

| Arg names | Len | Data type | Description |
|-----------|-----|-----------------|----------------------------|
| FROM | 10 | Char YYMMDDHHMM | Interval start date/time |
| PSCAP | 4 | Integer | Parallel server capacity |
| R1CAP | 4 | Integer | Resource 1 capacity |
| R2CAP | 4 | Integer | Resource 2 capacity |
| TO | 10 | Char YYMMDDHHMM | Interval end date and time |

Communication block address

About this task

This is the address returned by INIT request processing, which should remain unmodified for all following requests.

Return code

When EQQYCOM returns control, this fullword shows the outcome of the request:

- 0 The request was successful.
- 4 One or more of the dependencies, specified by the application description of the INSERT LTOC request, was not set up because no applicable predecessor occurrence exists. This return code could also result from an INSERT request for any of LTPRE, CPOP, CPOC, CPPRE, and CPSR, if the dependency was not set up.
- 8 The request was unsuccessful. An error message has been written to the message log data set.

LIST request

The LIST request retrieves a list of records from the specified database or data set. The first entry in the list is made available for processing. Other records in the list can be retrieved using the SELECT request.

When you use the LIST request, the resulting list consists only of the common segments of the records. Appendix A, "Program interface record format," on page 271 describes the data fields that make up the common segment of each record. If you want to retrieve a complete record, you must use the SELECT request.

After a successful LIST request for a particular resource code, the list remains available until you build a new list for the same resource code, or until a TERM request. This means that you can have several active lists if required, but only one at a time for each resource code.

When retrieving current plan occurrences and operations, the default is to retrieve all matching objects except those in deleted status. When STATUS is provided as an argument, the specified selection overrides the default processing.

In case of large amount of data, the use of queries without filter argument might exceed any available storage and needs to be limited. For program interface applications, invoked by a clist and IKJEFT01, a test allocation is done finding out how much storage is available (between a minimum of 32KB and a maximum of 64MB) and thereafter a fraction (1/4) of it is allocated to receive the unknown amount of data from the IBM Workload Scheduler subsystem.

Action code

LIST

Resource code

The resource code identifies the record type the list will comprise. You can specify these values:

ADCOM

Application description common segment

| | |
|----------------------|---|
| ADKEY | Application description key segment |
| AWSCL | All workstations closed |
| CLCOM | Calendar common segment |
| CPEXT | Current plan operation extended name segment |
| CPCONDCO | Current plan condition common segment |
| CPOC, CPOCCOM | Current plan occurrence |
| CPOPCOM | Current plan operation common segment |
| CPOPSRU | Current plan operation segment with information about the operation in relation to a special resource |
| CPWSCOM | Current plan workstation common segment |
| CPWSVCOM | Current plan virtual workstation common segment |
| CSRCOM | Current plan special resource common segment |
| ETT | Event triggered tracking criteria |
| GENDAYS | Run dates generated by run cycle rule segment |
| JCLVCOM | JCL variable table common segment |
| JLCOM | Job log common segment |
| JSCOM | JCL common segment |
| LTOCCOM | LTP occurrence common segment |
| OICOM | Operator instruction common segment |
| PRCOM | Period common segment |
| RGCOM | Run cycle group common segment |
| RGKEY | Run cycle group key segment |
| SRCOM | Special resource common segment |
| WSCOM | Workstation description common segment. |
| WSVCOM | Virtual workstation destination common segment |

Data area

When EQQYCOM returns control to your program after a successful LIST request, this fullword contains the address of a data area containing the first record from the requested list. Only the common segment of the requested record is provided when you use the LIST request. Appendix A. Program Interface Record Format describes the fields in each record common segment.

The header section for this record contains, besides the normal header information, a field containing a count of the number of elements in the list. This count field is in the final header entry, that is, the entry that has a blank segment-name field. The count is stored in the field that normally contains the segment offset. A complete description of headers can be found under “Header format” on page 10.

Note: The resource code JSCOM retrieves JCL records from the JCL repository data set and not from a JCL library. But a SELECT request tries to get JCL records from a JCL library if they are not found in the JCL repository data set.

Arguments

Argument names specify field names of the record to be tested to determine if the record should be included in the list.

For each argument name specified, a corresponding argument value must be specified. The argument value you specify is compared with values in the actual database records to determine if the record should be included in the list.

Argument values can be:

- Character values. Any number of characters terminated by a blank or comparison operator. Character values can be specified generically, using asterisks and percent signs as masking characters. An asterisk (*) can be used in place of any number of characters or a null string. A percent sign (%) can be used in place of exactly one character.

Note:

1. Because the first blank or comparison-operator symbol ends the argument value, you cannot search for fields that contain imbedded blanks or comparison-operator symbols.
 2. Generic search arguments, * and %, cannot be used in the year part (YY) of date arguments.
- Numeric values, which must occupy a fullword.

A comparison operator can follow the argument value, either with or without an intervening blank. The record is included in the list if:

- = The argument value is equal to the record value.
- ≠ The argument value is not equal to the record value.
- > The argument value is greater than the record value.
- >= The argument value is greater than or equal to the record value.
- < The argument value is less than the record value.
- <= The argument value is less than or equal to the record value.

If no comparison operator is supplied, equals (=) is assumed.

Note:

1. When you want to use a comparison operator (such as <, >, or ≠) in an argument, and the argument contains an IA value that includes a date and time, supply the full value as the argument value. The comparison operator can follow this value.
2. To prevent unpredictable results when the system assigns an area that was just freed from a previous request, remember to do the following:
 - a. GETMAIN an area size of one additional byte to the length of the specific argument's request.
 - b. Insert a blank character at the end of the argument value.

To clarify what unpredictable results could take place, consider the following sequence in a PIF request:

- a. GETMAIN 27 bytes (to store ADID, IA, and the > (greater than) operator)
- b. LIST request
- c. FREEMAIN
- d. GETMAIN 26 bytes (to store ADID,IA)
- e. SELECT request

As shown in this sequence, if the GETMAIN assigned to the SELECT request is the same as the one of the LIST request, the > operator is still present in the SELECT storage and this can originate unwanted results.

For example, if the current plan contains such occurrences as:

| | | | |
|--------|----------|-------|---|
| AAAAAA | 98/01/21 | 08.00 | C |
| AAAAAA | 98/01/22 | 07.00 | C |
| BBBBBB | 98/01/22 | 08.00 | C |
| BBBBBB | 98/01/23 | 08.00 | C |

and you want to list all occurrences whose IA value is greater than the first IA value, you must supply '9801210800<' as the argument value. Alternatively, if you want to list all the occurrences whose application name is greater than the first application name, for example, you can supply a string of any number of characters terminated by a comparison operator, such as 'AAA<='. You can thus use the comparison operators in different ways, depending on the type of data you use as the argument.

The comparison operators do not work with the generic search arguments.

Argument name: MATCHTYP

This argument can have the following values:

- EXA
- PFX
- SFX

With argument MATCHTYP specified, characters * and % are treated as normal characters instead of as generic matching characters, and blank as a normal character instead of ending the selection value. MATCHTYP EXA, PFX, and SFX affect:

- The STATUS and UFVALUE arguments of the CPOPCOM segment
- The ETTNAME argument of the ETT segment
- The RESNAME argument of the SRCOM and CSRCOM segments

If the MATCHTYP argument is specified, characters *, %, blank and comparison operators in a STATUS/ETTNAME/RESNAME argument, values are treated as normal characters.

When MATCHTYP is specified together with RESNAME or ETTNAME, the selection value must be padded with blanks up to the full resource name length of 44 characters. When RESNAME or ETTNAME are specified without MATCHTYP then the selection value is treated in the same way as any other selection value: it will be truncated at the first blank.

Note:

1. If MATCHTYP has the EXA value specified, then a record is selected only if the value in the record is exactly the same as the argument value.

2. If MATCHTYP has the PFX value specified, then a record is selected only if the start value in the record is the same as the argument value.
3. If MATCHTYP has the SFX value specified, then a record is selected only if the end value in the record is the same as the argument value.

The argument names and values that you can use to select records with a LIST request, are now described for each resource code.

Note: The values of PIF arguments as dates depend on the PIF base year, which is defined by the PIFCWB keyword on the INTFOPTS statement, or the CWBASE keyword of the INIT statement. The value of the VALTO argument for default high date depends on the PIFHD keyword of the INTFOPTS statement or the HIGHDATE keyword of the INIT statement. See *IBM Workload Scheduler for z/OS: Customization and Tuning* for more details on these statements.

List ADCOM, ADKEY arguments

Table 50. List ADCOM and ADKEY Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|--|
| ADID | 16 | Char | Application description ID |
| ADRULEP | 8 | Char | Name of period or run cycle group |
| GROUP | 8 | Char | Authority group name |
| GROUPDEF | 16 | Char | Group definition ID |
| MONITOR | 1 | Char | Y=application with at least one operation monitored by an external product N=application with no operation monitored by an external product |
| OWNER | 16 | Char | Owner ID |
| PRIORITY | 4 | Integer | Priority |
| STATUS | 1 | Char | Status: P=Pending A=Active |
| TYPE | 1 | Char | Application type: A=Application G=Group Default is A |
| VALFROM | 6 | Char YYMMDD | Valid-from date |
| VALTO | 6 | Char YYMMDD | Valid-to date |

Note:

1. The VALTO argument value depends on the PIFHD keyword of the INTFOPTS statement, or the HIGHDATE keyword of the INIT statement. See *IBM Workload Scheduler for z/OS: Customization and Tuning*.
2. IBM Workload Scheduler for z/OS assumes application type A, if you do not specify the TYPE argument name.
3. The ADSAI segment is retrieved only if the system automation information is defined for the selected application.

List AWSCL arguments

Table 51. List AWSCL Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|-------------|
| DATE | 6 | Char YYMMDD | Date |

List CLCOM arguments

Table 52. List CLCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------|
| CALENDAR | 16 | Char | Calendar ID |

List CPCONDCO arguments

Table 53. List CPCONDCO Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| ADID | 16 | Char | Application description ID |
| IA | 10 | Char | Input arrival date and time |
| OPNO | 4 | Integer | Operation number |
| CONDID | 4 | Integer | Condition ID. Valid values are from 1 to 999. |
| CONDVAL | 1 | Char | Final condition status: U = Undefined T = True F = False |

List CPOC, CPOCCOM arguments

Table 54. List CPOC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|--|
| ADID | 16 | Char | Application description ID |
| GROUP | 8 | Char | Authority group |
| GROUPDEF | 16 | Char | Group definition ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| MCPADDED | 1 | Char | MCP added, Y or N |
| MONITOR | 1 | Char | Y=occurrence with at least one operation monitored by an external product N=occurrence with no operation monitored by an external product |
| OWNER | 16 | Char | Owner ID |
| PRIORITY | 4 | Integer | Priority |
| RERUN | 1 | Char | Rerun requested, Y or N |
| STATUS | 1 | Char | Occurrence status |

Note: By default, occurrences in deleted status are not retrieved when the STATUS argument is not supplied. If you do not provide the STATUS argument, the request is processed as STATUS ≠ DELETED. When the STATUS argument is specified, its value can be W, S, C, E, U, D.

List CPOPCOM arguments

Table 55. List CPOPCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|--|
| ADID | 16 | Char | Application description ID |
| CLNSTAT | 1 | Char | Data Set cleanup status |
| CLNTYPE | 1 | Char | Data Set cleanup type |
| CONDRJOB | 1 | Char | Conditional recovery job |
| DPREM | 1 | Char | Removable by DP |
| ERRCODE | 4 | Char | Error code |
| EXECDEST | 8 | Char | Execution destination. To indicate a local destination, specify ***** |
| EXPJCL | 1 | Char | Expanded JCL option |
| EXTNAME | 54 | Char | Operation extended name |
| EXTSE | 16 | Char | Scheduling Environment name |
| GROUP | 8 | Char | Authority group |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time of the occurrence |
| JOBCRT | 1 | Char | Critical job: P=Critical path target W=Eligible for WLM assistance N=Not eligible for WLM assistance |
| JOBNAME | 8 | Char | Job name |
| JOBPOL | 1 | Char | Workload monitor late job policy. ' ' (blank) = default L = Long duration D = Deadline S = Latest start time C = Conditional mode |
| MONITOR | 1 | Char | Y=operation monitored by an external product N=operation not monitored by an external product |
| OPNO | 4 | Integer | Operation number |
| OWNER | 16 | Char | Owner ID |
| PRIORITY | 4 | Integer | Priority |
| SHADOWJ | 1 | Char | Shadow job, Y or N |
| STATUS | 1 | Char | Operation status |
| UFNAME | 16 | Char | User field name |
| UFVALUE | 54 | Char | User field value |
| UNEXPRC | 1 | Char | Y=Unexpected RC is ON N=Unexpected RC is OFF |
| USRSYS | 1 | Char | User sysout support |
| VIRTDEST | 8 | Char | Submission destination. To indicate a local destination, specify ***** |

Table 55. List CPOPCOM Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| WAITFORW | 1 | Char | Started on WAIT workstation, Y or N |
| WAITSE | 1 | Char | Waiting for Scheduling Environment, N or Y |
| WLMSCLS | 8 | Char | WLM service class |
| WMPRED | 1 | Char | Waiting for mandatory pending predecessors, Y or N |
| WPMPRED | 1 | Char | Waiting for either mandatory pending or pending predecessors, Y or N |
| WPPRED | 1 | Char | Waiting for pending predecessors, Y or N |
| WSNAME | 4 | Char | Workstation name |

Note:

1. By default, operations in deleted status are not retrieved when the STATUS argument is not supplied. If you do not provide the STATUS argument, the request is processed as STATUS ≠ DELETED.
2. The ADSAI segment is retrieved only if the system automation information is defined for the selected application.

List CPOPSRU arguments

Table 56. List CPOPSRU Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|------------------------------|
| LISTTYPE | 5 | Char | Type of list, INUSE or WAITQ |
| RESNAME | 44 | Char | Special resource name |

Note: Both arguments are required. The argument value specified for RESNAME is the name of the special resource for which the In-Use list or Wait Queue is to be retrieved. Generic characters are not supported. It is processed as if MATCHTYP EXA was specified; exact match is required for record selection. The argument MATCHTYP is NOT supported.

List CPWSCOM arguments

Table 57. List CPWSCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------------------------------|
| WSAUTO | 1 | Char | Automation workstation, Y or N |
| WSNAME | 4 | Char | Workstation name |
| WSREP | 1 | Char | Workstation reporting attribute |
| WSRETYPE | 1 | Char | Remote engine type (Z, D, or blank) |
| WSTWS | 1 | Char | Fault-tolerant workstation, Y or N |
| WSTYPE | 1 | Char | Workstation type |
| WSVIRT | 1 | Char | Virtual workstation, Y or N |
| WSWAIT | 1 | Char | WAIT workstation, Y or N |

Table 57. List CPWSCOM Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------------------------|
| WSZCENTR | 1 | Char | z-centric workstation, Y or N |

List CPWSVCOM arguments

Table 58. List CPWSVCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| WSNAME | 4 | Char | Virtual workstation name |
| WSDEST | 8 | Char | Virtual workstation destination. To indicate a local destination, specify ***** |

List CSRCOM arguments

Table 59. List CSRCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| RESALCS | 1 | Char | If any operation is currently allocating the resource shared, Y or N |
| RESAVAIL | 1 | Char | Whether or not the resource is available, Y or N |
| RESGROUP | 8 | Char | Resource group name |
| RESHIPER | 1 | Char | Whether or not it is a DLF control resource, Y or N |
| RESNAME | 44 | Char | Resource name |
| RESWAIT | 1 | Char | Whether or not any operation is waiting for the resource. |

Note: All the arguments are optional. The argument MATCHTYP is supported.

List ETT arguments

Table 60. List ETT Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---------------------------|
| ADID | 16 | Char | Associated application ID |
| ETTNAME | 44 | Char | Name of trigger |
| ETTTYPE | 1 | Char | Type of trigger |

List GENDAYS arguments

The LIST GENDAYS PIF call generates run dates for a run cycle that is provided in input by using a particular structure. The request is not linked to a job stream. It only uses calendar and periods definitions.

Table 61. List GENDAYS Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---------------|
| CALENDAR | 16 | Char | Calendar ID |
| FDAYRULE | 1 | Char | Free day rule |

Table 61. List GENDAYS Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|---------------------------------------|
| FROM | 6 | Char YYMMDD | One day before the Out of effect date |
| IAT | 4 | Char HHMM | Input arrival time |
| RULEDEF | * | Structure | Rule definition |
| TO | 6 | Char YYMMDD | In effect date |

Note:

- The earliest value for FROM is the first day of the current month in a year four years previous to the current year.
- The latest value for FROM is the first day of January in a year seven years after the current year.
- The latest value for TO is the 31st of December in a year seven years after the current year.

For example, if the current date is 13/09/23, then: 090901 < FROM < 200101 and FROM < TO < 201231.

- RULEDEF is made up by a structure similar to the one used for a rule definition in ADRUN. The first four bytes declare the length of the rule, while the remaining bytes are the rule text, which is preceded by the ADRULE keyword. For example:

```
Dcl 1 ruledef,
  2 rulelen bin(31),
  2 ruletxt char(30);
```

```
rulelen = 30;
ruletxt = 'ADRULE EVERY DAY(FRIDAY) YEAR ';
```

- Your PIF program need to first run the LIST request, followed by a loop of SELECT (NEXT) on the GENDAYS resource (no SEQn support).

List JCLVCOM arguments

Table 62. List JCLVCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-----------------------|
| JCLVTAB | 16 | Char | JCL variable table ID |

List JLCOM arguments

Table 63. List JLCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|-----------------------------|
| ADID | 16 | Char | Application ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| JOBNAME | 8 | Char | z/OS job name |
| OPNO | 4 | Integ | Operation number |
| WSNAME | 4 | Char | Workstation name |

List JSCOM arguments

Table 64. List JSCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|-----------------------------|
| ADID | 16 | Char | Application ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| JOBNAME | 8 | Char | z/OS job name |
| OPNO | 4 | Integer | Operation number |
| WSNAME | 4 | Char | Workstation name |

List LTOCCOM arguments

Table 65. List LTOCCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|---------------------|
| ADID | 16 | Char | Application ID |
| GROUP | 8 | Char | Authority group |
| GROUPDEF | 16 | Char | Group definition ID |
| IAD | 6 | Char YYMMDD | Run date |
| IAT | 4 | Char HHMM | Input arrival time |
| OWNER | 16 | Char | Owner ID |

List OICOM arguments

Table 66. List OICOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|------------------|
| ADID | 16 | Char | Application ID |
| OPNO | 4 | Integer | Operation number |

List PRCOM arguments

Table 67. List PRCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------|
| PERIOD | 8 | Char | Period name |
| PRTYPE | 1 | Char | Period type |

List RGCOM, RGKEY arguments

Table 68. List RGCOM, RGKEY Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--------------------|
| RGID | 8 | Char | Run cycle group ID |

Note: To list all the records of the run cycle group, run:

1. LIST RGKEY to obtain the first record and the total number of records in the run cycle group.
2. A loop of SELECT RGKEY next to list all the other records.

List SRCOM arguments

Table 69. List SRCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---------------------------|
| RESGROUP | 8 | Char | Special resource group ID |
| RESHIPER | 1 | Char | DLF resource indicator |
| RESNAME | 44 | Char | Special resource name |

List WSCOM arguments

Table 70. List WSCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| WSAUTO | 1 | Char | Automation workstation, Y or N |
| WSNAME | 4 | Char | Workstation name |
| WSREP | 1 | Char | Workstation reporting attribute |
| WSRETYPE | 1 | Char | Remote engine type: D = distributed, Z = z/OS or blank |
| WSTWS | 1 | Char | Fault-tolerant workstation, Y or N |
| WSTYPE | 1 | Char | Workstation type |
| WSVIRT | 1 | Char | Virtual workstation, Y or N |
| WSWAIT | 1 | Char | WAIT Workstation, Y or N |
| WSZCENTR | 1 | Char | z-centric workstation, Y or N |

List WSVCOM arguments

Table 71. List WSVCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| WSNAME | 4 | Char | Virtual workstation name |
| WSDEST | 8 | Char | Virtual workstation destination. To indicate a local destination, specify ***** |

Communication block address

This is the address returned by INIT request processing, which must remain unmodified for all following requests.

Return code

When EQQYCOM returns control, this fullword shows the outcome of the request:

- 0 The request was successful. Either all the data are returned or an incomplete list if the message EQQG009W is issued.
- 4 The request was unsuccessful, for one of these reasons:
 - The requestor is not authorized to read the records.
 - No records meet the criteria specified by the arguments.
- 8 The request was unsuccessful. An error message has been written to the message log data set.

MODIFY request

The MODIFY request modifies one or more fields in an LTP or current plan record. The arguments can be used both to identify the record to be modified, and to provide new values for this record. Or, the arguments can be used just to identify a record, and later requests can be used to perform particular actions. For example, with a MODIFY request, you can identify a particular current plan occurrence record. Then, with later INSERT requests, you can insert new operation records for that occurrence.

The MODIFY request can be used to modify information in the current plan. Requests that cause a modification of the current plan, except CSR requests, require a later EXECUTE request for the modification to actually take effect.

Action code

MODIFY

Resource code

CPCOND

Current plan condition segment

CPEXT

Current plan operation extended name

CPOC Current plan occurrence

CPOP Current plan operation

CPREND

Distributed remote job info

CPRENZ

z/OS remote job info

CPSAI

System automation information for the current plan operation

CPUSRF

User field information for the current plan operation

CPWS Current plan workstation

CPWSV

Current plan virtual workstation destination

CSR Current plan special resource

IVL Current plan workstation open interval

LTOC LTP occurrence

VIVL Current plan virtual workstation destination open interval

Data area

Not used.

Arguments

With the arguments described here, you specify the names and values of fields, either to identify a particular record, or provide updated information for a record.

Note: The values of PIF arguments as dates depend on the PIF base year, which is defined by the PIFCWB keyword on the INTFOPTS statement, or the CWBASE keyword of the INIT statement. The value of the VALTO argument for default high date depends on the PIFHD keyword of the INTFOPTS statement or the HIGHDATE keyword of the INIT statement. See *IBM Workload Scheduler for z/OS: Customization and Tuning* for more details on these statements.

Modify CPCOND arguments

When you are modifying an existing current plan condition, the CONDID argument is required to identify the condition to be modified. All remaining arguments are optional and provide the information used to modify the condition.

Note: Always identify an operation with an INSERT or MODIFY CPOP request before a MODIFY CPCOND request.

Table 72. Modify CPCOND Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| CONDID | 4 | Integer | Condition ID. Valid values are from 1 to 999. |
| COUNT | 4 | Integer | Condition counter. Use it to define the rule type: 0 = All the condition dependencies, in the corresponding INSERT CPSIMP list, must be true $n > 0$ = At least n out of the specified condition dependencies must be true The default is the current value. |
| DESC | 16 | Char | Descriptive text |

Modify CPEXT arguments

Create or modify the extended name of an operation in the current plan

Table 73. Modify CPEXT Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| EXTNAME | 54 | Char | Operation extended name. To delete the operation extended name, enter blanks between single quotation marks or EXTNAME=. |
| EXTSE | 16 | Char | Scheduling Environment name. Special characters are allowed. To delete the SE name, enter blanks between single quotation marks or EXTSE=. |

Modify CPOC arguments

When you are modifying an existing current plan occurrence, the ADID and IA arguments identify the occurrence to be modified. All remaining arguments provide the information used to modify the occurrence. The only valid values for the STATUS argument are W (Waiting) and C (Complete).

Table 74. Modify CPOC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|----------------------------|
| ADID | 16 | Char | Application description ID |

Table 74. Modify CPOC Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|--|
| ALLMON | 1 | Char | Y=all operations of occurrence monitored by an external product N=all operations of occurrence not monitored by an external product |
| DEADLINE | 10 | Char YYMMDDHHMM | Deadline date and time |
| ERRCODE | 4 | Char | Error code |
| GROUPDEF | 16 | Char | Group definition ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| IANEW | 10 | Char YYMMDDHHMM | New input arrival date and time |
| JCLVTAB | 16 | Char | JCL variable table |
| PRIORITY | 4 | Integer | Priority |
| STATUS | 1 | Char | Occurrence status |

Modify CPOP arguments

When you are modifying an existing current plan operation, the OPNO argument is required to identify the operation to be modified. All remaining arguments are optional and provide the information used to modify the operation. If you are inserting, modifying, or deleting a predecessor connection or special resource specification for the operation, the MODIFY CPOP request is required only to identify the operation that will be referred to in the following INSERT, MODIFY, or DELETE request. Then, only the OPNO argument is required.

Note: Always identify an occurrence with a MODIFY CPOC request before a MODIFY CPOP request.

Table 75. Modify CPOP Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| AEC | 1 | Char | Automatic error completion |
| AJR | 1 | Char | Automatic job hold/release |
| ASUB | 1 | Char | Automatic job submission |
| CLATE | 1 | Char | Cancel if late |
| CLNTYPE | 1 | Char | Data Set cleanup type |
| CONDRJOB | 1 | Char | Conditional recovery job |
| DEADWTO | 1 | Char | Issue deadline WTO |
| DESC | 24 | Char | Operation descriptive text |
| DURATION | 4 | Integer | Estimated duration in 100th of second |
| EDUR | 4 | Char HHMM | Estimated duration |
| ERRCODE | 4 | Char | Error code Note: You cannot change the error code if the operation runs on a fault tolerant workstation and is in error status. |
| EXPJCL | 1 | Char | Expanded JCL option |

Table 75. Modify CPOP Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|--------------------|--|
| FORM | 8 | Char | Form number or blanks |
| HRC | 4 | Integer | Highest successful return code |
| JCLASS | 1 | Char | Job class |
| JOBCRT | 1 | Char | Critical job: P=Critical path target W=Eligible for WLM assistance N=Not eligible for WLM assistance |
| JOBNAME | 8 | Char | Job name |
| JOBPOL | 1 | Char | Workload monitor late job policy: ' ' (blank) = default L = Long duration D = Deadline S = Latest start time C = Conditional mode |
| MONITOR | 1 | Char | Y=Operation monitored by an external product N=Operation not monitored by an external product |
| OPCMD | 2 | Char | Operation command: BD = Bind shadow job EX = Execute operation KJ = Kill operation ¹ KR = Kill recovery job ² MH = Hold operation MR = Release operation NP = NOP operation PN = Prompt reply no PY = Prompt reply yes UN = Un-NOP operation Note: 1. Applies only to operations running on IBM Workload Scheduler for z/OS Agents or on distributed workstations that are directly connected to the end-to-end server (OPCMaster). 2. Applies only to recovery jobs running on distributed workstations that are directly connected to the end-to-end server (OPCMaster). |
| OPDL | 10 | Char YYMMDDHHMM | Operation deadline date and time or blank |
| OPIA | 10 | Char YYMMDDHHMM | Operation input arrival date and time or blank |
| OPNO | 4 | Integer | Operation number |
| PSUSE | 4 | Integer | Parallel servers required |
| R1USE | 4 | Integer | Resource 1 required |
| R2USE | 4 | Integer | Resource 2 required |

Table 75. Modify CPOP Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| RERUT | 1 | Char | Reroutable operation |
| RESTA | 1 | Char | Restartable operation |
| STATUS | 1 | Char | Operation status |
| TIMEDEP | 1 | Char | Time-dependent job |
| USERDATA | 16 | Char | Information stored in operation user data |
| USRSYS | 1 | Char | User sysout support |
| WLMSCLS | 8 | Char | WLM service class |
| WSNAME | 4 | Char | Workstation name |

Note: If the argument DURATION is used with the argument EDUR, an error message occurs.

Modify CPREND arguments

Table 76. Modify CPREND Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------------------------------|
| COMPBNDF | 1 | Char | Complete if bind fails option (Y N) |
| REJOBNM | 40 | Char | Remote job name |
| REJSNM | 16 | Char | Remote job stream name |
| REJSWS | 16 | Char | Remote job stream workstation |

Note:

1. The occurrence and operation to which the remote job information refers are identified, respectively, by the INSERT and/or MODIFY CPOC (ADID, IA) and INSERT and/or MODIFY CPOP (OPNO) sequences.
2. You can use modify CPREND only if the operation runs on an remote engine workstation.
3. When you run MODIFY CPOP to modify the workstation type from remote engine to any other type, the remote job info related to the operation are automatically deleted.

Modify CPRENZ arguments

Table 77. Modify CPRENZ Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------------------------------|
| COMPBNDF | 1 | Char | Complete if bind fails option (Y N) |
| READID | 16 | Char | Remote application name |
| REOPNO | 4 | Integer | Remote operation number |

Note:

1. The occurrence and operation to which the remote job information refers are identified, respectively, by the INSERT and/or MODIFY CPOC (ADID, IA) and INSERT and/or MODIFY CPOP (OPNO) sequences.

2. You can use modify CPRENTZ only if the operation runs on an remote engine workstation.
3. When you run MODIFY CPOP to modify the workstation type from remote engine to any other type, the remote job info related to the operation are automatically deleted.

Modify CPSAI arguments

Table 78. Modify CPSAI Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| AUTFUNC | 8 | Char | System Automation automated function (for operation). It must be an alphanumeric value, uppercase format. The first character cannot be numeric. |
| COMMETXT | 255 | Char | System Automation command text. It must be set and cannot be blank. |
| COMPINFO | 64 | Integer | System Automation completion information. |
| SECELEM | 8 | Char | System Automation security element. It must be set and cannot be blank. |

Note:

1. The occurrence and operation to which the system automation information refers are identified, respectively, by the MODIFY CPOC (ADID, IA) and MODIFY CPOP (OPNO) sequences.
2. You can use modify CPSAI only if the operation runs on an automation workstation.

Modify CPUSRF arguments

When you are modifying an existing current plan user field, the UFNAME argument is required to identify the user field to be modified. The UFVALUE argument provides the information used to modify the user field.

Note: Always identify an operation with an INSERT or MODIFY CPOP request before a MODIFY CPUSRF request.

Table 79. Modify CPUSRF Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------------|
| UFNAME | 16 | Char | User field name. |
| UFVALUE | 54 | Char | User field value. |

Modify CPWS arguments

When you are modifying a current plan workstation, the WSNAME argument is required; it identifies the workstation. The remaining arguments contain the modified information.

Table 80. Modify CPWS Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| ALTWS | 4 | Char | When the workstation is set to failed or offline then another workstation can be specified for rerouting. Specify ALTWS if operations should be rerouted; if ALTWS is not supplied then no rerouting takes place. Note: For virtual workstations, the argument is ignored. |
| PSC | 1 | Char | Control on parallel server. Note: For virtual workstations, the argument is ignored. |
| R1C | 1 | Char | Control on resource 1. Note: For virtual workstations, the argument is ignored. |
| R2C | 1 | Char | Control on resource 2. Note: For virtual workstations, the argument is ignored. |
| STARTACT | 1 | Char | Action to be taken on current plan operations that have a status of started when the workstation status is set to failed or offline. Values are restart (R), set to error (E), or leave operation as is (L). Note: If the STARTACT argument is omitted when a workstation is set to failed or offline then no action is performed on the operations, as though STARTACT L was specified. |
| STATUS | 1 | Char | New status of active (A), failed (F), or offline (O). Note: The status failed is valid only for non-fault tolerant workstations. |
| WSLNK | 1 | Char | L=Workstation Linked (only FTW) U=Workstation Unlinked (only FTW) Blank=For non-fault tolerant WS |
| WSNAME | 4 | Char | Workstation name |
| WSREP | 1 | Char | Workstation reporting attribute. Note: For virtual workstations, the argument is ignored. |

Modify CPWSV arguments

When you are modifying a current plan virtual workstation, the WSNAME and WSDEST arguments are required; they identify the virtual workstation destination. The remaining arguments contain the modified information.

Table 81. Modify CPWSV Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| WSNAME | 4 | Char | Virtual workstation name |
| WSDEST | 8 | Char | Virtual Workstation destination. To indicate a local destination, specify ***** |

Table 81. Modify CPWSV Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| PSC | 1 | Char | Control on parallel server |
| R1C | 1 | Char | Control on resource 1 |
| R2C | 1 | Char | Control on resource 2 |
| STARTACT | 1 | Char | Action to be taken on current plan operations that have a status of started when the workstation status is set to failed or offline. Values are restart (R), set to error (E), or leave operation as is (L). Note: If the STARTACT argument is omitted when a workstation is set to failed or offline then no action is performed on the operations, as though STARTACT L was specified. |
| STATUS | 1 | Char | New status of active (A), failed (F), or offline (O) |

Modify CSR arguments

MODIFY CSR takes as selection argument the resource name in the RESNAME argument. This argument is required. The resource name must be padded to the full length of 44 characters. The special resource name cannot start with a quote since it will be removed from the first position, if present, during argument parsing. It is processed as if MATCHTYP EXA was specified and an exact match is required for record selection. Alternatively, the common segment CSRCOM can be given as the selection argument. Remaining arguments are optional and contain modifications.

Table 82. Modify CSR Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| DEFAVAIL | 1 | Char | Default availability, N or Y |
| DEFQTY | 4 | Integer | Default quantity, 1 to 999999 |
| MAXLIMIT | 4 | Integer | Maximum usage limit. From 0 (no limit) to 999999. |
| MAXTYPE | 1 | Char | Type of action when maximum usage limit is reached: Y N R |
| ONCOMPL | 1 | Char | Action on complete Y N R |
| ONERROR | 2 | Char | Action on error, F, FX, FS, K, or blank |
| QUANTITY | 4 | Integer | Override quantity, numeric 1 to 999999, or 0 to indicate that there is no overriding quantity. |
| RESAVAIL | 1 | Char | Override availability, N, Y, or blank to indicate there is no overriding availability |
| RESDEVIA | 4 | Integer | Deviation, -999999 to 999999. |
| RESNAME | 44 | Char | Resource name |
| USEDFOR | 1 | Char | Used for C, P, B, or N |

Note: MATCHTYP is NOT supported.

Modify IVL arguments

When you are modifying a workstation open interval, the FROM argument is required to identify the interval to be modified. All remaining arguments are optional and provide the information used to modify the open interval.

Note: Always identify a workstation with a MODIFY CPWS request before a MODIFY IVL request.

Table 83. Modify IVL Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|----------------|--|
| ALTWS | 4 | Char | Workstation to take over if this one fails or is set offline |
| FROM | 10 | Char YMMDDHHMM | Interval start date and time |
| PSCAP | 4 | Integer | Parallel server capacity |
| R1CAP | 4 | Integer | Resource 1 capacity |
| R2CAP | 4 | Integer | Resource 2 capacity |

Modify LTOC arguments

When you are modifying an existing LTP occurrence, the ADID, IAD, and IAT arguments identify the occurrence to be modified. All remaining arguments provide the information used to modify the occurrence.

Table 84. Modify LTOC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|----------------|----------------------------|
| ADID | 16 | Char | Application description ID |
| DEADLINE | 10 | Char YMMDDHHMM | Deadline date and time |
| ERRCODE | 4 | Char | Error code |
| GROUPDEF | 16 | Char | Group definition ID |
| IAD | 6 | Char YMMDD | Input arrival date |
| IAT | 4 | Char HHMM | Input arrival time |
| JCLVTAB | 16 | Char | JCL variable table |
| PRIORITY | 4 | Integer | Priority |

Modify VIVL arguments

When you are modifying a virtual workstation destination open interval, the FROM argument is required to identify the interval to be modified. All remaining arguments are optional and provide the information used to modify the open interval.

Note: Always identify a workstation with a MODIFY CPWSV request before a MODIFY VIVL request.

Table 85. Modify VIVL Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|----------------|------------------------------|
| FROM | 10 | Char YMMDDHHMM | Interval start date and time |
| PSCAP | 4 | Integer | Parallel server capacity |
| R1CAP | 4 | Integer | Resource 1 capacity |
| R2CAP | 4 | Integer | Resource 2 capacity |

Communication block address

This is the address returned by INIT request processing, which should remain unmodified for all following requests.

Return code

When EQQYCOM returns control, this fullword shows the outcome of the request.

- 0 The request was successful.
- 4 The MODIFY CPOP request might end with return code 4 if the operation input arrival value specified in the request is earlier than the occurrence. If this happens, run the execute request for the modification to be enforced.
- 8 The request was unsuccessful. An error message has been written to the message log data set.

OPTIONS request

The OPTIONS request lets you specify options to be used when performing PIF requests. You can use these options to automatically:

- Resolve external dependencies when adding LTP or CP occurrences
- Improve the time taken to retrieve information about operations
- Request the address of the area where the message ID is returned
- Prevent messages being written to the message log.
- Handles different versions of the same application. If you delete, insert or replace an application, this operation might cause the change of the valid-to date of all versions involved. By default, different versions of the same application are not supported.

Automatic resolution of external dependencies involves:

- The external predecessors of the occurrence you are inserting are in the LTP or current plan. In the LTP, if more than one occurrence of a specified predecessor application occurs, IBM Workload Scheduler for z/OS selects as predecessor the one with the nearest earlier input arrival time. In the current plan, if more than one occurrence of a specified predecessor application occurs, IBM Workload Scheduler for z/OS selects as predecessor the one with the nearest earlier input arrival time and containing a candidate predecessor.
- All predecessor occurrences selected by the preceding rule are updated so that they specify the new occurrence as a successor.

If automatic resolution is not required, external dependencies that exist in the application descriptions you are inserting are removed before the LTP or current plan is updated.

By default, automatic resolution is *not* performed. When you use the OPTIONS request, the option you choose remains in effect until the end of the current program interface session or until altered by another OPTIONS request. An OPTIONS request can be made any time after the INIT request.

Action code

OPTIONS

Resource code

Not used.

Data area

The data area is used only if the RETMSG or RETMSGID argument name is specified.

The data area address is set to locate an area for a message ID. This address is available on return from the OPTIONS request. At each subsequent program interface request (excluding the TERM request), the ID of an issued message is returned in this area.

The first 3 characters of the returned message ID are **MSG**. The last character is either:

I Information

W Warning

E Error

Blank If the message is suppressed by the SUPMSG OPTIONS request.

The message ID area is blank if no message is issued for a request. If a program interface request causes more than one message to be written to the message log, the message returned is the one considered to be the highest severity. The severity levels are E, blank, W, and I. The highest severity is E (error), and the lowest severity is I (information). If more than one message has the same severity level, the first message issued takes precedence.

When RETMSG is specified, the message ID is part of a larger area. The data available at the address returned is:

Returned message area

| Offset | Length | Type | Description |
|--------|--------|------|---|
| -4 | 2 | Int | Text line length |
| -2 | 2 | Int | Number of text lines |
| 0 | 8 | Char | Message ID |
| +8 | * | Char | Text lines, the length is the number of text lines multiplied by the text line length |

Arguments

Argument name=LTDEPR

Automatic resolution of external dependencies when inserting new LTP occurrences.

Argument value=Y

Yes.

Argument value=N

No (default).

Argument name=CPDEPR

Automatic resolution of external dependencies when inserting new current plan occurrences.

Argument value=Y

Add successor and predecessor dependencies.

Argument value=N

Do not add any dependencies (default).

Argument value=P

Add predecessor dependencies.

Argument value=S

Add successor dependencies.

Argument name=FASTPATH

FASTPATH can make the search for operations considerably faster when you want only to retrieve computer and printer operations.

Argument value=Y

If you specify Y (YES), IBM Workload Scheduler for z/OS searches the current plan for computer or printer operations matching the job name search argument. It then selects *all* operations in the occurrences that contain these computer or printer operations (that is, even operations at general workstations), and retrieves these operations based on the remaining search arguments that you have specified.

Argument value=N

If you specify N (NO), which is the default value, all operations are retrieved that match the search argument criteria that you have specified.

Argument name=RETMSG

This argument lets you request the address of the area where the message ID and message text is returned. The address points to the message ID, the layout of the area is described in the paragraph "Data area" on page 60. There is no argument value for this argument name.

Argument name=RETMSGID

This argument lets you request the address of the area where the message ID is returned. There is no argument value for this argument name.

Argument name=SUPMSG

SUPMSG lets you prevent a message from being written to the message log. You can prevent more than one message from being written to message log by issuing multiple OPTIONS requests with the SUPMSG argument specified.

Argument value=MSG_{msgid}

Specify MSG followed by the message identifier. To obtain the message identifier, remove the IBM Workload Scheduler for z/OS prefix (EQQ) from the beginning of the message and the severity indicator from the end of the message.

For example, to prevent message EQQW002E from being written to the message log, specify an argument value of MSGW002.

Argument name=ADVERS

Application description versions support when delete, insert or replace an AD record.

Argument value=Y

Yes. When inserting or replacing an AD record, and another record with the same ADID exists, the VALTO and VALFROM values will be set so that the different versions of the application have consecutive validity intervals, with the same logic used by the ISPF dialogs.

Argument value=N

No (default). The AD record is stored as provided by the user.

Argument name=ADOICHK

Use this option to specify whether or not you want AD/OI consistency checks to be made every time an application is deleted or modified.

Consistency checks involve looking in the application description data base for matches for all the operator instructions in the application. Any operator instructions without a match are deleted.

The checks are made immediately after the application description PIF action has completed with a zero return code.

Argument value=Y

Yes. Consistency checks are performed whenever an application description record is deleted or replaced using the PIF.

Argument value=N

No (default). Consistency checks are not performed.

Argument name=VERADGRD

Application descriptions that are members of an application group have the name of the group definition in field ADGROUPIP of segment ADCOM. VERADGRD controls the verification of this field when a new application description is created or an existing one is replaced. The verification is done for active application descriptions.

Argument value=F

The group definition is verified to check that it exists, is active and valid for at least a part of the validity period of the application description being created or updated.

Argument value=Y

Same as for value=F, except that the application group id is accepted if the application description already has this application group id. It could be an update without any change to the application group id or an insert of a new version when there already are active versions with the same application group id.

Argument value=N (default)

No check is made to verify that the application group exists.

Argument name=VERSRWSN

The special resource description, SR, has fields representing workstations, the full workstation names or generic names; field SRDWSNAME of segment SRDWS for default connected workstations, field SRIWSNAME of segment SRIWS for workstations connected to an interval. VERSRWSN controls the verification of these fields when a new special resource is created or an existing one is replaced.

Argument value=F

The workstation fields are verified against the workstation description file. Each workstation field in the resource description must match at least one of the workstation descriptions.

Argument value=Y

Same as for value=F except that the workstation value is accepted if the resource description already has this workstation name. It could be an update without any change to the workstation names.

Argument value=N (default)

No check is made to verify that the workstation description exists.

Argument name=DURSEC

This argument lets you decide the duration format of Insert and Replace Action of AD/WS record. You can also specify this option request in the INIT statement of the EQQYPARM file. ADOPDUR and WSOPDUR fields contain duration value in minutes. APOPDURI and WSOPDURI fields contain duration value in hundredths of a second. If DURSEC is not specified, Adopur/Wsopdur value will be used.

Argument Value=Y

Adopduri/Wsopduri will be always used.

Argument Value=N

The field Adopduri/Wsopduri will be checked to have the same value of the field Adopdur/Wsopdur when the field Adopduri/Wsopduri is rounded up to a number of minutes. If this happens, it means that no change occurred in the duration value and the field Adopduri/Wsopduri will be used. If the Adopduri/Wsopduri value is different from the Adopdur/Wsopdur one, it means that the user changed duration value in Adopdur/Wsopdur and this field will be used.

Communication block address

This is the address returned by INIT request processing, which should remain unmodified for all following requests.

Return code

When EQQYCOM returns control, this fullword shows the outcome of the request:

- 0 The request was successful.
- 8 The request was unsuccessful. An error message has been written to the message log data set.

REPLACE request

The REPLACE request replaces an existing record in the application description or operator instruction database with a record provided by your program. If the record type is other than an application description record, then the record provided by your program must have the same key fields as a record on the database; otherwise, no replace is performed.

If the record type is an application description record, then the record provided by your program can have the *STATUS* field modified, even if this field is part of the key. In this case, you must supply the old *STATUS* value and the *VALTO* value of the application to be replaced in the arguments. You must also set the *ADVERS* argument value to Y in the *OPTIONS* request as Y as well.

The replacing record is placed in the data area by your program. Arguments are not used if the resource code is different from AD or if you set the *ADVERS* argument value in the *OPTIONS* request to N.

Action code

REPLACE

Resource code

The resource code identifies the record type you want to replace. You can specify these values:

AD Application description record
AWSCL
All workstation closed record
CL Calendar record
CSR Current plan special resource
ETT Event triggered tracking criteria record
JCLV JCL variable table record
JS Job control language record
OI Operator instruction record
PR Period record
RG Run cycle group record
SR Special resource record
WS Workstation record
WSV Virtual workstation destination record

Notes:

1. If you do not provide the application description (AD) record TYPE, or the AD record TYPE is not recognized, *application* is assumed. The priority field is not used for an AD group definition
2. The format of duration used in the data area, in Replace AD/WS will be defined by the DURSEC option, described in the paragraph "OPTIONS request" on page 59

Data area

You must put the address of your data area in the fullword whose address is in the parameter list. The data area consists of a header and the actual record to be written to the database. Ensure that the header and data record are in the correct format. "Header format" on page 10 describes the format for a header. Appendix A. Program Interface Record Format describes the format for the data records.

In the CSRCOM segment of the CSR record only a subset of the fields can be changed:

CSRUSEDFOR
CSRONERROR
CSROVAV
CSROVQ
CSRDEVI
CSRDEFQUANT
CSRDEFVAIL

The values in the rest of the CSRCOM fields are ignored and the values in the resource record are left unchanged.

Arguments

Replace AD arguments

Table 86. Replace AD Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|---------------|-----------------------------|
| STATUS | 1 | Char | Status: P=Pending; A=Active |
| VALTO | 6 | Char YYYYMMDD | Valid-to date |

For resource codes other than AD no arguments are supported. The new record must be made available via the data address parameter.

Communication block address

This is the address returned by INIT request processing, which should remain unmodified for all following requests.

Return code

When EQQYCOM returns control, this fullword shows the outcome of the request:

- 0 The request was successful.
- 4 The record; AD, AWSCL, CL, ETT, JS, JCLPREP, JCLPREPA, JCLV, OI, PR, SR, or WS is being updated by another user. The record is replaced.
- 8 The request was unsuccessful. An error message has been written to the message log data set.

RESET request

The RESET request deletes the current MCP block. This effectively cancels a series of modify current plan requests that have been collected in an MCP block, if it is performed before an EXECUTE request.

RESET is required when an error occurs, if you have made more than 1 MODIFY or INSERT CPOC request before an EXECUTE request. If you do not specify RESET, successful MODIFY or INSERT requests are processed in the next EXECUTE MCPBLK request.

Action code

RESET

Resource code

About this task

MCPBLK

Data area

About this task

Not used.

Arguments

About this task

Not used.

Communication block address

About this task

This is the address returned by INIT request processing, which should remain unmodified for all following requests.

Return code

About this task

When EQQYCOM returns control, this fullword shows the outcome of the request:

- 0 The request was successful.
- 8 The request was unsuccessful. An error message has been written to the message log data set.

SELECT request

About this task

The SELECT request retrieves a record and makes it available to your program. You can:

- Retrieve a record directly from IBM Workload Scheduler for z/OS by specifying field names and values in arguments, which identify the record you want to retrieve. When you retrieve a record directly from IBM Workload Scheduler for z/OS, you can get the complete record rather than just the common segment that is available from a list.
- Retrieve one of the records from a list built by a previous LIST or SELECT request by providing the resource code (common segment name), and the pointer to the offset of the common segment data area that contains the common segment of the record. This pointer is in the header record of the common segment.

Action code

About this task

SELECT

Resource code

About this task

If you want to retrieve one of the records from a previously built list, *you must use the same resource code that you used when you built the list with the LIST request*. The arguments NEXT, PREV, FIRST, and LAST direct the selection to a list. The resource code shows which list previously built contains the required record. There can be a maximum of one active list for each resource code.

If you want to retrieve a record directly from IBM Workload Scheduler for z/OS, the resource code indicates the record type. You can specify these values:

- AD** Application description record
- ADCOM**
Application description, common segment only
- AWSCL**
All workstations closed record
- CL** Calendar record
- CLCOM**
Calendar record, common segment
- CPCOND**
Current plan condition segment
- CPCOND**
Current plan condition common segment

CPOC, CPOCCOM
 Current plan occurrence record
CPOP Current plan operation record
CPOPCOM
 Current plan operation record, common segment
CPOPSRU
 Current plan operation segment with information about the operation in
 relation to a special resource
CPST Current plan status record
CPUSRF
 Current plan user field record
CPWS Current plan workstation record
CPWSV
 Current plan virtual workstation destination record
CPWSCOM
 Current plan workstation record, common segment
CPWSCOM
 Current plan virtual workstation destination record, common segment
CSR Current plan special resource
CSRCOM
 Current plan special resource, common segment
ETT Event triggered tracking criteria record
JCLPREP
 Retrieve promptable setup variables for the current operation
JCLPREPA
 Resolve all nonpromptable setup variables for the current operation
JCLV JCL variable table record
JCLVCOM
 JCL variable table record, common segment
JLCOM
 JS file job log common segment
JS Job control language record
JSCOM
 Job control language record, common segment
LTOC LTP occurrence record
LTOCCOM
 LTP occurrence record, common segment
OI Operator instruction record
OICOM
 Operator instruction record, common segment
PR Period record
PRCOM
 Period record, common segment
RG Run cycle group record
RGCOM
 Run cycle group common segment
SR Special resource record
WS Workstation description record
WSCOM
 Workstation description record, common segment
WSV Virtual workstation destination record
WSVCOM
 Virtual workstation destination record, common segment
XENV Execution environment segment

Note:

1. The SELECT JS and SELECT JSCOM requests try to retrieve JCL from the JCL repository. If no JCL is found, it is retrieved from the JCL library or through the job-library-read exit, EQQUX002. The full key is required, that is, the application ID, the input arrival time, and the operation number. You might need to precede the SELECT JS request by a LIST CPOPCOM request to get the key values.
2. LIST JSCOM requests try to retrieve JCL only from the JCL repository.
3. SELECT CPOPSRU can be issued for list elements only, from a list created by LIST CPOPSRU.

Data area

About this task

When EQQYCOM returns control to your program after a successful SELECT request, this fullword contains the address of the data area containing the requested record.

If you are retrieving a record from a list, only the common segment of the record is returned. A description of the fields in the common segment of each record can be found in Appendix A. Program Interface Record Format.

If you are retrieving a record directly from IBM Workload Scheduler for z/OS, the complete record with all segments can be returned, depending on the resource type. A description of the segments in each record and the fields in each segment can be found in Appendix A. Program Interface Record Format.

The header section for this record contains, besides the normal header information, a field containing one of these items:

- The index number of the record in the list, if the record was retrieved from a LIST. For example, **1** for the first record in the list, **2** for the second.
- The length of the data area (header and data), if the record was not retrieved from a LIST.

This field is in the final header entry, that is, the entry that has a blank segment name field. The count is stored in the field that normally contains the segment offset. A complete description of headers can be found under “Header format” on page 10.

Arguments

About this task

Retrieving a record from a list

If you want to retrieve a record from a list built by a previous LIST request, you must use one of these argument names:

- NEXT** Retrieve the next record from the list.
PREV Retrieve the previous record from the list.
FIRST Retrieve the first record from the list.
LAST Retrieve the last record from the list.

A corresponding argument value is not used.

When a LIST is created, IBM Workload Scheduler for z/OS sets the first element in the list as the *current* element. Each time a SELECT request is performed on a list, the current element is updated according to which of these argument names

was used. If you have several lists active, IBM Workload Scheduler for z/OS remembers the current element for each of them.

In combination with one of the above arguments, you can use one or more arguments described under “Retrieving a record directly from IBM Workload Scheduler for z/OS.” This is best illustrated with an example:

```
Action code:  SELECT
Resource code: ADCOM
Argument names:      Values:
                   NEXT      -
                   STATUS     A
                   PRIORITY   9
```

Figure 4. Example of arguments for processing a list

Assuming a previously successful LIST request has executed for the ADCOM resource, the parameters in this example cause IBM Workload Scheduler for z/OS to search the ADCOM list forward from the current element until it finds an element with STATUS A and PRIORITY 9.

This example gives you a mechanism for processing the list you have previously built using a LIST request. After a successful SELECT request, the required record from the list is available in the data area.

Retrieving a record directly from IBM Workload Scheduler for z/OS

When you are retrieving a record directly from IBM Workload Scheduler for z/OS as opposed to a record from a list, the arguments identify which record you want to retrieve. Two ways you can do this are:

- Specify field names of the record as argument names. The argument values specify values for these fields that identify the particular record you want to retrieve. Argument values can be:
 - Character values. A blank character terminates the field.
 - Numeric values, which must occupy a fullword.

You must specify sufficient arguments to *uniquely* identify a record. You can use a comparison operator after the argument values. The default, an equals sign (=), is assumed if you do not.

- Use the common part of the record, which you have previously retrieved with a LIST or a SELECT request, to identify the required record. Here the argument name specifies the resource code (common segment name), and the argument value specifies the address of the common segment data area that contains the common segment of the record. See Table 4 on page 15.

CPST (current plan status) is only one record; therefore, select arguments are not required.

Note: The values of PIF arguments as dates depend on the PIF base year, which is defined by the PIFCWB keyword on the INTFOPTS statement, or the CWBASE keyword of the INIT statement. The value of the VALTO argument for default high date depends on the PIFHD keyword of the INTFOPTS statement or the HIGHDATE keyword of the INIT statement. See *IBM Workload Scheduler for z/OS: Customization and Tuning* for more details on these statements.

You can specify the following values.

Select AD, ADCOM arguments

Table 87. Select AD, ADCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|--|
| ADID | 16 | Char | Application description ID |
| ADRULEP | 8 | Char | Name of period or run cycle group |
| GROUP | 8 | Char | Authority group name |
| GROUPDEF | 16 | Char | Group definition ID |
| MONITOR | 1 | Char | Y=application with at least one operation monitored by an external product N=application with no operation monitored by an external product |
| OWNER | 16 | Char | Owner ID |
| PRIORITY | 4 | Integer | Priority |
| STATUS | 1 | Char | Status: P=Pending A=Active |
| TYPE | 1 | Char | Application type: A=Application G=Group. Default is A |
| VALFROM | 6 | Char YYMMDD | Valid-from date |
| VALTO | 6 | Char YYMMDD | Valid-to date |

Note:

1. IBM Workload Scheduler for z/OS assumes application type A if you do not specify the AD argument name TYPE.
2. The ADSAI segment is retrieved only if the system automation information is defined for the selected application.

Select AWSCL arguments

Table 88. Select AWSCL Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|-------------|
| DATE | 6 | Char YYMMDD | Date |

Select CL, CLCOM arguments

Table 89. Select CL, CLCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------|
| CALENDAR | 16 | Char | Calendar ID |

Note: If the name of the default calendar is specified in the EQQYPARM INIT statement, SELECT CL without the CALENDAR argument will return the default calendar. Otherwise CALENDAR is a required argument.

Select CPCOND, CPCONDCO arguments

Table 90. Select CPCOND, CPCONDCO Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| ADID | 16 | Char | Application description ID |
| IA | 10 | Char | Input arrival date and time |
| OPNO | 4 | Integer | Operation number |
| CONDID | 4 | Integer | Condition ID. Valid values are from 1 to 999. |
| CONDVAL | 4 | Char | Final condition status: U = Undefined T = True F = False |

Select CPOC, CPOCCOM arguments

Table 91. Select CPOC Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|--------------------|--|
| ADID | 16 | Char | Application description ID |
| GROUP | 8 | Char | Authority group |
| GROUPDEF | 16 | Char | Group definition ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| MCPADDED | 1 | Char | MCP added, Y or N |
| MONITOR | 1 | Char | Y=occurrence with at least one operation monitored by an external product N=occurrence with no operation monitored by an external product |
| OWNER | 16 | Char | Owner ID |
| PRIORITY | 4 | Integer | Priority |
| RERUN | 1 | Char | Rerun requested, Y or N |
| STATUS | 1 | Char | Occurrence status |

Note: When the STATUS argument is specified, its value can be W, S, C, E, U, D.

Select CPOP, CPOPCOM arguments

Table 92. Select CPOP, CPOPCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|----------------------------|
| ADID | 16 | Char | Application description ID |
| CLNSTAT | 1 | Char | Data Set cleanup status |
| CLNTYPE | 1 | Char | Data Set cleanup type |
| CONDRJOB | 1 | Char | Conditional recovery job |
| DPREM | 1 | Char | Removable by DP |
| ERRCODE | 4 | Char | Error code |

Table 92. Select CPOP, CPOPCOM Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|-------------------|--|
| EXECDEST | 8 | Char | Execution destination. To indicate a local destination, specify ***** |
| EXPJCL | 1 | Char | Expanded JCL option |
| EXTNAME | 54 | Char | Operation extended name |
| EXTSE | 16 | Char | Scheduling Environment name |
| GROUP | 8 | Char | Authority group |
| IA | 10 | Char YYYYMMDDHHMM | Input arrival date and time |
| JOBCRT | 1 | Char | Critical job: P=Critical path target W=Eligible for WLM assistance N=Not eligible for WLM assistance |
| JOBNAME | 8 | Char | Job name |
| JOBPOL | 1 | Char | Workload monitor late job policy: ' ' (blank) = default L = Long duration D = Deadline S = Latest start time C = Conditional mode |
| MONITOR | 1 | Char | Y=operation monitored by an external product N=operation not monitored by an external product |
| OPNO | 4 | Integer | Operation number |
| OWNER | 16 | Char | Owner ID |
| PRIORITY | 4 | Integer | Priority |
| SHADOWJ | 1 | Char | Shadow job, Y or N |
| STATUS | 1 | Char | Operation status |
| UNEXPRC | 1 | Char | Y=Unexpected RC is ON N=Unexpected RC is OFF |
| USRSYS | 1 | Char | User sysout support |
| VIRTDEST | 8 | Char | Submission destination. To indicate a local destination, specify ***** |
| WAITFORW | 1 | Char | Started on WAIT workstation, Y or N |
| WAITNAME | 1 | Char | Waiting for Scheduling Environment, Y or N |
| WLMSCLS | 8 | Char | WLM service class |
| WMPRED | 1 | Char | Waiting for mandatory pending predecessors, Y or N |
| WPMPRED | 1 | Char | Waiting for either mandatory pending or pending predecessors, Y or N |

Table 92. Select CPOP, CPOPCOM Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| WPPRED | 1 | Char | Waiting for pending predecessors, Y or N |
| WSNAME | 4 | Char | Workstation name |

Note: The ADSAI segment is retrieved only if the system automation information is defined for the selected application.

Select CPUSRF arguments

By running the Select CPUSRF, the CPUSRFELEM segment is retrieved for all the user fields related to the operation.

Table 93. Select CPUSRF Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-----------------------------|
| ADID | 16 | Char | Application description ID |
| IA | 10 | Char | Input arrival date and time |
| OPNO | 4 | Integer | Operation number |

Select CPWS, CPWSCOM arguments

Table 94. Select CPWS, CPWSCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------------------------------|
| WSAUTO | 1 | Char | Automation workstation, Y or N |
| WSNAME | 4 | Char | Workstation name |
| WSREP | 1 | Char | Workstation reporting attribute |
| WSRETYPE | 1 | Char | Remote engine type (Z, D, or blank) |
| WSTWS | 1 | Char | Fault-tolerant workstation, Y or N |
| WSTYPE | 1 | Char | Workstation type |
| WSVIRT | 1 | Char | Virtual workstation, Y or N |
| WSWAIT | 1 | Char | WAIT Workstation, Y or N |
| WSZCENTR | 1 | Char | z-centric workstation, Y or N |

Select CPWSV, CPWSVCOM arguments

About this task

Table 95. Select CPWSV, CPWSVCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| WSNAME | 4 | Char | Virtual workstation name |
| WSDEST | 8 | Char | Virtual workstation destination. To indicate a local destination, specify ***** |

Select CSR, CSRCOM arguments

About this task

Table 96. Select CSR, CSRCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| RESALCS | 1 | Char | Whether or not any operation is currently allocating the resource shared, Y or N |
| RESAVAIL | 1 | Char | Whether or not the resource is available, Y or N |
| RESGROUP | 8 | Char | Resource group name |
| RESHIPER | 1 | Char | Whether or not it is a DLF control resource, Y or N |
| RESNAME | 44 | Char | Resource name |
| RESWAIT | 1 | Char | Whether or not any operation is waiting for the resource. |

The argument MATCHTYP is supported.

Select ETT arguments

About this task

Table 97. Select ETT Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| ADID | 16 | Char | Associated application ID |
| ETTNAME | 44 | Char | Name of trigger |
| ETTTYPE | 1 | Char | Type of trigger: 2=job 3=special resource |

Select JCLPREP arguments

Table 98. Select JCLPREP Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|--------------------|-----------------------------|
| ADID | 16 | Char | Application ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| OPNO | 4 | Integer | Operation number |

"JCL preparation using PIF" on page 80 describes JCL preparation using the program interface.

Select JCLPREPA arguments

About this task

Table 99. Select JCLPREPA Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|-----------------------------|
| ADID | 16 | Char | Application ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| OPNO | 4 | Integer | Operation number |

Table 99. Select JCLPREPA Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|--------------------------|--|
| SIMTIME | 12 | Char CCYYMMDDHHMM | Simulated time. CCYY can have the values 1984 to 2071. |
| SIMTYPE | 8 | Char "FULL" or "PARTIAL" | Simulation type |

"JCL preparation using PIF" on page 80 describes JCL preparation using the program interface.

Select JCLV, JCLVCOM arguments

Table 100. Select JCLV, JCLVCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-----------------------|
| JCLVTAB | 16 | Char | JCL variable table ID |

Select JLCOM arguments

About this task

Table 101. Select JLCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|--------------------|-----------------------------|
| ADID | 16 | Char | Application ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| JOBNAME | 8 | Char | z/OS Job name |
| OPNO | 4 | Integer | Operation number |
| WSNAME | 4 | Char | Workstation name |

Select JS, JSCOM arguments

About this task

Table 102. Select JS, JSCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------------|-----------------------------|
| ADID | 16 | Char | Application ID |
| IA | 10 | Char YYMMDDHHMM | Input arrival date and time |
| JOBNAME | 8 | Char | z/OS Job name |
| OPNO | 4 | Integer | Operation number |
| WSNAME | 4 | Char | Workstation name |

Select LTOC, LTOCCOM arguments

Table 103. Select LTOC, LTOCCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-------------|---------------------|
| ADID | 16 | Char | Application ID |
| GROUP | 8 | Char | Authority group |
| GROUPDEF | 16 | Char | Group definition ID |
| IAD | 6 | Char YYMMDD | Input arrival date |

Table 103. Select LTOC, LTOCCOM Arguments (continued)

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--------------------|
| IAT | 4 | Char HHMM | Input arrival time |
| OWNER | 16 | Char | Owner ID |

Select OI, OICOM arguments

Table 104. Select OI, OICOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|----------------------|------------------------|
| ADID | 16 | Char | Application ID |
| OPNO | 4 | Integer | Operation number |
| VALTO | 10 | Char (YYMMDDHHMM) | Valid-to date and time |

Select PR, PRCOM arguments About this task

Table 105. Select PR, PRCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|-------------|
| PERIOD | 8 | Char | Period name |
| PRTYPE | 1 | Char | Period type |

Select RG, RGCOM arguments

Table 106. Select RG, RGCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--------------------------------|
| RGID | 8 | Char | Run cycle group ID |
| RGOWNER | 16 | Char | Run cycle group owner |
| RGCALEND | 16 | Char | Run cycle group calendar |
| RGVARTAB | 16 | Char | Run cycle group variable table |
| RUNNAME | 8 | Char | Run cycle name |
| RUNCAL | 16 | Char | Run cycle calendar |
| RUNVTAB | 16 | Char | Run cycle variable table |
| RUNSETID | 8 | Char | Run cycle subset ID |

Select SR, SRCOM arguments About this task

Table 107. Select SR, SRCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---------------------------|
| RESGROUP | 8 | Char | Special resource group ID |
| RESHIPER | 1 | Char | DLF resource indicator |
| RESNAME | 44 | Char | Special resource name |

Select WS, WSCOM arguments

Table 108. Select WS, WSCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|---|
| WSAUTO | 1 | Char | Automation workstation, Y or N |
| WSNAME | 4 | Char | Workstation name |
| WSREP | 1 | Char | Workstation reporting attribute |
| WSRETYPE | 1 | Char | Remote engine type: D = distributed, Z = z/OS or blank |
| WSTWS | 1 | Char | Fault-tolerant workstation, Y or N |
| WSTYPE | 1 | Char | Workstation type |
| WSVIRT | 1 | Char | Virtual workstation, Y or N |
| WSWAIT | 1 | Char | WAIT workstation, Y or N |
| WSZCENTR | 1 | Char | z-centric workstation, Y or N |

Select WSV, WSVCOM arguments

Table 109. Select WSV, WSVCOM Arguments

| Arg names | Length | Data type | Description |
|-----------|--------|-----------|--|
| WSNAME | 4 | Char | Virtual workstation name |
| WSDEST | 8 | Char | Virtual workstation destination. To indicate a local destination, specify ***** |

Selecting a record using a common segment

If you have already retrieved the common segment of a record but you then want to retrieve the entire record, you can specify the segment name as an argument name and the address of the previously retrieved common segment as the argument value address.

For current plan operations, segment CPOPSRU can be used as well as the common segment.

Communication block address

About this task

This is the address returned by INIT request processing, which should remain unmodified for all following requests.

Return code

About this task

When EQQYCOM returns control, this fullword shows the outcome of the request:

- 0 The request was successful.
- 4 The request was unsuccessful.

No records meet the criteria specified by the arguments.

- 6 You are not authorized to read the record. You specified a unique key in the SELECT request; the record exists, but you do not have authority to read it.
- 8 The request was unsuccessful. An error message has been written to the message log data set. This can occur if more than one record in the database satisfies the field values specified by your arguments. For example, you want to select an application description record with the ID APPL1, and there are two such application descriptions in the database with different validity dates. Your arguments must specify both the application ID and the valid-from date to uniquely identify the record.

SETSTAT request

The SETSTAT request changes the condition status from undecided to true or false, if the original status is undecided because of missing step-end information.

It produces the same result as the T and F commands available from the MCP dialog.

Action code

SETSTAT

Resource code

About this task

CPSIMP

Data area

About this task

Not used.

Arguments

About this task

The arguments identify which condition dependency with undefined status is to be reset.

The same arguments apply as for the INSERT CPSIMP request, listed in Table 40 on page 34.

To identify the new status, use the following argument:

Table 110. Setstat CPSIMP Argument

| Arg name | Length | Data type | Description |
|----------|--------|-----------|--|
| NEWSTAT | 1 | Char | Requested status: T = True F = False |

Communication block address

This is the address returned by INIT request processing, which must remain unmodified for all following requests.

Return code

When EQQYCOM returns control, this fullword shows the outcome of the request:

- 0 The request was successful.
- 8 The request was unsuccessful. An error message has been written to the message log data set.

TERM request

About this task

The TERM request terminates the program interface session and performs this cleanup processing:

- FREEMAIN of storage
- Close data sets
- Detach subtasks
- Termination of the IBM Workload Scheduler for z/OS session.

It must be the last request of a session. A TERM request is necessary if the INIT request executed successfully.

Action code

TERM

Resource code

Not used.

Data area

Not used.

Arguments

About this task

Not used.

Communication block address

This is the address returned by INIT request processing, which should remain unmodified for all following requests, including the TERM request.

Return code

When EQQYCOM returns control, this fullword shows the outcome of the request:

- 0 The request was successful. A program interface session has been successfully terminated.
- 8 The request was unsuccessful. An error message has been written to the message log data set.

Note: If EQQYCOM abends, processing is immediately interrupted and the control is passed to the system, therefore no return code is provided. Resolve the abend according to the programming language you are using.

JCL preparation using PIF

You can perform JCL preparation through the program interface using resource type **JS** for JCL records, or **JCLPREP** for promptable variables. You can use the resource type **JCLPREPA** rather than a combination of **JS** and **JCLPREP** requests. You can also use **JCLPREPA** to simulate variable substitution. This lets you perform *trial substitution* of your variables without updating a job.

IBM Workload Scheduler for z/OS Managing the Workload describes variable substitution and job tailoring in detail.

Substituting variables

JCL preparation can be:

SETUP=PROMPT

A user must assign the value.

SETUP=YES

A value is automatically assigned at JCL preparation, or at submit time if no JCL preparation is performed.

SETUP=NO

A value is assigned at submit time.

SELECT JCLPREP retrieves the promptable variables that do not have a value assigned. When returned, the data area parameter locates a JCL setup variable record, the header, a common segment, and a sequence of variable segments **JSVV**. The field **JSVVVALUE** of the **JSVV** segment can be assigned a new value.

INSERT JCLPREP is used to make the promptable variables in the JCL setup variable record assigned to the JCL record. When all promptable variables of the JCL record are assigned, **SELECT JCLPREP** receives a return code 4.

To update the JCL record, you must execute a **SELECT JS** request followed by an **INSERT** or **REPLACE JS** request. When the **SELECT JS** is returned, the retrieved JCL record will have promptable variables resolved if a value was assigned in the **SELECT JCLPREP**, **INSERT JCLPREP** sequence. Nonpromptable setup variables are also resolved, while submit variables remain unresolved. An **INSERT** or **REPLACE JS** request is required to have the updated JCL reflected in the database and must be complete to end the JCL preparation session.

If the JCL record is not present on the JS file, an **INSERT JS** request is required. A **LIST JS** request will get return code 4 if the JCL record is not found in the JS file. **SELECT JS** will retrieve the JCL from the job library **EQQJBLIB**.

If the JCL record does not contain promptable variables, **SELECT JCLPREPA** must be used to assign values to nonpromptable setup variables. So, if the first **SELECT JCLPREP** results in return code 4, a **SELECT JCLPREPA** must be executed instead of the **SELECT JS** before the **INSERT** or **REPLACE JCL** request.

The sample library member **EQQPIFAP** contains a sample program that resolves JCL variables using the program interface. See Appendix C, "Sample library (SEQQSAMP)," on page 361 for more information about individual members of the sample library.

Example of a PIF request logic flow:

```

INIT
OPTIONS
DO while(RC=0)
  SELECT JCLPREP      (opno of the JOB operation)
    set up the prompt var
  INSERT JCLPREP     (opno of the JOB operation)
END
LIST JSCOM           (opno and wsn of JOB operation)
SELECT JS            (opno and wsn of JOB operation)
  check RC from the LIST JSCOM
  if RC=0 then
    REPLACE JS
  if RC=4 then
    INSERT JS
TERM

```

Note: If there is a SETUP operation for this computer operation, and if you want to set it to Complete, add the following statements before the TERM request:

```

MODIFY CPOC
MODIFY CPOP          (opno and wsn of SETUP operation)
EXECUTE

```

For a description of the SETUP and JOB setup operations, see *IBM Workload Scheduler for z/OS: Managing the Workload*.

Simulating variable substitution

You can use JCLPREPA arguments to perform trial substitutions, before normal substitution by IBM Workload Scheduler for z/OS. You might need to do this, for example, if you use a product that checks JCL.

You can request partial or full simulation. For partial simulation, only nonpromptable setup variables are substituted. For full simulation:

- Submit variables are substituted.
- Nonpromptable setup variables are substituted.
- Promptable setup variables are substituted using the default values. You must specify the defaults when calling PIF, otherwise no substitution takes place and the JCL might contain &, ?, and % characters.
- PHASE=SETUP directives are returned to the caller, even though IBM Workload Scheduler for z/OS only simulates submission.
- You can supply a time value in the SIMTIME argument for IBM Workload Scheduler for z/OS-supplied variables that contain a *current time* value. IBM Workload Scheduler for z/OS uses the current time if you do not specify SIMTIME.
- JCL is returned even if errors were found, except for the case when the JCL exceeds the JS size. Error and warning messages are inserted in the JCL.

Chapter 2. The Application Programming Interface (API)

This chapter explains how you use the IBM Workload Scheduler for z/OS application programming interface (API) to communicate with IBM Workload Scheduler for z/OS. Through the API you can:

- Extract information about the current plan (GET request)
- Update or add current-plan operations (PUT request)
- Delete operations in the current plan (DEL request)
- Report events to IBM Workload Scheduler for z/OS (CREATE request).

IBM Workload Scheduler for z/OS uses the services of APPC to communicate with an application transaction program (ATP). Before you can use the API, IBM Workload Scheduler for z/OS support for APPC must be active. For details, see *IBM Workload Scheduler for z/OS: Planning and Installation*.

This chapter describes CPI-C verbs that are supported by IBM Workload Scheduler for z/OS. ATPs that use CPI-C are more easily integrated and transported across supported environments. For more information about CPI-C verbs, refer to *CPI-C Communications Reference*.

Samples are provided with IBM Workload Scheduler for z/OS to help you set up and use the API. Appendix C, “Sample library (SEQQSAMP),” on page 361 describes these samples.

Communicating with IBM Workload Scheduler for z/OS

To establish communication with IBM Workload Scheduler for z/OS, your ATP must initialize and then allocate a conversation. The ATP must supply all information that is required to initialize the conversation; for example, the partner transaction program (TP) name and its LU, and a user ID and password that is used for security checking. Supply TP name EQQAPI to communicate with IBM Workload Scheduler for z/OS. For GET, PUT, and DEL requests, the LU that the ATP sends requests to (the target LU) must be owned by the controller. For CREATE requests, if the target LU is not owned by an IBM Workload Scheduler for z/OS address space where an event writer task is started, the ATP must send requests so that the events are broadcast on the target z/OS system. “Broadcasting events” on page 95 describes how you broadcast events on the target system.

When communication is established, your ATP sends a request to IBM Workload Scheduler for z/OS in a *send buffer*. IBM Workload Scheduler for z/OS responds by issuing a *receive*, inviting more requests from your ATP while it is processing the request. When you have completed your requests, you should issue several *receive* requests to ensure all data is received by the ATP. In cases where the *receive* type is *Receive_Immediate*, or if the buffers are large, data is returned in *packets*.

When the request has been processed, IBM Workload Scheduler for z/OS builds a buffer that is sent to your ATP the next time that the ATP issues a *receive* request. This buffer is called a *receive buffer*.

If there is more than one active request from your ATP at a given time, you can identify each request by setting the token field (APPTOKEN in the APP section) to a unique value. The value could be, for example, a time stamp.

You can continue to make requests while the conversation is established. When you want to end the conversation, your ATP must issue a deallocate verb.

Note: The data that you send to IBM Workload Scheduler for z/OS must be in EBCDIC format. IBM Workload Scheduler for z/OS returns the data in the same format. If you use ASCII code, ensure that your data is converted to EBCDIC before a request is sent to IBM Workload Scheduler for z/OS, and converted to ASCII when data is received by the ATP. Also, binary values might have to be swapped because the order of the byte representation (high-low, low-high) is machine dependent.

These publications contain detailed information about writing an application program in the APPC environment:

APPC and CPI-C Implementations

APPC Programming Considerations

APPC Application Examples

CPI-C support provided by IBM Workload Scheduler for z/OS

Your ATPs can issue requests to IBM Workload Scheduler for z/OS through the API using CPI-C. Although your programs can use any CPI-C verbs, you should consider this information before you write your programs. It describes how the partner TP, IBM Workload Scheduler for z/OS, responds to certain verbs:

CMACCP

Accept_Conversation

CMACCP is not applicable because the ATP must initialize and allocate the conversation.

CMALLC

Allocate

CMALLC must be issued by the ATP to allocate the conversation.

CMCFMD

Confirmed

CMCFMD is returned by IBM Workload Scheduler for z/OS when a confirm verb is issued by the ATP. But IBM Workload Scheduler for z/OS does not perform additional processing for a confirm request. The confirmed verb is issued when the request is received.

CMINIT

Initialize_Conversation

CMINIT must be issued by the ATP to initialize the conversation.

CMRCV

Receive

The ATP should repeat CMRCV calls to ensure that it receives the requested data. This is because when IBM Workload Scheduler for z/OS receives the send state from the ATP and has no data to send at that time, it issues a receive inviting the ATP to send more requests. So the ATP determines the frequency of the polling.

CMSED

Set_Error_Direction

CMSED can be issued but is not used by IBM Workload Scheduler for z/OS.

CMSERR

Send_Error

CMSERR can be issued but is not used by IBM Workload Scheduler for z/OS.

CMSLD

Set_Log_Data

CMSLD can be issued but is not used by IBM Workload Scheduler for z/OS.

CMSTPN

Set_TP_Name

Specify TP name `EQQAPI`, which is the default name. IBM Workload Scheduler for z/OS recognizes these TP names:

EQQTRK

Supplied by trackers that communicate with the controller through `APPC`

EQQAPI

Supplied by user programs (ATPs) that communicate with IBM Workload Scheduler for z/OS through the API.

API buffer layouts

There are two buffer types, send buffers and receive buffers. All buffers are in EBCDIC format and must be in contiguous storage. The buffers can contain these sections:

APP Fixed section**APPFLD**

Field section

APPDAT

Data section

APPOBJ

Object section

APPSEL

Selection section

APPVAL

Selection value section

The sections that a send buffer should contain depends on the request that you make. Table 111 shows the sections that you can include for each request:

Table 111. Contents of a Send Buffer

| Request | Buffer sections ¹ | | | | | |
|---------|------------------------------|----------|----------|----------|-----------------------|-----------------------|
| | APP | APPOBJ | APPSEL | APPVAL | APPFLD | APPDAT |
| GET | Required | Optional | Optional | Optional | Optional | Not used |
| PUT | Required | Required | Required | Required | Required | Required |
| DEL | Required | Required | Required | Required | Not used | Not used |
| CREATE | Required | Required | Required | Required | Required ² | Required ² |

Note:

1. APP must be the first section in a buffer. There is no restriction on the order of other section types.

2. Not used for BACKUP_EVENT object.

Figure 5 is an example of the layout of a send buffer for a GET request. The arrows show the buffer parts that each section type points to. APP and APPOBJ point to related sections using triplet fields, which specify the offset, the length, and the number of the section type. APPSEL uses offset and length fields to point to an APPVAL section. All offsets are relative to the start of the buffer (offset 0).

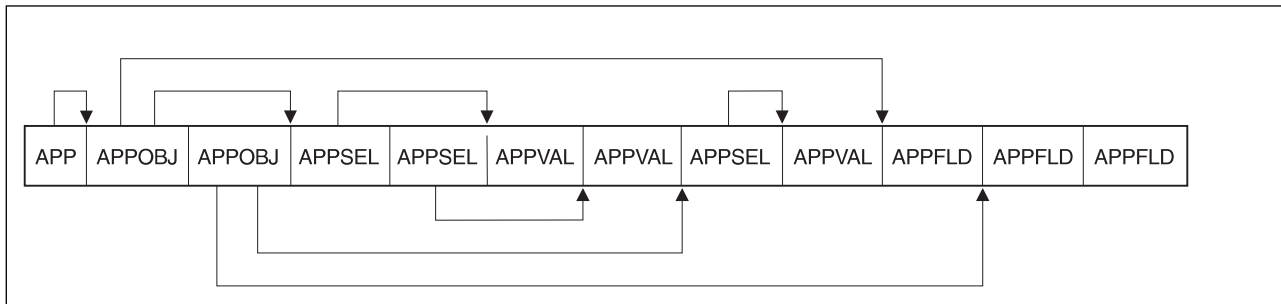


Figure 5. Example of a send buffer layout for a GET request

When a receive buffer is returned from IBM Workload Scheduler for z/OS, the buffer contains the entire send buffer. Some fields are updated by IBM Workload Scheduler for z/OS, for example, return codes and reason codes. For a GET request, data sections are also added if the requested information was found. One data section is added for each object instance found, and the data section triplet in APPOBJ is updated to point to the data.

If an error occurs during verification of the send buffer, IBM Workload Scheduler for z/OS returns a receive buffer that contains the whole of the send buffer unaltered, plus an additional APP section at the start of the buffer. This additional APP section is updated to indicate the error type.

Each buffer section is described here in more detail.

APP - Fixed section

The buffer that your program passes to IBM Workload Scheduler for z/OS must contain a fixed section, and it must be the first section in the buffer. It identifies the buffer, its size, the default request type, and points to object sections. The buffer must contain only 1 fixed section, even if multiple requests are passed in the same buffer.

The fixed section has this format:

Table 112. App-Fixed Section

| Offsets | | | | | |
|---------|-----|-----------|-----|---------|------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 80 | APP | APPC BUFFER MAPPING |
| 0 | (0) | CHARACTER | 4 | APPDESC | BLOCK DESCRIPTOR (APP) |
| 4 | (4) | CHARACTER | 2 | APPVER | VERSION NUMBER (02) |
| 6 | (6) | BITSTRING | 2 | * | RESERVED |

Table 112. App-Fixed Section (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|-----------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 8 | (8) | CHARACTER | 3 | APPTYPE | EYE CATCHER (DIA) |
| 11 | (B) | BITSTRING | 1 | APPFLAGS | RESERVED |
| 12 | (C) | SIGNED | 4 | APPTOTSZ | TOTAL SIZE |
| 16 | (10) | CHARACTER | 8 | APP_TYPE | DIALOG DATA TYPE (GET PUT DEL CREATE) |
| 24 | (18) | SIGNED | 4 | APP_RETCODE | *RETURN CODE |
| 28 | (1C) | SIGNED | 4 | APP_RSNCODE | *REASON CODE |
| 32 | (20) | | 12 | APP_OBJ_TRIPLET | OBJECT SECTION TRIPLET |
| 32 | (20) | SIGNED | 4 | APP_OBJ_OFF | OFFSET TO FIRST OBJECT SECTION |
| 36 | (24) | SIGNED | 4 | APP_OBJ_LEN | LENGTH OF AN OBJECT SECTION |
| 40 | (28) | SIGNED | 4 | APP_OBJ_NBR | NUMBER OF OBJECT SECTIONS |
| 44 | (2C) | SIGNED | 4 | APP_ERR_OFF | *OFFSET TO VERIFICATION ERROR |
| 48 | (30) | CHARACTER | 8 | * | RESERVED |
| 56 | (38) | CHARACTER | 16 | APPTOKEN | *TOKEN FIELD |
| 72 | (48) | CHARACTER | 8 | * | RESERVED |

Note: Descriptions prefixed with an asterisk (*) indicate fields that IBM Workload Scheduler for z/OS updates.

In the fixed section:

APPDESC

Is the block descriptor and has the value APP.

APPVER

Is the version number and has the value 02.

Note: You can continue to use existing buffers with version number 01, but you cannot include new requests or fields in these buffers.

* Offset 6 (X'6'). Set this reserved field to binary zeros (X'00')

APPTYPE

Is the eye catcher and has the value DIA.

APPFLAGS

Set this reserved field to binary zeros (X'00').

APPTOTSZ

Is the total size of the buffer.

APP_TYPE

Is the request type that is the default for all requests. It is used if you do not provide a value for APPOBJ_TYPE in an object section of the buffer. If you set this field to blanks (X'40'), you must specify a request in each object section of the buffer.

APP_OBJ_TRIPLET

Contains the offset to the first APPOBJ section, the length of all sections, and the number of sections. If the APP_OBJ_NBR field contains binary zeros (X'00') for a GET request, IBM Workload Scheduler for z/OS returns a *data dictionary*. The data dictionary is a description of all objects and all fields that the API supports for a GET request. CREATE objects are not described.

APP_RETCODE

Is the return code that is set by IBM Workload Scheduler for z/OS. In the send buffer, set this field to binary zeros (X'00'). For more information, see "Return codes and reason codes generated by IBM Workload Scheduler for z/OS" on page 96.

APP_RSNCODE

Is the reason code that is set by IBM Workload Scheduler for z/OS. In the send buffer, set this field to binary zeros (X'00'). For more information, see "Return codes and reason codes generated by IBM Workload Scheduler for z/OS" on page 96.

APP_ERR_OFF

Is set by IBM Workload Scheduler for z/OS when APP_RSNCODE indicates an error that has an offset associated with it. It is the offset in the buffer where a verification error was found. In the send buffer, set this field to binary zeros (X'00').

* Offset 48 (X'30'). Set this reserved field to binary zeros (X'00').

APPTOKEN

Is a value that your program can set to uniquely identify a buffer. It could be, for example, a time stamp. APPTOKEN can be useful if there is more than one active request from your ATP at a time.

* Offset 72 (X'48'). Set this reserved field to binary zeros (X'00').

APPOBJ - Object section

This section identifies the object and optionally the request type. The buffer must contain an object section for all requests except a GET request. A buffer can contain more than one object section, but all object sections must be in contiguous storage; that is, they must follow one another. The part of the buffer containing object sections is pointed to by the APP_OBJ_TRIPLET in the fixed section. APPOBJ itself points to APPSEL, APPFLD, and APPDAT sections if they are specified in a send buffer.

If your send buffer does not contain an object section for a GET request, that is, it contains only the fixed section, the buffer that IBM Workload Scheduler for z/OS returns contains a description of all objects and all fields that are supported by the API for a GET request.

The object section has this format:

Table 113. APPOBJ-Object Section

| Offsets | | | | | |
|---------|-----|-----------|-----|--------|---|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 84 | APPOBJ | OBJECT SECTION APPOBJ_PTR = ADDR(APP) + APP_OBJ_OFF |

Table 113. APPOBJ-Object Section (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|--------------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | | 24 | APPOBJ_ID | OBJECT IDENTIFIER |
| 0 | (0) | CHARACTER | 16 | APPOBJ_NAME | OBJECT NAME |
| 16 | (10) | CHARACTER | 8 | APPOBJ_KEY_TYPE | KEY TYPE |
| 24 | (18) | | 12 | APPOBJ_FLD_TRIPLET | FIELD SECTION TRIPLET |
| 24 | (18) | SIGNED | 4 | APPOBJ_FLD_OFF | OFFSET TO FIRST FIELD SECTION |
| 28 | (1C) | SIGNED | 4 | APPOBJ_FLD_LEN | LENGTH OF A FIELD SECTION |
| 32 | (20) | SIGNED | 4 | APPOBJ_FLD_NBR | NUMBER OF FIELD SECTIONS |
| 36 | (24) | | 12 | APPOBJ_SEL_TRIPLET | SELECTION SECTION TRIPLET |
| 36 | (24) | SIGNED | 4 | APPOBJ_SEL_OFF | OFFSET TO FIRST SELECTION SECTION |
| 40 | (28) | SIGNED | 4 | APPOBJ_SEL_LEN | LENGTH OF A SELECTION SECTION |
| 44 | (2C) | SIGNED | 4 | APPOBJ_SEL_NBR | NUMBER OF SELECTION SECTIONS |
| 48 | (30) | | 12 | APPOBJ_DAT_TRIPLET | DATA SECTION TRIPLET |
| 48 | (30) | SIGNED | 4 | APPOBJ_DAT_OFF | OFFSET TO FIRST DATA SECTION |
| 52 | (34) | SIGNED | 4 | APPOBJ_DAT_LEN | LENGTH OF ALL DATA SECTIONS |
| 56 | (38) | SIGNED | 4 | APPOBJ_DAT_NBR | NUMBER OF DATA SECTIONS |
| 60 | (3C) | CHARACTER | 8 | APPOBJ_TYPE | DIALOG DATA TYPE (GET PUT DEL CREATE) |
| 68 | (44) | SIGNED | 4 | APPOBJ_RET | *OBJECT LEVEL RETURN CODE |
| 72 | (48) | SIGNED | 4 | APPOBJ_RSN | *OBJECT LEVEL REASON CODE |
| 76 | (4C) | CHARACTER | 8 | APPOBJ_AUTH | *RACF AUTHORITY (READ or UPDATE) |

Note: Descriptions prefixed with an asterisk (*) indicate fields that IBM Workload Scheduler for z/OS updates.

In the object section:

APPOBJ_NAME

Identifies the object type. "Specifying object names" on page 93 describes valid names.

APPOBJ_KEY_TYPE

Is the key type. If you set this field to blanks (X'40'), a default value is used. "Specifying key types" on page 94 describes valid key types.

APPOBJ_FLD_TRIPLET

Contains the offset to the first APPFLD section, the length of each section, and the number of sections. If the APPOBJ_FLD_NBR field contains all binary zeros (X'00') for a GET request, IBM Workload Scheduler for z/OS returns all fields in the selected object instances.

APPOBJ_SEL_TRIPLET

Contains the offset to the first APPSEL section, the length of each section, and the number of sections. Set these fields to binary zeros (X'00') if there are no APPSEL sections.

APPOBJ_DAT_TRIPLET

Contains the offset to the first APPDAT section, the length of all sections, and the number of sections. Set these fields to binary zeros (X'00') if there are no APPDAT sections. IBM Workload Scheduler for z/OS updates these fields if data is returned for a GET request.

APPOBJ_TYPE

Is the request type for this object. If you set this field to blanks (X'40'), APP_TYPE determines the request type.

APPOBJ_RET

Is the object level return code that is set by IBM Workload Scheduler for z/OS. In the send buffer set this field to binary zeros (X'00'). For more information, see "Return codes and reason codes generated by IBM Workload Scheduler for z/OS" on page 96.

APPOBJ_RSN

Is the object level reason code that is set by IBM Workload Scheduler for z/OS. In the send buffer set this field to binary zeros (X'00'). For more information, see "Return codes and reason codes generated by IBM Workload Scheduler for z/OS" on page 96.

APPOBJ_AUTH

Is the access authority (read or update) that your ATP has to the specified object. For GET, PUT, and DEL requests, IBM Workload Scheduler for z/OS updates this field before the buffer is returned. It is not updated for a CREATE request. You could use APPOBJ_AUTH to establish your access by issuing a GET request for the object, before attempting further read or update requests. In the send buffer set this field to blanks (X'40').

APPSEL - Selection section

This section identifies a particular field in an object. By specifying a field name and a comparison operator in APPSEL, you can limit the instances of the object that IBM Workload Scheduler for z/OS finds. APPSEL is pointed to by the APPOBJ_SEL_TRIPLET in its object section and must itself point to an APPVAL section where a selection value is specified. To identify one particular instance of an object, you might need to specify more than one APPSEL in the send buffer. The selection sections for a particular APPOBJ must be in contiguous storage.

If you do not specify APPSEL for a GET request, IBM Workload Scheduler for z/OS returns all instances of the object.

The selection section has this format:

Table 114. APPSEL-Selection Section

| Offsets | | | | | |
|---------|------|-----------|-----|------------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 36 | APPSEL | SELECTION SECTION ADDRESS OF FIRST SELECTION SECTION FOR THIS OBJECT: APPSEL_PTR = ADDR(APP) + APPOBJ_SEL_OFF |
| 0 | (0) | CHARACTER | 16 | APPSEL_NAME | OBJECT FIELD NAME |
| 16 | (10) | CHARACTER | 2 | APPSEL_OPER | OPERATOR |
| 18 | (12) | CHARACTER | 10 | * | RESERVED |
| 28 | (1C) | SIGNED | 4 | APPSEL_VALUE_OFF | VALUE OFFSET |
| 32 | (20) | SIGNED | 4 | APPSEL_VALUE_LEN | VALUE LENGTH |
| | | | | | |

In the selection section:

APPSEL_NAME

Is a field name in the object.

APPSEL_OPER

Is a comparison operator.

* Offset 18 (X'12'). Set this reserved field to binary zeros (X'00').

APPSEL_VALUE_OFF

Is the offset to the APPVAL section.

APPSEL_VALUE_LEN

Is the length of the APPVAL section.

For more information, see "Selecting object instances" on page 94. Field names are described in Appendix B, "API object fields," on page 345.

APPVAL - Selection value section

This section contains a value that you want IBM Workload Scheduler for z/OS to search for within the object, according to the selection criteria that you specified in APPSEL. APPVAL is pointed to by APPSEL; it must be included if APPSEL is specified in the buffer. One APPVAL is required for each APPSEL. Selection value sections need not be in contiguous storage.

Each APPVAL section can contain only one value. If you specify GN (generic compare) in the APPSEL_OPER field, the selection value can contain the generic search arguments asterisk (*) and percent (%). An asterisk represents a character string or a null string. The percent sign represents a single character. See *IBM Workload Scheduler for z/OS: Managing the Workload* for a complete description of generic search arguments.

The selection value section has this format:

Table 115. APPVAL-Selection Value Section

| Offsets | | | | | |
|---------|-----|------------|-----|------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | * | APPVAL | DATA SECTION ADDRESS OF FIRST DATA SECTION FOR THIS OBJECT: APPVAL_PTR=ADDR(APP) + APPSEL_VALUE_OFF |
| 0 | (0) | (See note) | * | APPVAL_DAT | DATA |

Note: The field type depends on the object field name that you specify in APPSEL_NAME. See Appendix B, “API object fields,” on page 345.

APPFLD - Field section

For PUT and CREATE requests, each field section identifies a field in the selected object that you want to update; for example, the status of an operation in the current plan. APPFLD is not used for a CREATE request when the object name is BACKUP_EVENT, or for DEL requests.

For the GET request, you can use APPFLD sections to limit the data that is returned to particular object fields. You need supply only the APPFLD_NAME in a send buffer. IBM Workload Scheduler for z/OS updates the APPFLD_LEN and APPFLD_TYPE fields before the buffer is returned. If you do not specify APPFLD for a GET request, the buffer returned contains all fields in the selected instances of the object.

Field sections are pointed to by the APPOBJ_FLD_TRIPLET in the object section. You can specify more than one APPFLD for each APPOBJ, but all field sections for a particular APPOBJ must be in contiguous storage.

The field section has this format:

Table 116. APPFLD-Field Section

| Offsets | | | | | |
|---------|------|-----------|-----|-------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 24 | APPFLD | FIELD SECTION ADDRESS OF FIRST FIELD SECTION FOR THIS OBJECT: APPFLD_PTR=ADDR(APP) + APPOBJ_FLD_OFF |
| 0 | (0) | CHARACTER | 16 | APPFLD_NAME | FIELD NAME |
| 16 | (10) | SIGNED | 4 | APPFLD_LEN | FIELD LENGTH |
| 20 | (14) | CHARACTER | 4 | APPFLD_TYPE | *FIELD DATA TYPE |

Note: Descriptions prefixed with an asterisk (*) indicate fields that IBM Workload Scheduler for z/OS updates.

In the field section:

APPFLD_NAME

The name of the field. "Selecting object fields to update or retrieve" on page 96 describes the fields that you can specify for each object type.

APPFLD_LEN

The length of the field and is used in identifying the value in APPDAT for this field. For a GET request, or when the object is BACKUP_EVENT, set this field to binary zeros (X'00').

APPFLD_TYPE

The data type and is updated by IBM Workload Scheduler for z/OS before the buffer is returned. Set this field to blanks (X'40') in a send buffer.

APPDAT - Data section

For PUT and CREATE requests, APPDAT contains the new values for the fields identified in the APPFLD sections. Only one APPDAT must be specified for each APPOBJ. The values must be in the same order as the corresponding APPFLD sections.

For a GET request, data sections are found only in a receive buffer. IBM Workload Scheduler for z/OS returns in the receive buffer one data section for each instance of the object. Each APPOBJ section in the send buffer is updated by IBM Workload Scheduler for z/OS to point to associated data sections when the receive buffer is returned. The data sections are always the last sections in the receive buffer, and are returned in contiguous storage by object.

APPDAT is not used for DEL requests.

The data section has this format:

Table 117. APPDAT-Data Section

| Offsets | | | | | |
|---------|-----|------------|-----|------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | * | APPDAT | DATA SECTION ADDRESS OF FIRST DATA SECTION FOR THIS OBJECT: APPDAT_PTR=ADDR(APP) + APPOBJ_DAT_OFF |
| 0 | (0) | (See note) | * | APPDAT_DAT | DATA |

Note: The field type depends on the object field name that you specify in APPFLD_NAME or that IBM Workload Scheduler for z/OS retrieves. See Appendix B, "API object fields," on page 345.

Specifying object names

You identify the object type by specifying a name in the APPOBJ_NAME field of the object section. Table 118 describes the object names that you can specify:

Table 118. API Object Names

| Object | Valid requests | Description |
|--------------|----------------|------------------------|
| CP_STATUS | GET | Current plan status |
| CP_OPERATION | GET, PUT, DEL | Current plan operation |

Table 118. API Object Names (continued)

| Object | Valid requests | Description |
|------------------|----------------|---|
| CP_RESOURCE | GET | Current plan operation special resource |
| CP_WORK_STATION | GET | Current plan workstation (common part) |
| CP_OPEN_INTERVAL | GET | Current plan workstation open interval |
| CP_OPER_EVENT | CREATE | Current plan operation event. |
| CP_OPINFO_EVENT | CREATE | Current plan operation user data event |
| CP_SR_EVENT | CREATE | Current plan special resource event. |
| BACKUP_EVENT | CREATE | Backup event. |
| CP_WS_EVENT | CREATE | Current plan workstation event. |

Note:

1. You can add (PUT) an operation only to an existing application occurrence. You cannot add an occurrence through the API.
2. You cannot delete (DEL) an operation if it is the only operation in an occurrence. You cannot delete an occurrence through the API.

Selecting object instances

IBM Workload Scheduler for z/OS uses these criteria to identify object instances:

Key type

Identifies the relationship between the instances of an object that are located using the selection criteria, and the object instances that you want IBM Workload Scheduler for z/OS to find.

Selection field

The name of an object field that is used in locating instances of the object.

Selection value

The value in the selection field that is used in locating instances of the object.

Operator

A comparison operator that determines how the selection value is used in locating instances of the object.

Specifying key types

You can specify a key type in each object section of the buffer. If the key type field contains blanks (X'40'), a default is used. You can specify these key types:

SAME The objects found are those matching the selection criteria. This value is valid for, and is the default for, these objects:

CP_OPERATION
 CP_WORK_STATION
 CP_STATUS
 CP_OPER_EVENT
 CP_OPINFO_EVENT
 CP_SR_EVENT
 BACKUP_EVENT
 CP_WS_EVENT.

PRED The objects found are those that are predecessors to the object matching the selection criteria. This value is valid for the CP_OPERATION object but only with a GET request.

SUCC The objects found are those that are successors to the object matching the selection criteria. This value is valid for the CP_OPERATION object but only with a GET request.

OWNER

The objects found are those whose owner matches the selection criteria. This value is valid for, and is the default for, the objects CP_OPEN_INTERVAL (owner is CP_WORK_STATION) and CP_RESOURCE (owner is CP_OPERATION).

Specifying selection criteria

You specify selection criteria in the APPSEL and APPVAL sections of the buffer to limit the instances of an object that are located by IBM Workload Scheduler for z/OS. APPSEL contains a selection field name and a comparison operator that determines how the value for this field is used. You supply the field value in the APPVAL section. Appendix B, "API object fields," on page 345 describes the field names and field values of each object, and the selection type of each field. There are select fields:

Required

For a GET request with a key type of OWNER, PRED, or SUCC, you must specify these fields and the operator must be EQ to ensure that there is only one possible match. When the key type is SAME, these fields are optional.

For PUT and DEL requests, you must specify these fields. Also, the key type must be SAME and the operator EQ.

Optional

You can specify the field in the APPSEL section, but it is not required.

Not supported

You must not specify the field in the APPSEL section.

You can specify these operators in the APPSEL section:

Table 119. Operators That You Can Specify in the APPSEL Section

| Operator | Description |
|----------|--------------------------|
| EQ or = | Equal to |
| NE or ^= | Not equal to |
| GT or > | Greater than |
| LT or < | Less than |
| GE or >= | Greater than or equal to |
| LE or <= | Less than or equal to |
| GN | Generic compare |

Broadcasting events

The LU that your ATP sends requests to is owned by an IBM Workload Scheduler for z/OS address space where the APPC subtask is started. When you send a CREATE request to the LU, the address space processes the request and creates an event. If you want to report an event to more than one IBM Workload Scheduler

for z/OS address space, or an event writer is not started in the address space that owns the target LU, you must broadcast the event.

To broadcast an event, specify `SUBSYSTEM_NAME` in the `APPSEL` section but do not provide a name in the `APPVAL` section, or provide the name `MSTR` in `APPVAL`. The event is sent using the subsystem interface (SSI) to all IBM Workload Scheduler for z/OS address spaces started on the same z/OS image as the target.

Selecting object fields to update or retrieve

You select object fields to update or retrieve by specifying values in the `APPFLD` and `APPDAT` sections of a buffer.

The `APPFLD` section identifies an object field. For `GET` requests, `APPFLD` identifies a field in each located object instance that you want IBM Workload Scheduler for z/OS to return in the receive buffer. For `PUT` and `CREATE` requests, `APPFLD` identifies the field that you want to update. `APPFLD` is not used for `DEL` requests or when the object for a `CREATE` request is `BACKUP_EVENT`.

For a `GET` request, the `APPDAT` section is not used in a send buffer. `APPDAT` sections are returned in a receive buffer if data is found. For `PUT` and `CREATE` requests, `APPDAT` contains the new values for the fields identified in `APPFLD` sections. You must specify only 1 `APPDAT` per `APPOBJ`.

Appendix B, “API object fields,” on page 345 describes the fields that you can update or retrieve.

Return codes and reason codes generated by IBM Workload Scheduler for z/OS

If a request through the API causes a severe error in an IBM Workload Scheduler for z/OS subtask, you receive one of these CPI-C return codes:

`CM_PROGRAM_ERROR_NO_TRUNC`
`CM_PROGRAM_ERROR_PURGING`.

The conversation is deallocated, and CPI-C return code `CM_RESOURCE_FAILURE_NO_RETRY` is set. Here, do not resend the buffer to IBM Workload Scheduler for z/OS until problem determination establishes a reason for the previous error. For information about CPI-C return codes, refer to *CPI-C Communications Reference*.

Besides CPI-C return codes, IBM Workload Scheduler for z/OS can generate return codes and reason codes for the various requests that are made. Your program can test the results of the call to IBM Workload Scheduler for z/OS by inspecting return codes and reason codes in the `APP` and `APPOBJ` sections of the buffer.

Return codes and reason codes generated in the fixed section (APP)

A buffer always starts with a fixed section. Return codes and reason codes are generated in the fixed section when IBM Workload Scheduler for z/OS validates the buffer. The `APP_RETCODE` field can contain one of these codes:

0 Execution successful.

- 4 Execution successful but no data was returned. Either there was no data that matched the GET request, or the ATP is not authorized to access the data matching the GET request.
- 12 Execution unsuccessful; the buffer is invalid. IBM Workload Scheduler for z/OS has not attempted to process the request. A receive buffer is created that contains an APP control block followed by the entire send buffer. No updates are made to any fields in the send buffer. So this special receive buffer will start with 2 APP sections.

The APP_RSNCODE field can contain one of these codes:

- 0 Execution successful.
- 4 Buffer shorter than APP.
- 8 Eye catcher in APPDESC field is invalid. It must be APP.
- 12 Version number in APPVER field is invalid. It must be 02.
- 16 Type in APPTYPE field is invalid. It must be DIA.
- 20 APPTOTSZ invalid.
- 24 Data type invalid. Specify GET, PUT, DEL, or CREATE.
- 28 Object section not within buffer.
- 32 Object section overlays APP.
- 36 Selection section not within buffer.
- 40 Selection section overlays APP or object section.
- 44 Field section not within buffer.
- 48 Field section overlays APP or object section.
- 52 Required field not supplied.
- 56 Invalid object name in OBJ section.
- 60 Invalid field name in FLD section.
- 64 Invalid field name in SEL section.
- 68 APPTOKEN value invalid (duplicate).

Return codes and reason codes generated in the object section (APPOBJ)

The return codes and reason codes generated in the object section indicate an error after IBM Workload Scheduler for z/OS validated the buffer. No return and reason codes are generated in the object section for CREATE requests. For GET, PUT, and DEL requests, the APPOBJ_RET field can contain one of these codes:

- 0 Execution successful.
- 12 Execution unsuccessful.

The APPOBJ_RSN field can contain one of these codes:

- 0 Execution successful.
- 4 The operation does not exist.
- 8 An invalid update was attempted.
- 12 A security violation occurred.
- 16 An error was detected. For more information, check the message log (EQQMLOG) of the IBM Workload Scheduler for z/OS address space that the request was sent to.

Security

The access to IBM Workload Scheduler for z/OS can be controlled through security mechanisms provided by:

- APPC and RACF
- IBM Workload Scheduler for z/OS and RACF.

APPC and RACF

The APPC security mechanism provides access control in these areas:

- Access to logical units (LUs)
- Access control for LU to LU communication
- Access to transaction programs
- Security within the network.

IBM Workload Scheduler for z/OS recognizes these TP names:

EQQTRK

Supplied by trackers that communicate with the controller through APPC

EQQAPI

Supplied by user programs (ATPs) that communicate with IBM Workload Scheduler for z/OS through the API.

Refer to *APPC Management* for a detailed description of how to protect your APPC environment.

Refer to *ICSF/MVS™ Programmer's Guide* for a detailed description of how to protect information that crosses the network.

IBM Workload Scheduler for z/OS and RACF

IBM Workload Scheduler for z/OS performs security checking at the controller for GET, PUT, and DEL requests, for all ATPs that use the API. To establish a conversation, your ATP must supply a user ID and password, and optionally a profile that indicates the RACF user group. The user ID must have the required level of access.

For CREATE requests, IBM Workload Scheduler for z/OS does not perform security checking, because the request could be directed to more than one IBM Workload Scheduler for z/OS subsystem where security rules differ. You can prevent unauthorized use of CREATE requests through APPC security mechanisms by protecting the LU and the TP name.

You can protect access to IBM Workload Scheduler for z/OS resources at these levels:

1. The IBM Workload Scheduler for z/OS subsystem resource
2. Fixed resources
3. Subresources.

Access at one level determines the default access to the next level. The default is used if the required resource is not protected at the following level. To use the API, you must have at least read access to the IBM Workload Scheduler for z/OS subsystem, which is defined in the APPL class. GET, PUT, and DEL requests require this access to fixed resources:

GET CP read. SR read is also required to retrieve special resource information.

PUT CP update is required for CP_OPER_EVENT, CP_OPINFO_EVENT, and

CP_WS_EVENT. Additionally, EXEC update is required to request the EXEC command. BKP update is required for BACKUP_EVENT.

DEL Requires the same access as PUT.

You can further restrict access by specifying subresources, which are described in Table 120.

Table 120. Subresource Protection for Requests through the API

| Fixed resource | Subresource | Description |
|----------------|-------------|---------------------------------------|
| CP | CP.ADNAME | Application name |
| | CP.GROUP | Application authority group ID |
| | CP.JOBNAME | Operation job name |
| | CP.OWNER | Application owner |
| | CP.WSNAME | Workstation name |
| | CP.ZWSOPER | Workstation name used by an operation |
| | CP.CPGDDEF | Group definition ID name |
| RL | RL.ADNAME | Occurrence name |
| | RL.OWNER | Occurrence owner ID |
| | RL.GROUP | Occurrence authority-group ID |
| | RL.WSNAME | Current-plan workstation name |
| SR | SR.SRNAME | Special resource name |

Note: If you restrict access at the subresource level, selection criteria will find only those instances of an object that both match the selection criteria and that the user ID has access to.

If a request is denied for READ access to the IBM Workload Scheduler for z/OS subsystem resource or to a fixed resource, you receive CPI-C return code CM_SECURITY_NOT_VALID and the conversation is deallocated. Other security failures result in an error buffer with reason code 512 and the conversation remains allocated.

For a detailed explanation of security considerations, see *IBM Workload Scheduler for z/OS: Customization and Tuning*.

Part 2. Programming tools

Chapter 3. Batch command interface tool

About this task

Using IBM Workload Scheduler for z/OS you can control and automatically plan your production workload in your complex. You can use the program interface to issue various types of requests to the IBM Workload Scheduler for z/OS subsystem. The program interface supports different requests to read and update resources in IBM Workload Scheduler for z/OS databases. The resources can be:

- Operations and their dependencies
- Applications
- Operator instructions
- Calendars
- Periods
- Run cycle groups
- Workstations
- The current and long-term plans

Online tools

About this task

To update records in IBM Workload Scheduler for z/OS databases, you can use ISPF dialogs. Online administration is performed on the controlling system (the dialog is available on the system in your configuration that is running the controller).

The batch command interface

About this task

The Batch Command Interface is a sample program that you can run in batch and that issues various types of request to the IBM Workload Scheduler for z/OS subsystem.

Input to batch command interface

About this task

It is a PIF application. Note that dates provided in the INPUT must be in real format. If the PIFCWB and PIFHD parameters of the INTFOPTS initialization statement of the controller do not follow this rule, the EQQYPARM DD card invoking the BCIT must point to a member where INIT specifies the following:

- CWBASE (00)
- HIGHDATE (711231)

When the Batch Command Interface is called, it invokes PGM=EQQYCAIN, with an optional PARM field (the sample member EQQYCBAT contains an example of how this program can be invoked). The positional parameters that can be passed with the PARM field are as follows:

Table 121. Positional parameters that can be passed with the Batch Command Interface

| Description | Maximum length | Default | Values |
|----------------|----------------|--------------|-------------------|
| SUBSYSTEM NAME | 4 | OPCA | product name |
| WTO desired | 6 | MSGON | MSGOFF, MSGNONE |
| IA date | 6 | current date | CPSTDA/ yymmdd |
| IA time | 4 | current time | hhmm |
| CP needed | 6 | MUSTCP | MISSCP |

EXAMPLES:

PARM='OPGQ, MSGOFF, CPSTDA, 1400'

PARM='OPGQ,,991207'

PARM='OPGQ,,,MISSCP'

The meaning of the keywords is the following:

CPSTDA

Means that the default IA date is the starting date of the CP.

MISSCP

Means that the requested BCIT functions do not require the existence of a CP and that BCIT should not try to access it.

MSGNONE

Means that WTOs (including those issued in case of errors or for commands that do not have performance impacts) are not to be issued any longer.

MSGOFF

Means that WTOs (except those issued in case of errors or for commands that do not have performance impacts) are not to be issued any longer.

MSGON

Means that WTOs are to be issued in any case. It is strongly recommended not to use the default MSGON when the BCIT commands invoked might produce too many WTOs.

MUSTCP

Requests that BCIT accesses the CP, no matter if the requested functions need the existence of CP.

| A set of actions, called a program, is specified in a file SYSIN referenced by the DD
| card: //SYSIN DD. If the SYSIN DD card refers to a DASD file, ensure that the file
| is defined with RECFM=FB and LRECL=80.

A program with instructions is input for this sample.

An instruction is an action on a resource with any arguments to identify it and other arguments to process it.

An argument has two parts; the right part is its identification and the left part is its value. These two parts are separated by the equal sign (=). An argument is finished by a comma (,) or semicolon (;) if this argument is the last of the instruction, and by a period if it is the last of the program. The blanks before the identifier are

ignored but the blanks after the identifier are used. A blank is a character, not a delimiter, if it is before an argument or a value.

An instruction must be finished by a semicolon(;). Only the last instruction of the program is finished by a period (.). A program must be finished by a period (.).

The descriptive fields, such as DESC and EXTNAME, can include special characters only if the field is inserted within single quotation marks. For example,

```
DESC='XXXX.AAAAA,BBBBB;'
```

Note: If you need to use delimiters, such as single quotation marks, in descriptive fields follow these rules:

- To use the single quotation mark within a field, you must use two quotation marks.
- To use the single quotation mark at the beginning or at the end of a field, you must use three quotation marks.

Example of program:

```
ACTION=OPTIONS,BL=N,LTP=N;
ACTION=LIST,RESOURCE=ADCOM,ADID=description,IA=9202121500;
ACTION=LIST,RESOURCE=ADCOM,ADID=A%%B*,IA= 9202121500,
    PRIORITY=5;
ACTION=SELECT,
    RESOURCE=OICOM,
    ADID=ADABASE;
ACTION=OPTIONS,BL=Y;
ACTION=LIST,RESOURCE=LTOCCOM,
    ADID=ADABASE;
ACTION=COPY,RESOURCE=AD,ADID=ADABASE,STATUS=A,VALTO=991231,
    NADID=MATENCIOP,NSTATUS=P;
ACTION=INSERT,RESOURCE=LTPRE,ADID=MATENCIOP,IAD=920215,IAT=0915,
    PREADID=ADABASEOP,PREIAD=920215,PREIAT=0915;
ACTION=LIST,RESOURCE=LTOCCOM,ADID=MATENCIOP;
ACTION=INSERT,RESOURCE=LTOC,ADID=MATENCIOP,IAD=920215,IAT=0915,
    DEADLINE=9202150916;
ACTION=DELETE,RESOURCE=CPPRE,ADID=MATENCIOP,IA=9202062343,OPNO=30,
    PREADID=ADABASE,PREOPNO=10,PREIA=9201310700;
ACTION=LIST,RESOURCE=CPOPCOM,ADID=MATENCIOP,OPNO=30;
ACTION=MODIFY,RESOURCE=CPOP,ADID=MATENCIOP,IA=9202062343,OPNO=30,
    JOBNAME=MATENCIO,DESC='TEST MODIFY CPOP',EDUR=0100,PSUSE=1,R1USE=1,
    R2USE=1,JCLASS=B,OPIA=9202062344,OPDL=9202062345,WSNAME=WSTC,
    STATUS=A;
ACTION=MODIFY,RESOURCE=CPEXT,ADID=MATENCIOP,IA=9202062343,OPNO=30,
    EXTNAME='Operation Extended Name';
ACTION=MODIFY,RESOURCE=CPOC,ADID=MATENCIOP,IA=9202062300,
    IANEW=9202062343,DEADLINE=9202062345;
ACTION=INSERT,RESOURCE=CPPRE,ADID=MATENCIOP,IA=9202042342,OPNO=25,
    PREADID=ADABASE,PREOPNO=10,PREIA=9201310700;
ACTION=DELETE,RESOURCE=LTOC,ADID=MATENCIOP,IAD=920129,IAT=0700;
ACTION=IMPORT,RESOURCE=AD,ADID=MATENCIOP;
ACTION=INSERT,RESOURCE=AD,ADID=MATENCIOP;
ACTION=DELETE,RESOURCE=AD,ADID=MATENCIOP;
ACTION=SELECT,RESOURCE=AD,ADID=MATENCIOP;
ACTION=COPY,RESOURCE=OI,ADID=ADABASE,OPNO=10,
    NADID=MATENCIOP,NOPNO=10;
ACTION=INSERT,RESOURCE=CPOC,ADID=MATENCIOP,IA=9201291500;
ACTION=INSERT,RESOURCE=CPSR,ADID=MATENCIOP,IA=9201291500,RESNAME=name,
    RESUSAGE=X,OPNO=10,QUANTITY=1;
ACTION=INSERT,RESOURCE=CPOP,ADID=MATENCIOP,IA=9202042342,OPNO=25,
    JOBNAME=MATENCIO,WSNAME=FTW1,EXTNAME='Operation Extended Name'.
```

DD CARD EQQMLIB contains the scheduler messages.

Note: In the above example, for simplicity many commands are included in the same SYSIN, but this is not a recommended practice. Because BCIT processes all the statements contained in the SYSIN and does not stop processing if any statement fails (unless ERROR=Y is specified in the OPTIONS card), it is recommended to avoid inserting commands which should not be run in case of failure in one of the previous commands, or to specify ERROR=Y in the OPTIONS card.

BCIT output

About this task

Each instruction is processed and a return code is displayed in the LOG. Depending on the action and options, the outputs are different, and are written in different files. The different output files are:

AD If arguments BL=Y and BLPRT=N are coded in the action OPTIONS, the identifier, status, and VALTO of each listed application (AD) will be written in the file referenced by the AD DD card (LRECL=23).

BATCHL

If arguments BL=Y and BLPRT=Y are coded in the action, LIST and SELECT of ADs and OI results will be formatted as BATCH LOADER size and written in referenced files by the DD BATCHL card.

CPCOND

If argument BL=Y is coded in the OPTIONS action, the ACTION=LIST,RESOURCE=CPCONDCO,ADID=XXXX* result is written in the file referenced by CPCOND card (LRECL=80). It is the same for action=select.

CPOC If argument BL=Y is coded in the OPTIONS action, the ACTION=LIST,RESOURCE=CPOC,ADID=XXXX* result is written in the file referenced by CPOC card (LRECL=80). It is the same for action=select.

CPOP If argument BL=Y is coded in the OPTIONS action, the ACTION=LIST,RESOURCE=CPOPCOM,ADID=XXXX* result is written in the file referenced by CPOP card (LRECL=80). It is the same for action=select.

DATAFI

File that contains the output of LIST JCLVCOM

EQQDUMP

File that contains information to understand diagnostic data set error codes.

EQQMLOG

File that contains the scheduler messages for return code of 6 or higher.

EXPORTAD

If the action is an EXPORT and the resource is an AD, the file will contain structured AD segments as well, as it is easier to import them.

EXPORTOI

If the action is an EXPORT and the resource is an OI, the file will contain structured OI segments as well, as it is easier to import them.

EXPORTRG

If the action is an EXPORT and the resource is an RG, the file will contain structured RG segments as well, as it is easier to import them.

IMPORTAD

If the action is an IMPORT and the resource is an AD, the file will contain AD segments to be imported.

IMPORTOI

If the action is an IMPORT and the resource is an OI, the file will contain OI segments to be imported.

IMPORTRG

If the action is an IMPORT and the resource is an RG, the file will contain RG segments to be imported.

OI If arguments BL=Y and BLPRT=N are coded in the action OPTIONS, the identifier and operation number of each listed operator instruction(OI) will be written in the file referenced by the OI DD card (LRECL=23).

RG If arguments BL=Y and BLPRT=N are coded in the action OPTIONS, the identifier of each listed run cycle group (RG) will be written in the file referenced by the RG DD card (LRECL=23).

SYSPRINT

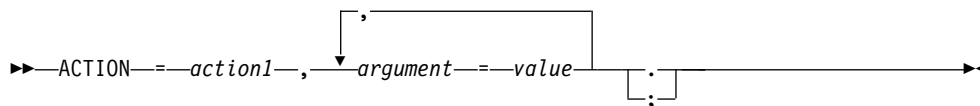
Required, it is used to print results of LIST and SELECT actions.

Instructions

About this task

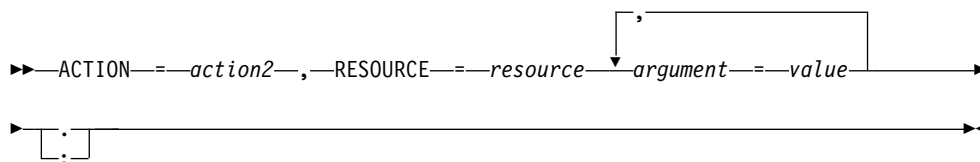
There are two sorts of instructions:

- The first does not refer to a resource and the action is still active during the whole program until the next instruction of the same type.



action1 OPTIONS

- The second refers to a resource and arguments that define this resource, and some active arguments. Some instructions are independent in their execution.



action2 LIST | SELECT | INSERT | DELETE | MODIFY | COPY | EXPORT | IMPORT | LISTSTAT

Return codes

The batch command interface returns a return code for each processed instruction, and the program keeps the highest return code.

Code Explanation

- 0 All instructions are successfully processed.
- 2 End of file, a forgotten period at the end of program or empty file.

- 4 Resource not found, or not authorized, or the operation input arrival is earlier than the occurrence input arrival.
- 5 One or more of the related dependencies does not exist.
- 6 Refer back to instruction return code.
- 8 Refer back to instruction return code.
- 10 More than one occurrence matches the specified arguments.
- 32 Input arrival date less than CP end date (applies to DELETE LTOC, DELETE LTPRE, INSERT LTOC, and MODIFY LTOC).
- 100 The field of the parm operand, specified in the preceding WTO, is wrong.
- 105 Error during the load of the EQQYCTDI interface module.
- 110 Syntax error (see the following message).
- 120 A DD card is missing (see the log for more information about which card is missing).
- 130 No SYSIN DD card found.
- 140 Exceeded the word capacity (54 characters). For example, this might occur if in the last card of the SYSIN file there is a period and nothing else.
- 150 Options incompatible with the command.
- 160 None of the input records match the value specified in ADID.
- 170 Not enough storage available.
- 180 The file input to the import function has been produced with a level of code not compatible.
- 190 MATCHTYP not specified and VALTO or VALFROM specified.
- 191 VALFROM or VALTO specified and one (or more) permanent OI(s) found among those selected.
- 195 MATCHTYP specified in a DELETE AD or DELETE OI command, without VALFROM and VALTO.
- 200 Current plan does not exist.
- 210 The input arrival date and time are not valid. See the messages for more information.
- 300 BCIT received invalid data from the scheduler.
- 310 Erroneous status code.
- 400 ADVERS=Y and NADID omitted requires that at least one between NSTATUS and NVALFROM be specified with a new value.
- 401 ADVERS=Y and NADID specifying a name for which there are already existing versions requires that both NSTATUS and NVALFROM be specified and their combination must not exist.
- 402 ADVERS=N and NADID omitted does not allow new arguments different from NSTATUS to be specified.
- 403 ADVERS=N and NADID omitted requires NSTATUS to specify a value different from the status of the ADID being copied.
- 404 ADVERS=N and NADID specifying a name for which there are already existing versions is not supported.
- 405 COPY AD not allowed because maximum number of versions already exists.
- 406 NVALTO specified in COPY AD when ADVERS=Y is active.

COPY

The COPY instruction copies an AD, JCLV, OI, or RG, or any combination of AD, JCLV, OI, and RG to the same IBM Workload Scheduler for z/OS subsystem with some argument change. This function quickly creates an AD, JCLV, OI, or RG from an extract segment. It is useful if you want to modify some arguments, such as ADID, RGID, STATUS, IA. You can use the batch command interface to copy JCLV and change only the JCLV name.

COPY AD

Copy an application, changing arguments. New arguments begin with N.

ACTION=COPY,RESOURCE=AD,STATUS=*ad_status*,
 ADID=*application_description*,
 GROUP=*authority_group_name*,
 NSTATUS=*new_adstatus*,
 NADID=*new_application_description*,
 NGROUP=*new_authority_group_name*,
 NOWNER=*new_owner*,
 NVALFROM=*new_valid_from_date*,
 NVALTO=*new_valid_to_date*,
 VALTO=*valid_to_date*;

adstatus

P or A (char(1))

application_description

Character (char(16))

authority_group_name

Character (char(8))

new_adstatus

P or A (char(1))

new_application_description

Character (char(16))

new_authority_group_name

Character (char(8))

new_owner

Character (char(16))

new_valid_from_date

Date(YYMMDD)

Note: The following rules apply when this command is run with the option ADVERS=Y active: NVALTO cannot be specified. If NADID is omitted, then NSTATUS must be specified and differ from the status of the ADID being copied or NVALFROM must be specified and differ from any valid from date of the ADID being copied. If NADID is specified and there is no existing version in the AD database, no restriction applies. If NADID is specified and there are existing versions in the AD database, both NSTATUS and NVALFROM must be specified and their combination must differ from any existing combination of NADID.

The following rules apply when this command is executed with the option ADVERS=N active: If NADID is omitted, the only new argument allowed is NSTATUS, which is also mandatory and must differ from the status of the ADID being copied. If NADID is specified and there is no existing version in the AD database, no restriction applies. If NADID is specified and there are existing versions in the AD database, the copy is not allowed.

new_valid_to_date

Date (YYMMDD)

valid_to_date

Date (YYMMDD)

COPY JCLV

Copy a JCL variables table. New arguments begin with N.

ACTION=COPY,RESOURCE=JCLV,JCLVTAB=*jcl_variable_table_id*,
 NJCLVTAB=*new_jcl_variable_table_id*;

jcl_variable_table_id

Character (char(16)) can be generic (%*)

new_jcl_variable_table

Character (char(16)) can be generic (%*)

Return Codes (for COPY requests):

- 0 Instruction is successfully processed.
- 4 Resource not found or user ID does not have RACF authorization.
- 8 Instruction failed and an error message is written to the EQQMLOG file.
- >8 Refer to "Return codes" on page 107.

COPY OI

Copy an operation instruction with arguments change. New arguments begin with N.

```
ACTION=COPY,RESOURCE=OI,ADID=application_description,
      NADID=new_application_description,
      NJOBNAME=new_jobname,
      NOPNO=new_operation_number,
      OPNO=operation_number;
```

application_description

Character (char(16))

new_application_description

Character (char(16))

new_jobname

Character (Char (8))

new_operation_number

Integer (integer(4))

operation_number

Integer (integer(4))

COPY RG

Copy a run cycle group, changing arguments. New arguments begin with N.

```
ACTION=COPY,RESOURCE=RG,RGID=run_cycle_group,
      NRGID=new_run_cycle_group,
      NRGOWNER=new_owner;
```

run_cycle_group

Character (char(8))

new_run_cycle_group

Character (char(8))

new_owner

Character (char(16))

DELETE

The DELETE instructions deletes a record from the database, current plan or long term plan.

AD An application description.

CPCOND

Current plan condition.

CPOC An occurrence to current plan.

CPOCPRE

A predecessor of a current plan occurrence. The predecessor can be an occurrence or an external operation.

CPOP An operation to current plan.

CPPRE

A predecessor of a current plan operation. The predecessor operation must be external.

CPSIMP

Current plan condition dependency.

CPSR A special resource to a current plan operation.

JS A JCL from the JCL repository.
JSCOM Multiple JCL entries from the JS file.
LTCPRE A conditional predecessor to long-term plan occurrence.
LTOC An occurrence to long-term plan.
LTPRE A predecessor to long-term plan occurrence.
OI Operator instructions.
RG A run cycle group.

DELETE AD

Suppress an application.

```

ACTION=DELETE,RESOURCE=AD,STATUS=adstatus,
ADID=application_description,
GROUP=authority_group_name,
GROUPDEF=group_definition,
MATCHTYP=match_type,
OWNER=owner,
PRIORITY=priority,
TYPE=type_of_ad,
VALFROM=valid_from_date,
VALTO=valid_to_date;
  
```

adstatus

A or P (char(1))

application_description

Character (char(16)) can be generic (%*)

authority_group_name

Character (char(8)) can be generic (%*)

group_definition

Character (char(16)) can be generic (%*)

match_type

Character (char(4)) can be generic (%*)

owner Character (char(16)) can be generic (%*)

priority

One integer (integer(4))

type_of_ad

A or G or * (char(1))

valid_from_date

Date (YYMMDD)

valid_to_date

Date (YYMMDD)

Note: MATCHTYP can have the following values: EXA, LOW, or HIGH. If you specify MATCHTYP, you must specify at least VALTO or VALFROM. If MATCHTYP=EXA, then only the version with validity period exactly matching the specified value is deleted. If MATCHTYP=LOW, then all the versions with validity from or validity to lower or equal to the value specified by VALFROM or VALTO are deleted. If MATCHTYP=HIGH, then all the versions with validity from or validity to higher or equal to the value specified by VALFROM or VALTO are deleted.

The validity periods of the versions not deleted will remain unchanged (as default) or will be adjusted to preserve the same logic used by the ISPF dialog if the command is run with ADVERS=Y specified in the last OPTIONS card.

DELETE CPCOND

Suppress a current plan condition.

```
ACTION=DELETE,RESOURCE=CPCOND,ADID=application_description,
IA=input_arrival_datetime,
OPNO=operation_number,
CONDID=condition_ID;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time: YYMMDDHHMM

operation_number

Integer (integer(4))

condition_ID

Integer (integer(3))

Specify all the operands (the complete condition key).

DELETE CPOC

Suppress a current plan occurrence.

```
ACTION=DELETE,RESOURCE=CPOC,ADID=application_description,
IA=input_arrival_datetime,
OSTATUS=oc_status;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time (YYMMDDHHMM) or 'X'

oc_status

Occurrence status: C, D, E, P, S, U, or W (char(1))

Note:

1. ADID and IA arguments identify an occurrence. If the value of IA (IA=X) is unknown, the OSTATUS argument completes the search for the occurrence. If there is more than one occurrence, the Batch Command Interface tool returns a return code of 10, and the occurrence is not modified. If IA is not specified as X, then OSTATUS is ignored.
2. If LTP=Y has been specified in the options card and the IA date specified or defaulted is higher than the CP end date, the occurrence will be deleted in the LTP. Of course, if IA=X is coded, the option LTP=Y is meaningless, so BCIT will only search if in the CP there is a single occurrence in the given OSTATUS.

DELETE CPOCPRE

Suppress a predecessor to a current plan occurrence.

```
ACTION=DELETE,RESOURCE=CPOCPRE,ADID=application_description,
IAD=input_arrival_date,
IA=input_arrival_datetime,
IAT=input_arrival_time,
PREADID=pre_application_description,
PREIAD=pre_input_arrival_date,
PREIA=pre_input_arrival_datetime,
PREIAT=pre_input_arrival_time,
PREOPNO=pre_operation_number;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_date

Date (YYMMDD)

input_arrival_datetime

Date and time (YYMMDDHHMM)

input_arrival_time
 Time (HHMM)
pre_application_description
 Character (char(16)) can be generic (%*)
pre_input_arrival_date
 Date (YYMMDD)
pre_input_arrival_datetime
 Date and time (YYMMDDHHMM)
pre_input_arrival_time
 Time (HHMM)
pre_operation_number
 Integer (integer(4))

ADID and IA arguments identify an occurrence.

The predecessor application is identified by PREADID, its occurrence by PREIA (if it is an occurrence predecessor) and its operation by PREOPNO (if it is an operation predecessor).

The operands ADID and PREADID are mandatory if the predecessor is external. The input arrival date and time is mandatory information, but it can be supplied in different ways:

1. Specifying IA in the normal way.
2. Specifying IA by entering 'YYMMDD' in place of the first six characters; the code will substitute them with the current date.
3. Specifying IAD and IAT separately.
4. Specifying only IAT; in which case the code will take the current date as the default for IAD.

If IA is omitted, then at least IAT is mandatory. The predecessor input arrival date and time is mandatory information only when there is an external predecessor. Also in this case it can be entered in different ways:

1. Specifying PREIA in the normal way.
2. Specifying PREIA by entering 'YYMMDD' in place of the first six characters; the code will substitute them with the current date.
3. Specifying PREIAD and PREIAT separately.
4. Specifying only PREIAT; in which case the code will take the current date as the default for PREIAD.

If PREIA is omitted and it is an external predecessor, then at least PREIAT is mandatory.

DELETE CPOP

Suppress an operation from current plan occurrence.

```

ACTION=DELETE,RESOURCE=CPOP,ADID=application_description,
IA=input_arrival_datetime,
OSTATUS=oc_status,
OPNO=operation_number;
  
```

application_description
 Character (char(16)) can be generic (%*)
input_arrival_datetime
 Date and time (YYMMDDHHMM) or 'X'
oc_status
 Occurrence status. C, D, E, P, S, U, or W (char(1)).

operation_number

Integer (integer(4))

Note: ADID and IA arguments identify an occurrence. If the value of IA (IA=X) is specified as 'X' (IA=X), the following might occur:

- There is only one occurrence of the given ADID: in this case it is not needed to specify OSTATUS.
- There is more than one occurrence, but only one in a specified status: in this case it is necessary to specify OSTATUS.
- There is more than one occurrence in a given status: in this case IA=X cannot work because adding OSTATUS is not enough to identify the occurrence and IA must contain exactly the input arrival date and time of the occurrence for which the deletion of the operation is requested.

If more than one occurrence matches the specified arguments, the BCI tool returns a return code of 10 and the operation will not be deleted. The OPNO argument identifies the operation with the occurrence. If IA is not specified as X, then OSTATUS is ignored. It is not possible to use DELETE CPOP if there is only one operation in the occurrence.

DELETE CPPRE

Suppress a predecessor to a current plan operation.

```
ACTION=DELETE,RESOURCE=CPPRE,ADID=application_description,
      IAD=input_arrival_date,
      IA=input_arrival_datetime,
      IAT=input_arrival_time,
      OPNO=operation_number,
      PREADID=pre_application_description,
      PREIAD=pre_input_arrival_date,
      PREIA=pre_input_arrival_datetime,
      PREIAT=pre_input_arrival_time,
      PREMAND=is_predecessor_mandatory,
      PREOPNO=pre_operation_number;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_date

Date (YYMMDD)

input_arrival_datetime

Date and time (YYMMDDHHMM)

input_arrival_time

Time (HHMM)

is_predecessor_mandatory

Character (char(1)) can be Y or N

operation number

Integer (integer(4))

pre_application_description

Character (char(16)) can be generic (%*)

pre_input_arrival_date

Date (YYMMDD)

pre_input_arrival_datetime

Date and time (YYMMDDHHMM)

pre_input_arrival_time

Time (HHMM)

pre_operation number

Integer (integer(4))

ADID and IA arguments identify an occurrence, OPNO identifies an occurrence operation.

The predecessor application is identified by PREADID, its occurrence by PREIA, and its operation by PREOPNO. If the predecessor is mandatory, set PREMAND to Y (the default is N).

|
|
|

The operands ADID and OPNO are mandatory. The PREOPNO argument is required only for an operation that depends on another operation. PREADID is mandatory if the predecessor is external.

The input arrival date and time is mandatory information, but it can be supplied in different ways:

1. Specifying IA in the normal way.
2. Specifying IA by entering 'YYMMDD' in place of the first six characters; the code will substitute them with the current date.
3. Specifying IAD and IAT separately.
4. Specifying only IAT; in which case the code will take the current date as the default for IAD.

If IA is omitted, then at least IAT is mandatory. The predecessor input arrival date and time is mandatory information only when there is an external predecessor. Also in this case it can be entered in different ways:

1. Specifying PREIA in the normal way.
2. Specifying PREIA by entering 'YYMMDD' in place of the first six characters; the code will substitute them with the current date.
3. Specifying PREIAD and PREIAT separately.
4. Specifying only PREIAT; in which case the code will take the current date as the default for PREIAD.

If PREIA is omitted and it is an external predecessor, then at least PREIAT is mandatory.

DELETE CPSIMP

Delete a condition dependency in the current plan.

```
ACTION=DELETE,RESOURCE=CPSIMP,ADID=application_description,
IA=input_arrival_datetime,
OPNO=operation_number,
CONDID=condition_ID,
COUNT=condition_counter,
DESC=descriptive_text,
PREADID=predecessor_application_description,
PREIA=predecessor_input_arrival_datetime,
PREOPNO=predecessor_operation_number,
PREPSTEP=step_name,
PRESTEP=procedure_invocation_step_name,
PRETYPE=check_type,
PRELOG=logical_operator,
PREVRC1=predecessor_return_code_value1,
PREVRC2=predecessor_return_code_value2,
PREVST=predecessor_status;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time: YYMMDDHHMM

operation_number

Integer (integer(4))

condition_ID

Integer (integer(3))

condition_counter

Integer (integer(3)). Use it to define the rule type:

0 All the condition dependencies in this condition must be true**n>0** At least *n* out of the condition dependencies in this condition must be true

The default is the current value.

descriptive_text

Character (char(16))

predecessor_application_description

Character (char(16))

predecessor_input_arrival_datetime

Date and time: YYMMDDHHMM

predecessor_operation_number

Integer (integer(4))

step_name

Character (char(8)). Use it to define a step level dependency. If the step is not in a procedure, this parameter identifies the job step name, otherwise it identifies the step name in the JCL procedure. It must correspond to the name of an EXEC PGM= statement.

procedure_invocation_step_name

Character (char(8)). Use it in conjunction with PREPSTEP when defining a step level dependency, only if the step is in a procedure, to identify the name of a step that invokes an in-stream or cataloged procedure. It must correspond to the name of an EXEC PROC= statement.

check_type

RC or ST (char(2))

logical_operator

Character (char(2)) can be:

GE Greater than or equal to. Valid only for RC check type.**GT** Greater than. Valid only for RC condition type.**LE** Less than or equal to. Valid only for RC condition type.**LT** Less than. Valid only for RC check type.**EQ** Equal to.**NE** Not equal to. Use it to specify conditions on final statuses only.**RG** Range.**predecessor_return_code_value1**

Character (char(4)). For values with less than four significant characters, use 0 as leading characters.

predecessor_return_code_value2

Character (char(4)) as second boundary in a range expressed by the RG logical operator. For values with less than four significant characters, use 0 as leading characters.

predecessor_status

Character (char(1)) valid only for ST check type

Specify the complete condition key that is the following operands: ADID, IA, OPNO, and CONDID.

DELETE CPSR

Suppress a special resource from a current plan operation.


```
ACTION=DELETE,RESOURCE=CPSR,ADID=application_description,
IA=input_arrival_datetime,
OPNO=operation_number,
RESNAME=resource_name;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time (YYMMDDHHMM)

operation number

Integer (integer(4))

resource_name

Character (char(44))

ADID and IA identify an occurrence, OPNO identifies an operation. RESNAME identifies operation special resource.

DELETE JS

Suppress a JCL from JCL repository. This action suppresses only one JCL at each time. Use JSCOM resource to perform a JCL mass deletion.

```
ACTION=DELETE,RESOURCE=JS,ADID=application_description,
IA=input_arrival_datetime,
OPNO=operation_number,
JOBNAME=jobname,
WSNAME=workstation_name;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time (YYMMDDHHMM) or 'X'

operation number

Integer (integer(4))

jobname

Character (char(8))

workstation_name

Character (char(4))

Note: ADID and IA arguments identify an occurrence. If the value of IA (IA=X) is specified as 'X', the OSTATUS argument completes the search for the occurrence. If more than one occurrence matches the specified arguments, the BCI tool returns a return code of 10 and the occurrence will not be modified. The OPNO argument identifies the operation with the occurrence.

DELETE JSCOM

This command allows to delete JCLs entries from the JS file. If STATUS is F, the command will delete all the entries that satisfy the other operands and that have an input arrival datetime value lower than the one specified by the IA operand. If STATUS is O, the command will delete those entries that satisfy the previous criteria and whose status in the JS file is C or S. If STATUS is C, the command will delete those entries that satisfy the previous criteria and whose status in the JS file is C. If STATUS is omitted, an additional subset will be selected: it is necessary that the corresponding occurrence in the current plan does not exist.

```
ACTION=DELETE,RESOURCE=JSCOM,ADID=application_description,
IA=input_arrival_datetime,
JOBNAME=jobname,
OPNO=operation_number,
STATUS=status,
WSNAME=workstation_name;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime
Date and time (YYMMDDHHMM)

jobname
Character (char(8))

operation_number
Integer (integer(4))

status F, O or C (char(1))

workstation_name
Character (char(4))

DELETE LTCPRE

Suppress a conditional predecessor of an occurrence from long-term plan.

```
ACTION=DELETE,RESOURCE=LTCPRE,ADID=application_description,
      IAD=input_arrival_date,
      IAT=input_arrival_time,
      PREADID=pre_application_description,
      PREIAD=pre_input_arrival_date,
      PREIAT=pre_input_arrival_time;
```

application_description
Character (char(16)) can be generic (%*)

input_arrival_date
Date (YYMMDD)

input_arrival_time
Time (HHMM)

pre_application_description
Character (char(16)) can be generic (%*)

pre_input_arrival_date
Date (YYMMDD)

pre_input_arrival_time
Time (HHMM)

PREADID, PREIAD, and PREIAT arguments identify conditional predecessor.

DELETE LTOC

Suppress a long-term plan occurrence.

```
ACTION=DELETE,RESOURCE=LTOC,ADID=application_description,
      GROUPDEF=group_definition,
      IAD=input_arrival_date,
      IAT=input_arrival_time;
```

application_description
Character (char(16)) can be generic (%*)

group_definition
Character (char(16)) can be generic (%*)

input_arrival_date
Date (YYMMDD)

input_arrival_time
Time (HHMM)

ADID, IAD, and IAT identify an occurrence from long-term plan.

DELETE LTPRE

Suppress a predecessor of an occurrence from long-term plan.

```
ACTION=DELETE,RESOURCE=LTPRE,ADID=application_description,
      IAD=input_arrival_date,
      IAT=input_arrival_time,
      PREADID=pre_application_description,
      PREIAD=pre_input_arrival_date,
      PREIAT=pre_input_arrival_time;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_date

Date (YYMMDD)

input_arrival_time

Time (HHMM)

pre_application_description

Character (char(16)) can be generic (%*)

pre_input_arrival_date

Date (YYMMDD)

pre_input_arrival_time

Time (HHMM)

PREADID, PREIAD, and PREIAT arguments identify predecessor.

DELETE OI

Suppress an operation instruction.

```
ACTION=DELETE,RESOURCE=OI,ADID=application_description,
      OPNO=operation_number,
      VALFROM=valid_from_datetime,
      VALTO=valid_to_datetime,
      MATCHTYP=type_of_match;
```

application_description

Character (char(16)) can be generic (%*)

operation_number

Integer (integer(4))

valid_from_datetime

Datetime (YYMMDDHHMM)

valid_to_datetime

Datetime (YYMMDDHHMM)

type_of_match

Character (char(4))

Note: MATCHTYP allows only three values to be specified: EXA , LOW, HIGH. If MATCHTYP is specified, then at least 1 operand between VALTO and VALFROM must be specified. If VALTO and/or VALFROM is specified, then MATCHTYP must be specified. VALTO and VALFROM cannot be specified if a permanent OI is selected among those to be deleted. If MATCHTYP=EXA, then only the version(s), with validity period(s) matching exactly the value(s) specified, will be deleted. If MATCHTYP=LOW, then all the version(s), with validity from and/or validity to lower or equal than the value(s) specified by VALFROM (and/or VALTO), will be deleted. If MATCHTYP=HIGH, then all the version(s), with validity from and/or validity to higher or equal than the value(s) specified by VALFROM (and/or VALTO), will be deleted.

DELETE RG

Suppress a run cycle group.

```
ACTION=DELETE,RESOURCE=RG,RGID=run_cycle_group_name,
      RGOWNER=owner,
      RGCALEND=calendar_name,
      RGVARTAB=variable_table_name,
      RUNNAME=run_cycle_name,
      RUNCALrun_cycle_calendar_name,
      RUNVTAB=run_cycle_variable_table_name,
      RUNSETID=run_cycle_subset_name;
```

run_cycle_group_name

Character (char(8)) can be generic (%*)

owner Character (char(16)) can be generic (%*)

calendar_name

Character (char(16)) can be generic (%*)

variable_table_name

Character (char(16)) can be generic (%*)

run_cycle_name

Character (char(8)) can be generic (%*)

run_cycle_calendar_name

Character (char(16)) can be generic (%*)

run_cycle_variable_table_name

Character (char(16)) can be generic (%*)

run_cycle_subset_name

Character (char(8)) can be generic (%*)

Return Codes (for DELETE requests)

0 Instruction is successfully processed.

4 Resource not found or user ID does not have RACF authorization.

This return code is not applicable to DELETE CPPRE; when the predecessor does not exist, RC=8 is set.

8 Instruction failed and an error message is written to the EQQMLOG file.

>8 Refer to "Return codes" on page 107.

EXPORT

The EXPORT instruction exports an AD, OI, RG, or all, from an IBM Workload Scheduler for z/OS to the same scheduler or to another subsystem. The AD is written to the EXPORTAD file. The OI is written to the EXPORTOI file. The RG is written to the EXPORTRG file. You can export AD, OI, RG, or all, with a selected segment or with generic arguments.

EXPORT AD

The syntax is the same as for LIST ADCOM, except for the operands TYPE and GROUPDEF, that are not supported.

Note: The output file must have been previously allocated with RECFM=VB (LRECL and BLKSIZE can have any value, because they are automatically set by the program).

EXPORT OI

The syntax is the same as for LIST OICOM.

Note: The output file must have been previously allocated with RECFM=VB (LRECL and BLKSIZE can have any value, because they are automatically set by the program).

EXPORT RG

The syntax is the same as for LIST RGCOR.

Note: The output file must have been previously allocated with RECFM=VB (LRECL and BLKSIZE can have any value, because they are automatically set by the program).

GROUPDEF support

The parameter GROUPDEF has been added to the following functions:

LIST Lists all applications in the AD, CP, or LTP that belong to the same GROUPDEF.

SELECT

Adds GROUPDEF as a selection criteria.

INSERT

If ADID=* is specified, GROUPDEF adds all occurrences that belong to the same group ID:

1. Current plan. If IA is not specified, the input arrival date is set to the current date, unless a different value is specified in the third parameter of the PARM field, and the input arrival time is set to the current time, unless a different value is specified in the fourth parameter of the parm field.
2. Long-term plan. IAD and IAT must be specified.

DELETE

The GROUPDEF support for the deletion applies only to the application description data base: there is no support of the GROUPDEF deletion in the CP.

MODIFY

If ADID=* is specified, all applications or occurrences belonging to the same group ID are modified:

1. Current Plan:
 - If IA is not specified, all occurrences are modified.
 - If IA is specified, only corresponding occurrences will be modified.
2. Long-term plan:
 - If IAD is specified, all occurrences corresponding to the date are modified.
 - If IAD and IAT are specified, only corresponding occurrences are modified.

IMPORT

The IMPORT instruction imports AD, OI, and RG resources exported using the EXPORT action. You can import selected resources from a file created by EXPORT with one or more selected segments or with specified arguments.

IMPORT AD

Import an application that has been exported. During export, some arguments could be changed.

Note: The file produced by export cannot be modified manually. It can contain more than one record. To select the correct record in the file used in input by the import function, ADID must be specified if more than one ad has to be imported with the same job. In these cases, a record is selected if ADID matches the value in one of the input records and for this record also the values of group and owner (if specified) match the ones in input. It is possible now to change the name of the application using a new operand of import NEWADID.

If NEWADID is omitted and:

- ADVERS=Y is active, it will be imported using the name specified in ADID provided that there are no already existing versions or, if there are, a new version can be created on the basis of the other operands.
- ADVERS=N is active, it will be imported using the name specified in ADID provided that there are no already existing versions.

If NEWADID is specified and:

- ADVERS=Y is active, it will be imported using the name specified in NEWADID provided that there are no already existing versions or, if there are, a new version can be created on the basis of the other operands.
- ADVERS=N is active, it will be imported using the name specified in NEWADID provided that there are no already existing versions.

If VALFROM and VALTO values are specified, they are set in the imported application. This can cause the failure in importing an application with multiversioning: The new value is forced to overwrite the original multiversioned validity causing the message EQQY724I to be issued.

```
ACTION=IMPORT,RESOURCE=AD,STATUS=ad_status,
      ADID=application_description,
      GROUP=authority_group_name,
      NEWADID=new_application_description,
      OWNER=owner,
      PRIORITY=priority,
      VALFROM=valid_from_date,
      VALTO=valid_to_date;
```

adstatus

P (pending) or A (ACTIVE) (char(1))

application_description

Character (char(16))

authority_group_name

Character (char(8))

new_application_description

Character (char(16))

owner Character (char(16))

priority

Integer: 1 through 9 (integer(4))

valid_from_date

Date (YYMMDD)

valid_to_date

Date (YYMMDD)

IMPORT OI

Imports an operator instruction that has been exported. During export, some arguments could be changed.

Note: The file produced by export cannot be modified manually. It can contain more than one record. In order to select the correct record in the file used in input by the import function, adid must be specified if more than one OI has to be imported with the same job. In these cases a record is selected if adid matches the value in one of the input records and for this record also the values of opno (if specified) matches the ones in input. It will be possible now to change the name of the application using a new operand of import newadid. If omitted the OI will be imported using the name specified in adid, provided that the ad is already present in the ad file.

```
ACTION=IMPORT,RESOURCE=OI,ADID=application_description,
      NEWADID=new_application_description,
      OPNO=operation_number;
```

application_description

Character (char(16))

new_application_description

Character (char(16))

operation_number

Integer (integer(4))

IMPORT RG

Import a run cycle group that has been exported. During export, some arguments could be changed.

Note: The file produced by export cannot be modified manually. It can contain more than one record. To select the correct record in the file used in input by the import function, RGID must be specified if more than one run cycle group has to be imported with the same job. In these cases, a record is selected if RGID matches the value in one of the input records and for this record also the values of group and owner (if specified) match the ones in input. It is possible now to change the name of the run cycle group using a new operand of import NEWRGID.

```
ACTION=IMPORT,RESOURCE=RG,RGID=run_cycle_group,
                                NEWRGID=new_run_cycle_group,
                                OWNER=owner;
```

run_cycle_group

Character (char(8))

new_run_cycle_group

Character (char(8))

owner Character (char(16))

INSERT

The INSERT instruction inserts resources into the corresponding tables.

AD An application description (Use BATCH LOADER or COPY or EXPORT/IMPORT) in ADs database.

OI Operator instructions (Use BATCH LOADER or COPY or EXPORT/IMPORT) in OI database.

RG A run cycle group (Use BATCH LOADER or COPY or EXPORT/IMPORT) in RG database.

CPCOND

Current plan condition

CPOC An occurrence in the current plan.

CPOCPRE

A predecessor to the current plan occurrence. The predecessor can be an occurrence or an external operation.

CPOP An operation in the current plan.

CPPRE

A predecessor to the current plan operation. Application of predecessor must be external.

CPSIMP

Current plan condition dependency

CPSR A special resource in the current plan operation.

LTOC An occurrence in the long-term plan.

LTPRE

A predecessor to the long-term plan occurrence.

Notes:

1. It is impossible to use the SDUR and EDUR arguments at the same time.
2. WSNAM and JOBNAM are mandatory operands. An ADID is identified by ADID and IA. If there is only one occurrence (regardless of its status) specify the correct value of IA or IA=X. If there is only one occurrence in a given status, specify the correct value of IA or IA=X and the OSTATUS to identify the occurrence. If there is more than one occurrence in the status specified by the OSTATUS operand, a return code of 10 is sent.
3. The character R in restartable_operation will be sent to PIF as blank.

4. For non-fault-tolerant workstations, the value of the centralized script field in the current plan is always set to ON regardless of the value entered in CSCSCRIPT.

INSERT CPCOND

Insert a condition in the current plan.

```
ACTION=INSERT,RESOURCE=CPCOND,ADID=application_description,
      IA=input_arrival_datetime,
      OPNO=operation_number,
      CONDID=condition_ID,
      COUNT=condition_counter,
      DESC=descriptive_text,
      PREADID=predecessor_application_description,
      PREIA=predecessor_input_arrival_datetime,
      PREOPNO=predecessor_operation_number,
      PREPSTEP=step_name,
      PRESTEP=procedure_invocation_step_name,
      PRETYPE=check_type,
      PRELOG=logical_operator,
      PREVRC1=predecessor_return_code_value1,
      PREVRC2=predecessor_return_code_value2,
      PREVST=predecessor_status;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time: YYMMDDHHMM

operation_number

Integer (integer(4))

condition_ID

Integer (integer(3))

condition_counter

Integer (integer(3)). Use it to define the rule type:

0 All the condition dependencies in this condition must be true

$n > 0$ At least n out of the condition dependencies in this condition must be true

The default is 0.

descriptive_text

Character (char(16))

predecessor_application_description

Character (char(16))

predecessor_input_arrival_datetime

Date and time: YYMMDDHHMM

predecessor_operation_number

Integer (integer(4))

step_name

Character (char(8)). Use it to define a step level dependency. If the step is not in a procedure, this parameter identifies the job step name, otherwise it identifies the step name in the JCL procedure. It must correspond to the name of an EXEC PGM= statement.

procedure_invocation_step_name

Character (char(8)). Use it in conjunction with PREPSTEP when defining a step level dependency, only if the step is in a procedure, to identify the name of a step that invokes an in-stream or cataloged procedure. It must correspond to the name of an EXEC PROC= statement.

check_type

RC or ST (char(2))

logical_operator

Character (char(2)) can be:

- GE** Greater than or equal to. Valid only for RC check type.
- GT** Greater than. Valid only for RC condition type.
- LE** Less than or equal to. Valid only for RC condition type.
- LT** Less than. Valid only for RC check type.
- EQ** Equal to.
- NE** Not equal to. Use it to specify conditions on final statuses only.
- RG** Range.

predecessor_return_code_value1

Character (char(4)). For values with less than four significant characters, use 0 as leading characters.

predecessor_return_code_value2

Character (char(4)) as second boundary in a range expressed by the RG logical operator. For values with less than four significant characters, use 0 as leading characters.

predecessor_status

Character (char(1)) valid only for ST check type

COUNT, DESC, and PREIA are optional argument.

The other argument are required. PREIA also is required for external predecessors.

To create an internal dependency, do not specify either PREADID or PREIA.

INSERT CPOC

Insert an occurrence to current plan.

```
ACTION=INSERT,RESOURCE=CPOC,ADID=application_description,
IA=input_arrival_datetime,
DEADLINE=deadline_date_datetime,
PRIORITY=priority,
ERRCODE=error_code,
DESC=descriptive_text,
GROUP=authority_group_name,
OWNER=owner,
ODESC=descriptive_text_of_owner,
GROUPDEF=application_groupid,
JCLVTAB= JCL_variable_table_id;
```

application_description

Character (char(16)) can be generic (%*)

application_groupid

Character (char 16))

authority_group_name

Character (char(8)) can be generic (%*)

deadline_date_datetime

Date and time (YYMMDDHHMM)

descriptive_text

Character (char(24))

descriptive_text_of_owner

Character (char(24))

error_code

Character (char(4))

input_arrival_datetime

Date and time (YYMMDDHHMM)

jcl_variable_table_id

Character (char(16)) can be generic (%*)

owner Character (char(16)) can be generic (%*)

priority

One integer (integer(4))

Note:

1. If LTP=Y was specified in the options card and the IA date specified or defaulted is higher than the CP end date, the occurrence will be inserted in the LTP. In this case the operands DESC, GROUP, OWNER, and ODESC, if specified, will be ignored.
2. The following facts must be carefully considered when GROUPDEF is specified:
 - Each of the applications belonging to the group is inserted separately, in alphabetical order (if CPDEPR=N is specified or defaulted in the options card) or in the order required to correctly resolve the dependencies (if CPDEPR=Y is specified in the options card). If any errors occur, BCIT ends and this results in only a partial insertion of the applications belonging to GROUPDEF.
 - If there are more than one version of a given application in the AD, you must check that the specified or defaulted IA falls within one of the corresponding validity periods; otherwise, the BCIT job will terminate with RC=8.
 - If CPDEPR was set to Y:
 - It is not allowed to specify for an operation an external dependency on itself or on another operation of the same application.
 - Cross dependencies are not supported. That is, if APPL1 contains an external dependency on an operation of APPL2 and APPL2 contains an external dependency on an operation of APPL1, this will be detected and the BCIT job will terminate with RC=8 (in this case, the INSERT will not be performed).
 - It is your responsibility to ensure that the CP does not contain occurrences of the applications belonging to GROUPDEF: if they exist, the dependencies will be resolved in a different way from expected.
 - It is not allowed to update the applications belonging to GROUPDEF at the same time as the batch job execution: the outcome of such concurrent update is unpredictable.

INSERT CPOCPRE

Insert a predecessor to a current plan occurrence.

```
ACTION=INSERT,RESOURCE=CPOCPRE,ADID=application_description,
      IAD=input_arrival_date,
      IA=input_arrival_datetime,
      IAT=input_arrival_time,
      OSTATUS=occurrence_status,
      PREADID=pre_application_description,
      PREIAD=pre_input_arrival_date,
      PREIA=pre_input_arrival_datetime,
      PREIAT=pre_input_arrival_time,
      PREOSTAT=pre_occurrence_status,
      PREOPNO=pre_operation_number;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_date

Date (YYMMDD)

input_arrival_datetime

Date and time (YYMMDDHHMM) or X

input_arrival_time

Time (HHMM)

occurrence_status

Occurrence status (char(1))

pre_application_description
 Character (char(16)) can be generic (%*)
pre_input_arrival_date
 Date (YYMMDD)
pre_input_arrival_datetime
 Date and time (YYMMDDHHMM) or X
pre_input_arrival_time
 Time (HHMM)
pre_occurrence_status
 Occurrence PRED status (char(1))
pre_operation_number
 integer (integer(4))

Note: An ADID is identified by ADID and IA (instead of IA it is possible to use IAD and IAT). If there is only one occurrence (regardless of its status) specify the correct value of IA or IA=X. If there is only one occurrence in a given status, specify the correct value of IA or IA=X and OSTATUS to identify the occurrence. If there is more than one occurrence in the status specified by the OSTATUS operand, a return code of 10 is sent. The same is true for the predecessor occurrence, which is identified by PREADID and PREIA (instead of PREIA it is possible to use PREIAD and PREIAT). If there is only one occurrence (regardless of its status) specify the correct value of PREIA or PREIA=X. If there is only one occurrence in a given status, specify the correct value of PREIA or PREIA=X and PREOSTAT to identify the occurrence. If there is more than one occurrence in the status specified by the PREOSTAT operand, a return code of 10 is sent.

INSERT CPOP

Insert an operation in current plan occurrence.

ACTION=INSERT,RESOURCE=CPOP,ADID=*application_description*,
 AEC=*automatic_error_completion*,
 AJR=*automatic_job_hold/release*,
 ASUB=*automatic_job_submission*,
 AUFUNC=*automated_function*,
 CLATE=*cancel_if_late*,
 CLNTYPE=*cleanup_type*,
 COIFBND=complete_if_bind_fails,
 COIFBNDZ=*complete_if_bind_fails*,
 COMMTEXT=*command_text*,
 COMPINFO=*completion_info*,
 CONDRJOB=*conditional_recovery_job*,
 CSCRIPT=*centralized_script*,
 DEADWTO=*issue_deadline_wto*,
 DESC=*descriptive_text*,
 DURATION=*duration*,
 EDUR=*estimated_duration*,
 EXPJCL=*expanded_jcl*,
 EXTNAME=*extended_name*
 EXTSE=*scheduling_environment_name*
 FORM=*form_number*,
 HRC=*highest_successful_retcode*,
 IA=*input_arrival_datetime*,
 JCLASS=*jobclass*,
 JOBCRT=*critical_job*,
 JOBNAME=*jobname*,
 JOBPOL=*WLM_job_policy*,
 MONITOR=*externally_monitored*,
 OPDL=*op_deadline_datetime*,
 OPIA=*op_input_arrival_datetime*,
 OPNO=*operation_number*,
 OSTATUS=*occurrence_status*,
 PSUSE=*paralleles_server_required*,

RIUSE=*resource1_required*,
 R2USE=*resource2_required*,
 READID=*remote_job_appl_desc*,
 REJOBNM=*remote_job_name*,
 REJSNM=*remote_job_stream_name*,
 REJSWS=*remote_ws_name*,
 REOPNO=*rem_job_op_num*,
 RERUT=*reroutable_operation*,
 RESTA=*restartable_operation*,
 SECELEM=*security_element*,
 SDUR=*estimated_duration_inseconds*,
 STATUS=*op_status*,
 TIMEDEP=*timedependent_job*,
 USRSYS=*user_sysout*,
 WLMSCLS=*WLM_service_class*,
 WSNAME=*wsname*;

application_description

Character (char(16)) can be generic (%*)

automated_function

Character (char(8))

automatic_error_completion

A character: Y or N (char(1))

automatic_job_hold/release

A character: Y or N (char(1))

automatic_job_submission

A character: Y or N (char(1))

cancel_if_late

A character: Y or N (char(1))

centralized_script

A character: Y or N (Char(1))

cleanup_type

A character: A, M, I or N (char(1))

command_text

Character (char(255))

complete_if_bind_fails

A character: Y or N (char(1)). Use C0IFBND for an IBM Workload Scheduler remote job, use C0IFBNDZ for an IBM Workload Scheduler for z/OS remote job.

completion_info

Character (char(64))

Note: This keyword is positional. For details, see *Managing the Workload*.

conditional_recovery_job

A character: Y or N (Char(1))

critical_job

A character: P, W, or N (char(1))

descriptive_text

Character (char(24))

duration

Time (Integer (4)) number of hundredths of a second

estimated_duration

Time (HHMM)

estimated_duration_inseconds

(Integer (4)) number of seconds

expanded_jcl

A character: Y or N (char(1))

extended_name

Character (char(54))

externally_monitored
A character: Y or N (char(1))

form_number
Character (char(8))

highest_successful_retcode
Integer (integer (4)): allowed values from 0 to 4095

input_arrival_datetime
Date and time (YYMMDDHHMM) or X

issue_deadline_Wto
A character: Y or N (char(1))

jobclass
A character (A-Z, 0-9) (char(1))

jobname
Character (char(8)) can be generic (%*)

occurrence_status
C, D, E, P, S, U, or W (Char(1))

op_deadline_datetime
Date and time (YYMMDDHHMM)

op_input_arrival_datetime
Date and time (YYMMDDHHMM)

op_status
A character (A, R, S, C, D, I, E, W, U, or *) (char(1))

operation_number
integer (integer(4))

paralleles_server_requi
Integer (integer(4))

remote_job_appl_desc
Character (char(16)). Only for an IBM Workload Scheduler for z/OS remote job.

remote_job_name
Character (char(40)). Only for an IBM Workload Scheduler remote job.

remote_job_op_num
Integer (integer(4)). Only for an IBM Workload Scheduler for z/OS remote job.

remote_job_stream_name
Character (char(16)). Only for an IBM Workload Scheduler remote job.

remote_ws_name
Character (char(16)). Only for an IBM Workload Scheduler remote job.

reroutable_operation
A character: Y or N (char(1))

resource1_required
Integer (integer(4))

resource2_required
Integer (integer(4))

restartable_operation
A character: Y, N or R (char(1))

scheduling_environment_name
Character (char(16))

security_element
Character (char(8))

timedependent_job
A character: Y or N (char(1))

user_sysout
A character: Y or N (char(1))

WLM_job_policy
L, D, S, or C (char(1))

WLM_service_class

Character (char(8)) can be generic (%*)

workstation_name

Character (char(4)) can be generic (%*)

INSERT CPPRE

Insert a predecessor to a current plan operation.

```
ACTION=INSERT,RESOURCE=CPPRE,ADID=application_description,
      IAD=input_arrival_date,
      IA=input_arrival_datetime,
      IAT=input_arrival_time,
      OSTATUS=occurrence_status,
      OPNO=operation_number,
      PREADID=pre_application_description,
      PREIAD=pre_input_arrival_date,
      PREIA=pre_input_arrival_datetime,
      PREIAT=pre_input_arrival_time,
      PREOSTAT=pre_occurrence_status,
      PREOPNO=pre_operation_number,
      TRPTIME=transport_time;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_date

Date (YYMMDD)

input_arrival_datetime

Date and time (YYMMDDHHMM) or X

input_arrival_time

Time (HHMM)

occurrence_status

Occurrence status (char(1))

operation number

Integer (integer(4))

pre_application_description

Character (char(16)) can be generic (%*)

pre_input_arrival_date

Date (YYMMDD)

pre_input_arrival_datetime

Date and time (YYMMDDHHMM) or X

pre_input_arrival_time

Time (HHMM)

pre_occurrence_status

Occurrence PRED status (char(1))

pre_operation_number

integer (integer(4))

transport_time

Integer (integer(4))

Note:

1. An ADID is identified by ADID and IA (instead of IA it is possible to use IAD and IAT). If there is only one occurrence (regardless of its status) specify the correct value of IA or IA=X. If there is only one occurrence in a given status, specify the correct value of IA or IA=X and OSTATUS to identify the occurrence. If there is more than one occurrence in the status specified by the OSTATUS operand, a return code of 10 is sent. The same is true for the predecessor occurrence, which is identified by PREADID and PREIA (instead of PREIA it is possible to use PREIAD and PREIAT). If there is only one occurrence (regardless of its status) specify the correct value of PREIA or PREIA=X. If there is only one occurrence in a given status, specify the correct

value of PREIA or PREIA=X and PREOSTAT to identify the occurrence. If there is more than one occurrence in the status specified by the PREOSTAT operand, a return code of 10 is sent.

2. To create an internal dependency, do not specify either PREADID or PREIA.

INSERT CPSIMP

Insert a condition dependency in the current plan.

```
ACTION=INSERT,RESOURCE=CPSIMP,ADID=application_description,
      IA=input_arrival_datetime,
      OPNO=operation_number,
      CONDID=condition_ID,
      COUNT=condition_counter,
      DESC=descriptive_text,
      PREADID=predecessor_application_description,
      PREIA=predecessor_input_arrival_datetime,
      PREOPNO=predecessor_operation_number,
      PREPSTEP=step_name,
      PRESTEP=procedure_invocation_step_name,
      PRETYPE=check_type,
      PRELOG=logical_operator,
      PREVRC1=predecessor_return_code_value1,
      PREVRC2=predecessor_return_code_value2,
      PREVST=predecessor_status;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time: YYMMDDHHMM

operation_number

Integer (integer(4))

condition_ID

Integer (integer(3))

condition_counter

Integer (integer(3)). Use it to define the rule type:

0 All the condition dependencies in this condition must be true

n>0 At least *n* out of the condition dependencies in this condition must be true

The default is the current value for this condition.

descriptive_text

Character (char(16))

predecessor_application_description

Character (char(16))

predecessor_input_arrival_datetime

Date and time: YYMMDDHHMM

predecessor_operation_number

Integer (integer(4))

step_name

Character (char(8)). Use it to define a step level dependency. If the step is not in a procedure, this parameter identifies the job step name, otherwise it identifies the step name in the JCL procedure. It must correspond to the name of an EXEC PGM= statement.

procedure_invocation_step_name

Character (char(8)). Use it in conjunction with PREPSTEP when defining a step level dependency, only if the step is in a procedure, to identify the name of a step that invokes an in-stream or cataloged procedure. It must correspond to the name of an EXEC PROC= statement.

check_type

RC or ST (char(2))

logical_operator

Character (char(2)) can be:

- GE** Greater than or equal to. Valid only for RC check type.
- GT** Greater than. Valid only for RC condition type.
- LE** Less than or equal to. Valid only for RC condition type.
- LT** Less than. Valid only for RC check type.
- EQ** Equal to.
- NE** Not equal to. Use it to specify conditions on final statuses only.
- RG** Range.

predecessor_return_code_value1

Character (char(4)). For values with less than four significant characters, use 0 as leading characters.

predecessor_return_code_value2

Character (char(4)) as second boundary in a range expressed by the RG logical operator. For values with less than four significant characters, use 0 as leading characters.

predecessor_status

Character (char(1)) valid only for ST check type

COUNT, DESC, and PREIA are optional argument.

The other argument are required. PREIA also is required for external predecessors.

To create an internal dependency, do not specify either PREADID or PREIA.

INSERT CPSR

Insert a special resource in the current plan.

ACTION=INSERT,RESOURCE=CPSR,ADID=MATENCIOP,IA=9201291500,RESNAME=name,RESUSAGE=X,OPNO=10.

**ACTION=INSERT,RESOURCE=CPSR,ADID=*application_description*,
IA=*input_arrival_datetime*,
ONCOMPL=*resource_on_complete*,
ONERROR=*resource_on_error*,
OPNO=*operation_number*,
RESNAME=*resource_name*,
RESUSAGE=*resource_usage*
QUANTITY=*resource_quantity*;**

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time (YYMMDDHHMM)

resource_on_complete

Character (char(1)) On complete action: Y, N, or R

resource_on_error

Character (char(1)) On error action: Y or N

operation_number

Integer (integer(4))

resource_name

Character (char(44))

resource_usage

Character: S or X (char(1))

resource_quantity

Integer (integer(4))

INSERT LTOC

Insert an occurrence in the long-term plan.


```
ACTION=INSERT,RESOURCE=LTOC,ADID=application_description,
      DEADLINE=deadline_datetime,
      ERRCODE=error_code,
      GROUPEF=group_definition,
      IAD=input_arrival_date,
      IAT=input_arrival_time,
      PRIORITY=priority;
```

application_description

Character (char(16)) can be generic (%*)

deadline_datetime

Date and time (YYMMDDHHMM)

error_code

Character (char(4))

group_definition

Character (char(16)) can be generic (%*)

input_arrival_date

Date (YYMMDD)

input_arrival_time

Time (HHMM)

priority

One integer (integer(4))

INSERT LTPRE

Insert a predecessor in a long-term plan operation.

```
ACTION=INSERT,RESOURCE=LTPRE,ADID=application_description,
      IAD=input_arrival_date,
      IAT=input_arrival_time,
      PREADID=pre_application_description,
      PREIAD=pre_input_arrival_date,
      PREIAT=pre_input_arrival_time;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_date

Date (YYMMDD)

input_arrival_time

Time (HHMM)

pre_application_description

Character (char(16)) can be generic (%*)

pre_input_arrival_date

Date (YYMMDD)

pre_input_arrival_time

Time (HHMM)

Return codes

- 0 Instruction successfully processed.
- 4 Resource not found or user ID have no RACF authorization.
- 8 Instruction failed and an error message is written to EQQMLOG.
- >8 Refer to "Return codes" on page 107.

LIST

With the LIST instruction, you can list all the resources of the scheduler. The LIST request retrieves all the segments (COMMON SEGMENT and ADDITIONAL SEGMENTS) of a resource. If you want only the common segment, you must use SELECT instruction for this resource. You can see that the resources names are suffixed by COM.

This action also permits you to give AD, OI, and RG in BATCH LOADER (options BL=Y and BLPRT=Y) or only extract their keys (options BL=Y and BLPRT=N) for using to build SELECT action with specific resources.

The resources used by the LIST action are as follows:

ADCOM

Application descriptions

ADKEY

Application description keys

CLCOM

Calendars

CPCONDCO

Current plan conditions

CPOC, CPOCCOM

Current plan occurrences

CPOPCOM

Current plan operations

CPWSCOM

Current plan workstations

CPWSVCOM

Current plan virtual workstations

JCLVCOM

JCL variable tables

JSCOM

JCL segments

LTOCCOM

Long-term plan occurrences

OICOM

Operator instructions

PRCOM

Periods

RGCOM

Run cycle groups

RGKEY

Run cycle group keys

WSCOM

Workstations

WSVCOM

Virtual workstations

LIST ADCOM

List of application descriptions.

```
ACTION=LIST,RESOURCE=ADCOM,STATUS=adstatus,
                                ADID=application_description,
                                GROUP=authority_group_name,
                                GROUPDEF=groupdef,
                                OWNER=owner,
                                PRIORITY=priority,
                                TYPE=type,
                                VALFROM=valid_from_date,
                                VALTO=valid_to_date;
```

adstatus

A or P (char(1))

application_description

Character (char(16)) can be generic (%*)

authority_group_name

Character (char(8)) can be generic (%*)

groupdef Character (char(16))
owner Character (char(16)) can be generic (%*)
priority One integer (integer(4))
type A or G (char(1))
valid_from_date Date: YYMMDD
valid_to_date Date: YYMMDD

An application key is the argument concatenation ADID, STATUS, and VALTO
 Further arguments only complete the list.

Note: The result stays in referenced files as:

SYSPRINT(LRECL=132)

If OPTIONS BL=N (detailed format)

AD (LRECL=23)

If OPTIONS BL=Y and BLPRT=N (ADs keys)

BATCHL (LRECL=80)

If OPTIONS BL=Y & BLPRT=Y(BATCH LOADER FORMAT)

See "GROUPDEF support" on page 120 for details on GROUPDEF support.

LIST ADKEY

List ADs Keys.

ACTION=LIST,RESOURCE=ADKEY,ADID=*application_description*,
 STATUS=*adstatus*,
 VALTO=*valid_to_date*;

application_description

Character (char(16)) can be generic (%*)

adstatus

A or P (char(1))

valid_to_date

Date: YYMMDD

LIST CLCOM

List calendars.

ACTION=LIST,RESOURCE=CLCOM,CALENDAR=*calendar*;

calendar

Character (char(16)) can be generic (%*)

LIST CPCONDCO

List current plan conditions.

ACTION=LIST,RESOURCE=CPCONDCO,ADID=*application_description*,
 IA=*input_arrival_datetime*,
 OPNO=*operation_number*,
 CONDID=*condition_ID*,
 CONDVAL=*condition_status*;

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time: YYMMDDHHMM

operation_number

Integer (integer(4))

condition_ID

Integer (integer(3))

condition_status

U, T, or F (char(1))

The workstation key contains ADID, IA, OPNO, and CONDID. Specify at least one of these operands.

LIST CPOC, CPOCCOM

List current plan occurrences.

In the RESOURCE keyword, you can specify CPOC or CPOCCOM because they are equivalent. CPOCCOM was introduced together with the support of application dependencies, to identify the part of the occurrence in the plan without the variable section of dependencies.

```
ACTION=LIST,RESOURCE=CPOC,ADID=application_description,
      GROUP=authority_group_name,
      GROUPDEF=group_definition,
      IA=input_arrival_datetime,
      MCPADDED=mcp_added,
      MONITOR=externally_monitored,
      STATUS=oc_status,
      OWNER=owner,
      PRIORITY=priority,
      RERUN=rerun;
```

application_description

Character (char(16)) can be generic (%*)

authority_group_name

Character (char(8)) can be generic (%*)

externally_monitored

Y or N (char(1))

group_definition

Character(char(16))

input_arrival_datetime

Date and time: YYMMDDHHMM

mcp_added

Y or N (char(1))

oc_status

C, D, E, P, S, U, or W (char(1))

owner Character (char(16)) can be generic (%*)

priority

One integer (integer(4))

rerun Y or N (char(1))

IA and ADID must be supplied.

Note: Results are written in referenced files as:

SYSPRINT(LRECL=132)

If OPTIONS BL=N (detailed format)

CPOC (LRECL=80)

If OPTIONS BL=Y (ADID,IA,STATUS)

LIST CPOPCOM

List of operations current plan occurrences.

```
ACTION=LIST,RESOURCE=CPOPCOM,ADID=application_description,
      GROUP=authority_group_name,
      CONDRJOB=conditional_recovery_job,
      DPREM=removable_by_DP,
      ERRCODE=error_code,
      EXTNAME=extended_name
```

EXTSE=*scheduling_environment_name*
 IA=*input_arrival_datetime*,
 JOBNAME=*jobname*,
 MATCHTYP=*match_type*,
 MONITOR=*externally_monitored*,
 OPNO=*operation_number*,
 OWNER=*owner*,
 PRIORITY=*priority*,
 SHADOWJ=*shadow_job*,
 STATUS=*op_status*,
 UNEXPRC=*unexpected_rc*,
 WAITFORW=*started_on_WAIT_workstation*
 WAITSE=*waiting_for_SE*,
 WMPRED=*waiting_for_Mandatory_Pending_Predecessors*,
 WPPRED=*waiting_for_Pending_OR_Mandatory_Pending_Predecessors*,
 WPPRED=*waiting_for_Pending_Predecessors*,
 WSNAME=*workstation_name*;

application_description

Character (char(16)) can be generic (%*)

authority_group_name

Character (char(8)) can be generic (%*)

conditional_recovery_job

Y or N (char(1))

removable_by_DP

Y or N (char(1))

error_code

Character (char(4))

extended_name

Character (char(54))

scheduling_environment_name

Character (char(16))

externally_monitored

Y or N (char(1))

input_arrival_datetime

Date and time: YYYYMMDDHHMM

jobname

Character (char(8)) can be generic (%*)

match_type

Character (char(3))

op_status

*, A, R, S, C, D, I, E, W, or U (char(1))

operation_number

Integer (integer(4))

owner Character (char(16)) can be generic (%*)

priority

One integer (integer(4))

shadow_job

Y or N (char(1))

started_on_WAIT_workstation

Y or N (char(1))

unexpected_rc

Y or N (char(1))

waiting_for_Mandatory_Pending_Predecessors

Y or N (char(1))

waiting_for_Pending_OR_Mandatory_Pending_Predecessors

Y or N (char(1))

waiting_for_Pending_Predecessors

Y or N (char(1))

waiting_for_SE

Y or N (char(1))

workstation_name

Character (char(4)) can be generic (%*)

Note: There is only one value accepted for MATCHTYP: EXA. If MATCHTYP=EXA is specified, STATUS=* will be interpreted as a normal character instead of as a generic matching character.

ADID and IA identify an occurrence with an operation identified by the OPNO parameter. Further arguments complete the list.

Note: Results are referenced in files as:

SYSPRINT(LRECL=132)

If OPTIONS BL=N (detailed format)

CPOP (LRECL=80)

If OPTIONS BL=Y (ADID,IA,OPNO,STATUS, JOBNAME)

LIST CPWSCOM

List current plan workstations.

```
ACTION=LIST,RESOURCE=CPWSCOM,WSAUTO=automation_workstation,
                               WSNAME=workstation_name,
                               WSREP=workstation_rep_attri,
                               WSTYPE=workstation_type,
                               WSTWS=fault_tolerant_workstation,
                               WSVIRT=virtual_workstation,
                               WSWAIT=wait_workstation,
                               WSZCENTR=z-centric_workstation;
```

automation_workstation

Y or N (char(1))

fault_tolerant_workstation

Y or N (char(1))

workstation_name

Character (char(4)) can be generic (%*)

workstation_rep_attri

A, S, C, or N (char(1))

workstation_type

C, G, R, or P (char(1))

virtual_workstation

Y or N (char(1))

wait_workstation

Y or N (char(1))

z-centric_workstation

Y or N (char(1))

The workstation key contains WSNAME, WSTYPE, WSREP and WSTWS.

At least one of the operands starting with WS must be specified.

LIST CPWSVCOM

List current plan virtual workstations.

```
ACTION=LIST,RESOURCE=CPWSVCOM,WSNAME=workstation_name,
                               WSDEST=workstation_destination;
```

workstation_name

Character (char(4)) can be generic (%*)

workstation_destination

Character (char(8)) can be generic (%*). To indicate a local destination, specify *****

The workstation key contains WSNAME and WSDEST.

At least one of the two operands must be specified.

LIST JCLVCOM

List of JCL variable tables.

ACTION=LIST,RESOURCE=JCLVCOM,JCLVTAB=*jcl_variable_table_id*

jcl_variable_table_id

Character (char(16)) can be generic (%*)

The output is written in the file referenced by the DATAFI DD card.

LIST JSCOM

List JCLs from current plan operations.

ACTION=LIST,RESOURCE=JSCOM,ADID=*application_description*,
 IA=*input_arrival_datetime*,
 JOBNAME=*jobname*,
 OPNO=*operation_number*,
 WSNAME=*workstation_name*;

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time: YYMMDDHHMM

jobname

Character (char(8)) can be generic (%*)

operation_number

Integer (integer(4))

workstation_name

Character (char(4)) can be generic (%*)

LIST LTOCCOM

List long-term plan occurrences.

ACTION=LIST,RESOURCE=LTOCCOM,ADID=*application_description*,
 GROUP=*authority_group_name*,
 IAD=*input_arrival_date*,
 IAT=*input_arrival_time*,
 OWNER=*owner*;

application_description

Character (char(16)) can be generic (%*)

authority_group_name

Character (char(8)) can be generic (%*)

input_arrival_date

Date: YYMMDD

input_arrival_time

Time: HHMM

owner Character (char(16)) can be generic (%*)

ADID, IAD, and IAT identify an occurrence in the long-term plan.

LIST OICOM

List operator instructions.

ACTION=LIST,RESOURCE=OICOM,ADID=*application_description*,
 OPNO=*operation_number*;

application_description

Character (char(16)) can be generic (%*)

operation_number

Integer (integer(4))

Note: Result created in referenced files with:

SYSPRINT(LRECL=132)

If OPTIONS BL=N (detailed result)

OI (LRECL=23)

If OPTIONS BL=Y and BLPRT=N (OI Keys)

BATCHL (LRECL=80)

If OPTIONS BL=Y and BLPRT=Y (BATCH LOADER format)

LIST PRCOM

List periods.

```
ACTION=LIST,RESOURCE=PRCOM,PERIOD=period_name,
                                PRTYPE=period_type;
```

period_name

Character (char(8)) can be generic (%*)

period_type

A, W, or N (char(1))

LIST RGCOM

List of run cycle groups.

```
ACTION=LIST,RESOURCE=RGCOM,RGID=run_cycle_group_name,
                                RGOWNER=owner,
                                RGCALEND=calendar_name,
                                RGVARTAB=variable_table_name,
                                RUNNAME=run_cycle_name,
                                RUNCALrun_cycle_calendar_name,
                                RUNVTAB=run_cycle_variable_table_name,
                                RUNSETID=run_cycle_subset_name;
```

run_cycle_group_name

Character (char(8)) can be generic (%*)

owner Character (char(16)) can be generic (%*)

calendar_name

Character (char(16)) can be generic (%*)

variable_table_name

Character (char(16)) can be generic (%*)

run_cycle_name

Character (char(8)) can be generic (%*)

run_cycle_calendar_name

Character (char(16)) can be generic (%*)

run_cycle_variable_table_name

Character (char(16)) can be generic (%*)

run_cycle_subset_name

Character (char(8)) can be generic (%*)

LIST RGKEY

List of run cycle group keys.

```
ACTION=LIST,RESOURCE=RGKEY,RGID=run_cycle_group_name,
                                RGOWNER=owner,
                                RGCALEND=calendar_name,
                                RGVARTAB=variable_table_name,
                                RUNNAME=run_cycle_name,
                                RUNCALrun_cycle_calendar_name,
                                RUNVTAB=run_cycle_variable_table_name,
                                RUNSETID=run_cycle_subset_name;
```


run_cycle_group_name
Character (char(8)) can be generic (%*)

owner Character (char(16)) can be generic (%*)

calendar_name
Character (char(16)) can be generic (%*)

variable_table_name
Character (char(16)) can be generic (%*)

run_cycle_name
Character (char(8)) can be generic (%*)

run_cycle_calendar_name
Character (char(16)) can be generic (%*)

run_cycle_variable_table_name
Character (char(16)) can be generic (%*)

run_cycle_subset_name
Character (char(8)) can be generic (%*)

LIST WSCOM

List workstations in WS database.

```
ACTION=LIST,RESOURCE=WSCOM,WSAUTO=automation_workstation,
WSNAME=workstation_name,
WSTYPE=workstation_type,
WSREP=workstation_rep_attri,
WSTWS=fault_tolerant_workstation,
WSVIRT=virtual_workstation,
WSWAIT=wait_workstation,
WSZCENTR=z-centric_workstation;
WSRETYPE=remotengine_workstation;
```

automation_workstation
Y or N (char(1))

fault_tolerant_workstation
Y or N (char(1))

remote-engine_workstation
Z or D (char(1))

virtual_workstation
Y or N (char(1))

wait_workstation
Y or N (char(1))

workstation_name
Character (char(4)) can be generic (%*)

workstation_rep_attri
A, S, C, or N (char(1))

workstation_type
C, G, R, or P (char(1))

z-centric_workstation
Y or N (char(1))

Note:

1. At least one of the operands starting with WS must be specified.
2. If you specify WSNAME, this setting supersedes all the other arguments.

LIST WSVCOM

List virtual workstations in WS database.

```
ACTION=LIST,RESOURCE=WSVCOM,WSNAME=workstation_name,
WSDEST=workstation_destination;
```

workstation_name
Character (char(4)) can be generic (%*)

workstation_destination

Character (char(8)) can be generic (%*). To indicate a local destination, specify *****

At least one of the two operands must be specified.

Return codes

- 0 The instruction is successfully processed.
- 4 The resource was not found or the user ID does not have the required RACF authorization.
- 8 The instruction failed, and an error message has been written to the EQQMLOG file.
- >8 Refer to "Return codes" on page 107.

LISTSTAT

The LISTSTAT instruction returns a code that reflects the status of the resource. You can list the status for CPOC or CPOPCOM resources.

LISTSTAT CPOC

Return a code according to the current status of an occurrence

```
ACTION=LISTSTAT,RESOURCE=CPOC,ADID=application_description,
                                GROUP=authority_group_name,
                                IA=input_arrival_datetime,
                                MCPADDED=mcp_added,
                                OWNER=owner,
                                PRIORITY=priority,
                                RERUN=rerun;
```

application_description

Character (char(16))

authority_group_name

Character (char(8))

input_arrival_datetime

Date and time (YYMMDDHHMM) or X

mcp_added

Y or N (char(1))

owner

Character (char(16))

priority

One integer (integer(4))

rerun

Y or N (char(1))

valid_from_date

Date (YYMMDD)

valid_to_date

Date (YYMMDD)

Notes:

1. The occurrence must be uniquely identified.
2. The correct value of IA or IA=X must be specified when there is only one occurrence of the given adid to avoid that the value specified or defaulted through the parm invocation parameter be passed to PIF.

LISTSTAT CPOPCOM

Return a code according to the current status of an operation.

```
ACTION=LISTSTAT,RESOURCE=CPOPCOM,ADID=application_description,
                                GROUP=authority_group_name,
                                ERRCODE=error_code,
                                IA=input_arrival_datetime,
                                JOBNAME=jobname,
```

OPNO=*operation_number*,
OWNER=*owner*,
PRIORITY=*priority*,
WSNAME=*wsname*;

application_description
Character (char(16))
authority_group_name
Character (char(8))
error_code
Character (char(4))
input_arrival_datetime
Date and time (YYMMDDHHMM) or X
jobname
Character (char(8))
operation_number
Integer (integer(4))
owner Character (char(16))
priority
One integer (integer(4))
workstation_name
Character (char(4))

Note: IA is the input arrival date and time of the occurrence, not of the operation. There are no mandatory operands, but those supplied must be enough to uniquely identify the operation. The correct value of IA or IA=X must be specified when there is only one occurrence of the given addid to avoid that the value specified or defaulted through the parm invocation parameter be passed to PIF.

Return codes

0 Instruction successfully processed
4 Resource not found or user ID has no RACF authorization
8 Instruction failed and a error message is written to EQQMLOG file
31 Occurrence status C (completed)
32 Occurrence status D (deleted)
33 Occurrence status E (ended in error)
34 Occurrence status P (processor pending)
35 Occurrence status S (started)
36 Occurrence status U (undecided)
37 Occurrence status W (no started operations)
40 Operation status *
41 Operation status A (waiting for input to arrive)
42 Operation status R (ready)
43 Operation status S (started)
44 Operation status C (completed)
45 Operation status D (deleted)
46 Operation status I (interrupted)
47 Operation status E (ended in error)

- 48 Operation status W (waiting for a predecessor)
- 49 Operation status U (undecided)
- >49 Refer to “Return codes” on page 107.

MODIFY

The MODIFY instruction modifies some resources in the corresponding database. The current version of the Batch Command Interface modifies the following resources:

CPCOND

Current plan condition.

CPEXT

An extended name for the current plan operation.

CPOC An occurrence in the current plan.

CPOP An operation in the current plan.

CPREND

Information about an IBM Workload Scheduler remote job.

CPRENTZ

Information about an IBM Workload Scheduler for z/OS remote job.

CPSAI

System automation information for the current plan operation.

LTOC An occurrence in the long-term plan.

MODIFY CPCOND

Modify a condition in the current plan.

```
ACTION=MODIFY,RESOURCE=CPCOND,ADID=application_description,
      IA=input_arrival_datetime,
      OPNO=operation_number,
      CONDID=condition_ID,
      COUNT=condition_counter,
      DESC=descriptive_text;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time: YYMMDDHHMM

operation_number

Integer (integer(4))

condition_ID

Integer (integer(3))

condition_counter

Integer (integer(3)). Use it to define the rule type:

0 All the condition dependencies in this condition must be true

n>0 At least *n* out of the condition dependencies in this condition must be true

The default is the current value.

descriptive_text

Character (char(16))

Specify the complete condition key that is the following operands: ADID, IA, OPNO, and CONDID.

MODIFY CPEXT

Create, modify, or delete the extended name of an operation in the current plan.

```
ACTION=MODIFY,RESOURCE=CPEXT,ADID=application_description,
EXTNAME=extended_name,
EXTSE=scheduling_environment_name,
IA=input_arrival_datetime,
OPNO=operation_number;
```

application_description

Character (char(16)) can be generic (%*)

extended_name

Character (char(54)). To delete the operation extended name, enter
EXTNAME= ' '

scheduling_environment_name

Character (char(16)).

input_arrival_datetime

Date and time (YYMMDDHHMM)

operation_number

Integer (integer(4))

Note: The action will fail if you are inserting an operation that is to be added to the Symphony file and the value of TWSJOBNAME is either EXTNAME or EXTNOCC. Under these conditions, to insert a new operation that uses an extended name, you should use the INSERT CPOP action and include the EXTNAME parameter in the request.

MODIFY CPOC

Modify an occurrence in the current plan. If the attempt fails, BCIT will perform at a predefined time interval other attempts (for a maximum of 3).

```
ACTION=MODIFY,RESOURCE=CPOC,ADID=application_description,
ALLMON=all_operations_externally_monitored,
IA=input_arrival_datetime,
OSTATUS=oc_status,
IANEW=input_arrival_datetime,
DEADLINE=deadline_datetime,
PRIORITY=priority,
ERRCODE=error_code,
STATUS=new_status,
GROUPDEF=group_definition,
JCLVTAB=jcl_variable_table_id;
```

all_operation_externally_monitored

A character: Y or N (char(1))

application_description

Character (char(16)) can be generic (%*)

deadline_datetime

Date and time (YYMMDDHHMM)

error_code

Character (char(4))

group_definition

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time (YYMMDDHHMM) or 'X'

jcl_variable_table_id

Character (char(16)) can be generic (%*)

new_status

New status for the occurrence (char(1))

oc_status

Occurrence status: C, D, E, P, S, U, or W (char(1))

priority

One integer (integer(4))

Note: ADID and IA arguments identify an occurrence. If the value of IA is specified as 'X' (IA=X), the OSTATUS argument completes the search for the occurrence. If IA=X is specified and more than one occurrence matches the specified arguments, the BCI tool returns a return code of 10 and the occurrence will not be modified.

MODIFY CPOP

Modify an operation in the current plan. If the attempts fails, BCIT will perform other attempts at a predefined time interval (for the maximum of 3 total).

```
ACTION=MODIFY,RESOURCE=CPOP,ADID=application_description,
AEC=automatic_error_completion,
AJR=automatic_job_hold/release,
ASUB=automatic_job_submission,
CLATE=cancel_if_late,
CLNTYPE=cleanup_type,
CONDRJOB=conditional_recovery_job,
COIFBND=complete_if_bind_fails,
COIFBNDZ=complete_if_bind_fails,
DEADWTO=issue_deadline_wto,
DESC=descriptive_text,
DURATION=duration,
EDUR=estimated_duration,
ERRCODE=error_code,
EXPJCL=expanded_jcl,
FORM=form_number,
GROUP=authority_group_name,
HRC=highest_successful_retcode,
IA=input_arrival_datetime,
JCLASS=jobclass,
JOBCRT=critical_job,
JOBNAME=jobname,
JOBPOL=WLM_job_policy,
MONITOR=externally_monitored,
OPCMD=operation_command,
OPDL=op_deadline_datetime,
OPIA=op_input_arrival_datetime,
OPNO=operation_number,
OSTATUS=oc_status,
PSUSE=paralleles_server_required,
R1USE=resource1_required,
R2USE=resource2_required,
READID=remote_job_appl_desc,
REJOBNM=remote_job_name,
REJSNM=remote_job_stream_name,
REJSWS=remote_ws_name,
REOPNO=remote_job_op_num,
RERUT=reroutable_operation,
RESTA=restartable_operation,
SDUR=estimated_duration_inseconds,
STATUS=op_status,
TIMEDEP=timedependent_job,
USRSYS=user_sysout,
WLMSCLS=WLM_service_class,
WSNAME=workstation_name;
```

application_description

Character (char(16)) can be generic (%*)

automatic_error_completion

A character: Y or N (char(1))

automatic_job_hold/release

A character: Y or N (char(1))

automatic_job_submission

A character: Y or N (char(1))

cancel_if_late
A character: Y or N (char(1))

cleanup_type
A character: A, M, I or N (char(1))

complete_if_bind_fails
A character: Y or N (char(1)). Use COIFBNDD for an IBM Workload Schedulerremote job, use COIFBNDZ for anIBM Workload Scheduler for z/OS remote job.

conditional_recovery_job
A character: Y or N (Char(1))

critical_job
A character: P, W, or N (char(1))

descriptive_text
Character (char(24))

duration
Time (Integer (4)) number of hundredths of a second

error_code
Character (char(4))

estimated_duration
Time (HHMM)

estimated_duration_inseconds
(Integer (4)) number of seconds

expanded_jcl
A character: Y or N (char(1))

externally_monitored
A character: Y or N (char(1))

form_number
Character (char(8))

highest_successful_retcode
Integer (integer (4)): allowed values from 0 to 4095.

input_arrival_datetime
Date and time (YYMMDDHHMM) or 'X'

issue_deadline_wto
A character: Y or N (char(1))

jobclass
A character: A–Z, 0–9 (char(1))

jobname
Character (char(8)) can be generic (%*)

oc_status
Occurrence status: C, D, E, P, S, U, or W (char(1))

op_deadline_datetime
Date and time (YYMMDDHHMM)

op_input_arrival_datetime
Date and time (YYMMDDHHMM)

op_status
A character: A, R, S, C, D, I, E, W, U, or * (char(1))

operation_command
Character (char(2)) can be:

- EX = Execute operation
- KJ = Kill operation¹
- KR = Kill recovery job²
- MH = Hold operation
- MR = Release operation
- NP = NOP operation
- PN = Prompt reply no
- PY = Prompt reply yes

- UN = Un-NOP operation

Note:

1. Applies only to operations running on IBM Workload Scheduler for z/OS Agents or on distributed workstations that are directly connected to the end-to-end server (OPCMaster).
2. Applies only to recovery jobs running on distributed workstations that are directly connected to the end-to-end server (OPCMaster).

operation_number

Integer (integer(4))

paralleles_server_required

Integer (integer(4))

remote_job_appl_desc

Character (char(16)). Only for a IBM Workload Scheduler for z/OS remote job.

remote_job_name

Character (char(40)). Only for an IBM Workload Scheduler remote job.

remote_job_op_num

Integer (integer(4)). Only for an IBM Workload Scheduler for z/OS remote job.

remote_job_stream_name

Character (char(16)). Only for an IBM Workload Scheduler remote job.

remote_ws_name

Character (char(16)). Only for an IBM Workload Scheduler remote job.

reroutable_operation

A character: Y or N (char(1))

resource1_required

Integer (integer(4))

resource2_required

Integer (integer(4))

restartable_operation

A character: Y, N or R (Char(1))

timedependent_job

A character: Y or N (char(1))

user_sysout

A character: Y or N (Char(1))

WLM_job_policy

L, D, S, or C (char(1))

WLM_service_class

Character (char(8)) can be generic (%*)

workstation_name

Character (char(4)) can be generic (%*)

Note:

1. The parameters of the MODIFY CPOP request are the same as the parameters of the INSERT CPOP request. In addition, the MODIFY CPOP request has the ERRCODE and the OPCMD parameters.
2. An ADID is identified by ADID and IA. If there is only one occurrence (regardless of its status) specify the correct value of IA or IA=X. If there is only one occurrence in a given status, specify the correct value of IA or IA=X and OSTATUS to identify the occurrence. If there is more than one occurrence in the status specified by the OSTATUS operand, a return code of 10 is sent and the occurrence will not be modified. The OPNO argument identifies the operation within the occurrence.
3. It is impossible to use SDUR and EDUR arguments at the same time.

4. The character R in WLM_job_policy and in restartable_operation will be sent to PIF as blank.

MODIFY CPREND

Modify the information about a job scheduled to run on a remote IBM Workload Scheduler engine.

```
ACTION=MODIFY,RESOURCE=CPREND,REJSNM=rem_job_stream_name,
                                REJOBNM=rem_job_name,
                                REJSWS=rem_ws_name,
                                COIFBNDF=comp_if_bind_fails;
```

comp_if_bind_fails

A character: Y or N (Char(1))

rem_job_name

Character (char(40))

rem_job_stream_name

Character (char(16))

rem_ws_name

Character (char(16))

MODIFY CPRENTZ

Modify the information about a job scheduled to run on an IBM Workload Scheduler for z/OS remote engine.

```
ACTION=MODIFY,RESOURCE=CPRENTZ,READID=rem_job_appl_desc,
                                REOPNO=rem_job_op_num,
                                COIFBNDF=comp_if_bind_fails;
```

comp_if_bind_fails

A character: Y or N (Char(1))

rem_job_appl_desc

Character (char(16))

rem_job_op_num

Integer (integer(4))

MODIFY CPSAI

Create, modify, or delete the system automation information for an operation in the current plan.

```
ACTION=MODIFY,RESOURCE=CPSAI,ADID=application_description,
                                AUTFUNC=automated_function,
                                COMMTXT=command_text,
                                COMPINFO=completion_info;
                                IA=input_arrival_datetime,
                                OPNO=operation_number,
                                SECELEM=security_element;
```

application_description

Character (char(16)) can be generic (%*).

automated_function

Character (char(8)). To delete the operation automated function, enter AUTFUNC=' '.

command_text

Character (char(255)).

completion_info

Character (char(64)). To delete the operation completion information, enter COMPINFO=' '.

Note: This keyword is positional. For details, see *Managing the Workload*.

input_arrival_datetime

Date and time (YYMMDDHHMM).

operation_number

Integer (integer(4)).

security_element

Character (char(8)). To delete the operation security element, enter SECELEM=' '.

MODIFY LTOC

Modify an occurrence in the long-term plan.

```
ACTION=MODIFY,RESOURCE=LTOC,ADID=application_description,
      DEADLINE=deadline_datetime,
      ERRCODE=error_code,
      GROUPDEF=group_definition
      IAD=input_arrival_date,
      IAT=input_arrival_time,
      PRIORITY=priority;
```

application_description

Character (char(16)) can be generic (%*)

deadline_datetime

Date and time (YYMMDDHHMM)

error_code

Character (char(4))

group_definition

Character(char(16))

input_arrival_date

Date (YYMMDD)

input_arrival_time

Time (HHMM)

priority

One integer (integer(4))

ADID, IAD, and IAT identify an occurrence to the long-term plan.

Return codes (for MODIFY requests)

0 Instruction is successfully processed.
 4 Resource not found or user ID does not have RACF authorization.
 8 Instruction failed and an error message displays on EQQMLOG file.
 10 More than one occurrence matches the specified arguments.
 >10 Refer to "Return codes" on page 107.

OPTIONS

The OPTIONS instruction gives you a choice of process options; you can specify options to be used when performing PIF requests. You can use these options to automatically resolve dependencies when adding current plan or long-term plan occurrences with INSERT.

```
ACTION=OPTIONS,ADOICHK=adoichk,
      ADVERS=advers,
      BL=batchl,
      BLPRT=blprt,
      CPDEPR=cpdepr,
      ERROR=error,
      LTDEPR=ltdepr,
      LTP=ltp;
```

adoichk

Y or N(char(1)). Specify whether or not you want AD/OI consistency checks to be made every time an application is deleted or modified. Consistency checks involve looking in the application description data base for matches for all the operator instructions in the application. Any operator instructions without match are deleted. Default: N.

advers Y or N (char(1)).

When Y is specified, in all the cases in which the running of the requested command impacts the AD database for applications having more than one validity period, BCIT ensures that the resulting different versions of the application have consecutive validity intervals, preserving the same logic used by the ISPF dialogs.

Note: Mixed use of ADVERS=Y and ADVERS=N is not allowed. Before running any BCIT command with ADVERS=Y, you must update the versions affected using the ISPF dialogs ensuring the basic versioning rule is followed. If any ADID has only one version in a given status, its value must be 711231; if there are more versions in a given status, the more recent one must a value of 711231. Default: N.

batchl Y or N (char(1)). Ask the batchl format.

- If BL=Y and BLPRT=Y, the commands LIST ADCOM or OICOM produce the following output:
 - AD (OI) statements (in the format that can be used in input to the batch loader utility) are written in the file referenced by the BATCHL DD card. Note that the PREJOBN keyword will never be produced because the BL utility accepts it as an alternative to PREOPNO, which is always supplied for all the predecessors.
 - AD (OI) keys are written in the file referenced by the AD (OI) DD card.
- If BL=Y and BLPRT=N, only AD and OI keys are written inside their corresponding file (referenced by DD AD or DD OI calls)
- If BL=Y, the LIST CPOC or CPOPCOM result is written in referenced file (CPOC or CPOP). Default: N

blprt Y or N (char(1)) write AD or OI keys in AD and OI files if BL=Y and BLPRT=N. Default: Y if BL=Y.

cpdepr

Y or N (char(1)) resolve automatically external dependences (extra application) during an insert of a new occurrence in a current plan. Default: N

error Y or N (char(1)) when Y is specified, processing of SYSIN statements stops in case of failure. Default: N

ltdepr Y or N (char(1)) resolve automatically external dependencies (extra application) during an insert of a new occurrence in an long-term plan. Default: N

ltp Y or N (char(1)). When Y is specified, if INSERT or DELETE actions are requested for resource CPOC and the input arrival time specified or defaulted is later than the current plan end time, the requested action will be performed on the long-term plan. Default: N

REPLACE

Use the BATCH LOADER.

SELECT

The SELECT instruction selects a resource. If the resource is not suffixed with COM, all segments will be selected and printed; otherwise only the common segment will be processed.

Whereas LIST retrieves all resources, SELECT retrieves only the selected resource. You have to identify the key of the selected resource, you cannot use generic keys. The list below, gives the resources retrieved by SELECT action.

The SELECT action also gives to you AD, OI, and RG in batch loader format (options BL=Y and BLPRT=Y). From keys created by a list action (options BL=Y and BLPRT=N), you can define SELECT instructions for AD, OI, and RG and send them like SYSIN to the Batch Command Interface tool.

The resources authorized by the SELECT action are as follows:

AD, ADCOM

Application description

CL, CLCOM

Calendars

CPCOND, CPCONDCO

Current plan condition

CPOC, CPOCCOM

Current plan occurrences

CPOP, CPOPCOM

Current plan operations

CPST Current plan status

CPWS, CPWSCOM

Current plan workstations

JCLPREP

Promptable variables setup to current plan

JCLPREPA

Automatic variables setup to current plan

JCLV, JCLVCOM

JCL variable table

JS, JSCOM

JCL segment

LTOC, LTOCCOM

Long-term plan (LTP) occurrences

OI, OICOM

Operator instructions

PR, PRCOM

Periods (period table)

RG, RGCOM

Run cycle groups

WS, WSCOM

Workstations (workstation table)

SELECT AD

Select an application description.

```
ACTION=SELECT,RESOURCE=AD, STATUS=adstatus,
                                ADID=application_description,
                                GROUP=authority_group_name,
                                OWNER=owner,
                                PRIORITY=priority,
                                VALFROM=valid_from_date,
                                VALTO=valid_to_date;
```

adstatus

Status: A or P (char(1))

application_description

Character (char(16))

authority_group_name

Character (char(8))

owner Character (char(16))
priority
 One integer (integer(4))
valid_from_date
 Date (YYMMDD)
valid_to_date
 Date (YYMMDD)

Note: The result is created in the referenced file with:

SYSPRINT(LRECL=132)

 If OPTIONS BL=N (detailed format)

BATCHL (LRECL=80)

 If OPTIONS BL=Y and BLPRT=Y(BATCH LOADER format)

SELECT CL

Retrieve a calendar.

ACTION=SELECT,RESOURCE=CL,CALENDAR=*calendar*;

calendar

 Character (char(16))

SELECT CPCOND

Retrieve current plan conditions.

ACTION=SELECT,RESOURCE=CPCOND,ADID=*application_description*,
 IA=*input_arrival_datetime*,
 OPNO=*operation_number*,
 CONDID=*condition_ID*,
 CONDVAL=*condition_status*;

application_description

 Character (char(16)) can be generic (%*)

input_arrival_datetime

 Date and time: YYMMDDHHMM

operation_number

 Integer (integer(4))

condition_ID

 Integer (integer(3))

condion_status

 U, T, or F (char(1))

Specify the complete condition key that is the following operands: ADID, IA, OPNO, and CONDID.

SELECT CPOC, CPOCCOM

Retrieve a current plan occurrence.

In the RESOURCE keyword, specify CPOC or CPOCCOM. CPOCCOM was introduced together with the support of application dependencies, to identify the part of the occurrence in the plan without the variable section of dependencies. If you specify CPOC, the variable section of dependency is returned, when present. If you specify CPOCCOM, the variable section of dependency is not returned.

ACTION=SELECT,RESOURCE=CPOC,ADID=*application_description*,
 GROUP=*authority_group_name*,
 GROUPDEF=*group_definition*,
 IA=*input_arrival_datetime*,
 MCPADDED=*mcp_added*,
 MONITOR=*externally_monitored*,

STATUS=*oc_status*,
OWNER=*owner*,
PRIORITY=*priority*,
RERUN=*rerun*;

application_description

Character (char(16))

authority_group_name

Character (char(8))

externally_monitored

Y or N (char(1))

group_definition

Character(char(16))

input_arrival_datetime

Date and time: YYMMDDHHMM

mcp_added

Y or N (char(1))

oc_status

The IBM Workload Scheduler for z/OS status code: C, D, E, P, S, U, or W
(char(1))

owner Character (char(16))

priority

One integer (integer(4))

rerun Y or N (char(1))

Note: The result is created in the referenced file by:

SYSPRINT(LRECL=132)

If OPTIONS BL=N (detailed format)

CPOC (LRECL=80)

If OPTIONS BL=Y (ADID,IA,STATUS)

SELECT CPOP

Retrieve an operation of current plan occurrence.

ACTION=SELECT,RESOURCE=CPOP,ADID=*application_description*,
CONDRJOB=*conditional_recovery_job*,
CSCRIPT=*centralized_script*,
DPREM=*removable_by_DP*,
ERRCODE=*error_code*,
EXTNAME=*extended_name*,
EXTSE=*scheduling_environment_name*,
GROUP=*authority_group_name*
IA=*input_arrival_datetime*,
JOBNAME=*jobname*,
MATCHTYP=*match_type*,
MONITOR=*externally_monitored*,
OPNO=*operation_number*,
OWNER=*owner*,
PRIORITY=*priority*,
SHADOWJ=*shadow_job*,
STATUS=*oc_status*,
UNEXPRC=*unexpected_rc*
WAITFORW=*started_on_WAIT_workstation*,
WAITSE=*waiting_for_SE*,
WMPRED=*waiting_for_Mandatory_Pending_Predecessors*,
WPMPRED=*waiting_for_Pending_OR_Mandatory_Pending_Predecessors*,
WPPRED=*waiting_for_Pending_Predecessors*,
WSNAME=*workstation_name*;

application_description

Character (char(16))

authority_group_name
 Character (char(8))

conditional_recovery_job
 Y or N (char(1))

centralized_script
 Y or N(char(1))

removable_by_DP
 Y or N (char(1))

error_code
 Character (char(4))

extended_name
 Character (char(54))

scheduling_environment_name
 Character (char(16))

externally_monitored
 Y or N(char(1))

input_arrival_datetime
 Date and time (YYMMDDHHMM)

jobname
 Character (char(8))

match_type
 Character (char(3))

op_status
 *, A, R, S, C, D, I, E, W, or U (char(1))

operation_number
 Integer (integer(4))

owner Character (char(16))

priority
 One integer (integer(4))

shadow_job
 Y or N (char(1))

started_on_WAIT_workstation
 Y or N (char(1))

unexpected_rc
 Y or N (char(1))

waiting_for_Mandatory_Pending_Predecessors
 Y or N (char(1))

waiting_for_Pending_OR_Mandatory_Pending_Predecessors
 Y or N (char(1))

waiting_for_Pending_Predecessors
 Y or N (char(1))

waiting_for_SE
 Y or N (char(1))

workstation_name
 Character (char(4))

Note: There is only one value accepted for MATCHTYP: EXA. If MATCHTYP=EXA is specified, STATUS=* will be interpreted as a normal character instead of as a generic matching character.

Note: The result is created in the referenced file by:

SYSPRINT(LRECL=132)

If OPTIONS BL=N (detailed format)

CPOP (LRECL=80)

If OPTIONS BL=Y (ADID,IA,OPNO,STATUS, JOBNAME)

SELECT CPST

Retrieve the current plan status.

ACTION=SELECT,RESOURCE=CPST

SELECT CPWS

List a current plan workstation.

ACTION=SELECT,RESOURCE=CPWS,WSAUTO=*automation_workstation*,
WSNAME=*workstation_name*,
WSTYPE=*workstation_type*,
WSREP=*workstation_rep_attri*,
WSTWS=*fault_tolerant_workstation*,
WSVIRT=*virtual_workstation*,
WSWAIT=*wait_workstation*,
WSZCENTR=*z-centric_workstation*
WSRETYPE=*remotengine_workstation*;

automation_workstation

Y or N (char(1))

fault_tolerant_workstation

Y or N (char(1))

remote-engine_workstation

Z or D (char(1))

virtual_workstation

Y or N (char(1))

wait_workstation

Y or N (char(1))

workstation_name

Character (char(4))

workstation_rep_attri

A, S, C, or N (char(1))

workstation_type

C, G, R, or P (char(1))

z-centric_workstation

Y or N (char(1))

At least one of the operands starting with WS must be specified.

SELECT CPWSV

Retrieve a current plan virtual workstation.

ACTION=SELECT,RESOURCE=CPWSV,WSNAME=*workstation_name*,
WSDEST=*workstation_destination*;

workstation_name

Character (char(4))

workstation_destination

Character (char(8)). To indicate a local destination, specify *****

Specify the complete workstation key that is both the operands.

SELECT JCLPREP

Retrieve a JCLPREP of an operation.

ACTION=SELECT,RESOURCE=JCLPREP,ADID=*application_description*,
IA=*input_arrival_datetime*,
OPNO=*operation_number*;

application_description

Character (char(16))

input_arrival_datetime

Date and time (YYMMDDHHMM)

operation_number
Integer (integer(4))

SELECT JCLPREPA

Retrieve a JCLPREPA of an operation.

ACTION=SELECT,RESOURCE=JCLPREPA,ADID=*application_description*,
IA=*input_arrival_datetime*,
OPNO=*operation_number*;

application_description
Character (char(16))

input_arrival_datetime
Date and time (YYMMDDHHMM)

operation_number
Integer (integer(4))

SELECT JCLV

Retrieve a variable table.

ACTION=SELECT,RESOURCE=JCLV,JCLVTAB=*jcl_variable_table_id*;

jcl_variable_table_id
Character (char(16)) can be generic (%*)

SELECT JS

List JCL of an operation.

ACTION=SELECT,RESOURCE=JS,ADID=*application_description*,
IA=*input_arrival_datetime*,
JOBNAME=*jobname*,
OPNO=*operation number*,
WSNAME=*workstation_name*;

application_description
Character (char(16))

input_arrival_datetime
Date and time (YYMMDDHHMM)

jobname
Character (char(8))

operation_number
Integer (integer(4))

workstation_name
Character (char(4)) can be generic (%*)

SELECT LTOC

Retrieve a long-term plan occurrence.

ACTION=SELECT,RESOURCE=LTOC,ADID=*application_description*,
GROUP=*authority_group_name*,
IAD=*input_arrival_date*,
IAT=*input_arrival_time*,
OWNER=*owner*;

application_description
Character (char(16))

authority_group_name
Character (char(8))

input_arrival_date
Date (YYMMDD)

input_arrival_time
Time (HHMM)

owner Character (char(16))

SELECT OI

Retrieve operator instructions of an operation.

```
ACTION=SELECT,RESOURCE=OI,ADID=application_description,
          OPNO=operation_number,
          VALTO=valid_to_date;
```

application_description

Character (char(16))

operation_number

Integer (integer(4))

valid_to_date

date (YYMMDDHHMM)

Note:

1. VALTO is an optional operand, which must be specified if more than one temporary OI satisfies the other operands and must not be specified if the request is issued for a permanent OI.
2. The result is created in the referenced files by:

SYSPRINT(LRECL=132)

If OPTIONS BL=N (detailed result)

BATCHL(LRECL=80)

If OPTIONS BL=Y and BLPRT=Y (BATCH LOADER format)

SELECT PR

Retrieve a period.

```
ACTION=SELECT,RESOURCE=PR,PERIOD=period_name,
          PRTYPE=period_type;
```

period_name

Character (char(8))

period_type

A,W,N (char(1))

SELECT RG

Select a run cycle group.

```
ACTION=SELECT,RESOURCE=RG,RGID=run_cycle_group_name,
          RGOWNER=owner,
          RGCALEND=calendar_name,
          RGVARTAB=variable_table_name,
          RUNNAME=run_cycle_name,
          RUNCAL=run_cycle_calendar_name,
          RUNVTAB=run_cycle_variable_table_name,
          RUNSETID=run_cycle_subset_name;
```

run_cycle_group_name

Character (char(8)) can be generic (%*)

owner Character (char(16)) can be generic (%*)

calendar_name

Character (char(16)) can be generic (%*)

variable_table_name

Character (char(16)) can be generic (%*)

run_cycle_name

Character (char(8)) can be generic (%*)

run_cycle_calendar_name

Character (char(16)) can be generic (%*)

run_cycle_variable_table_name

Character (char(16)) can be generic (%*)

run_cycle_subset_name

Character (char(8)) can be generic (%*)

SELECT WS

List a workstation.

```
ACTION=SELECT,RESOURCE=WS,WSAUTO=automation_workstation,
        WSNAME=workstation_name,
        WSTYPE=workstation_type,
        WSREP=workstation_rep_att,
        WSTWS=fault_tolerant_workstation,
        WSVIRT=virtual_workstation,
        WSWAIT=wait_workstation,
        WSZCENTR=z-centric_workstation;
        WSRETYPE=remotengine_workstation;
```

automation_workstation

Y or N (char(1))

fault_tolerant_workstation

Y or N (char(1))

remote-engine_workstation

Z or D (char(1))

virtual_workstation

Y or N (char(1))

wait_workstation

Y or N (char(1))

workstation_name

Character (char(4))

workstation_rep_attri

A, S, C, or N (char(1))

workstation_type

C, G, R, or P (char(1))

z-centric_workstation

Y or N (char(1))

Note:

1. At least one of the operands starting with WS must be specified.
2. If you specify WSNAME, this setting supersedes all the other arguments.

SELECT WSV

Retrieve a virtual workstation.

```
ACTION=SELECT,RESOURCE=WSV,WSNAME=workstation_name,
        WSDEST=workstation_destination;
```

workstation_name

Character (char(4))

workstation_destination

Character (char(8))

Specify the complete workstation key that is both the operands.

Return codes

- | | |
|----|--|
| 0 | Instruction successfully processed |
| 4 | Resource not found or user ID has no RACF authorization |
| 8 | Instruction failed and an error message is written to the EQQMLOG file |
| >8 | Refer to "Return codes" on page 107. |

SETSTAT

The SETSTAT request changes the condition status from undecided to true or false, if the original status is undecided because of missing step-end information. It applies to the following resource:

CPSIMP

Current plan condition dependency

SETSTAT CPSIMP

Insert a condition dependency in the current plan.

```
ACTION=SETSTAT,RESOURCE=CPSIMP,ADID=application_description,
      IA=input_arrival_datetime,
      OPNO=operation_number,
      CONDID=condition_ID,
      COUNT=condition_counter,
      DESC=descriptive_text,
      NEWSTAT=new_status,
      PREADID=predecessor_application_description,
      PREIA=predecessor_input_arrival_datetime,
      PREOPNO=predecessor_operation_number,
      PREPSTEP=step_name,
      PRESTEP=procedure_invocation_step_name,
      PRETYPE=check_type,
      PRELOG=logical_operator,
      PREVRC1=predecessor_return_code_value1,
      PREVRC2=predecessor_return_code_value2,
      PREVST=predecessor_status;
```

application_description

Character (char(16)) can be generic (%*)

input_arrival_datetime

Date and time: YYMMDDHHMM

operation_number

Integer (integer(4))

condition_ID

Integer (integer(3))

condition_counter

Integer (integer(3)). Use it to define the rule type:

0 All the condition dependencies in this condition must be true

***n*>0** At least *n* out of the condition dependencies in this condition must be true

The default is the current value for this condition.

descriptive_text

Character (char(16))

new_status

Character (char(1)) can be:

T True

F False

predecessor_application_description

Character (char(16))

predecessor_input_arrival_datetime

Date and time: YYMMDDHHMM

predecessor_operation_number

Integer (integer(4))

step_name

Character (char(8)). Use it to define a step level dependency. If the step is not in a procedure, this parameter identifies the job step name, otherwise it identifies the step name in the JCL procedure. It must correspond to the name of an EXEC PGM= statement.

procedure_invocation_step_name

Character (char(8)). Use it in conjunction with PREPSTEP when defining a step level dependency, only if the step is in a procedure, to identify the name of a step that invokes an in-stream or cataloged procedure. It must correspond to the name of an EXEC PROC= statement.

check_type

RC or ST (char(2))

logical_operator

Character (char(2)) can be:

GE Greater than or equal to. Valid only for RC check type.

GT Greater than. Valid only for RC condition type.

LE Less than or equal to. Valid only for RC condition type.

LT Less than. Valid only for RC check type.

EQ Equal to.

NE Not equal to. Use it to specify conditions on final statuses only.

RG Range.

predecessor_return_code_value1

Character (char(4)). For values with less than four significant characters, use 0 as leading characters.

predecessor_return_code_value2

Character (char(4)) as second boundary in a range expressed by the RG logical operator. For values with less than four significant characters, use 0 as leading characters.

predecessor_status

Character (char(1)) valid only for ST check type

COUNT, DESC, and PREIA are optional arguments.

The other argument are required. PREIA also is required for external predecessors.

Return codes

0 Instruction successfully processed.

4 Resource not found or user ID have no RACF authorization.

8 Instruction failed and an error message is written to EQQMLOG.

>8 Refer to "Return codes" on page 107.

Chapter 4. Control Language (OCL)

IBM Workload Scheduler for z/OS Control Language (OCL) is a programming language that increases the usability of the program interface (PIF) and offers a comprehensive set of instructions to control the objects managed by the scheduler.

Using OCL, you can adjust and experiment with:

- JCL variables
- Applications and related dependencies
- Application occurrences status
- Special resources
- Applications structure
- User-supplied dates

See “IBM Workload Scheduler for z/OS control language” on page 4 for more information about OCL as a REXX program.

What you can do using OCL

Using OCL, you can do the following:

- Through its IF-THEN-ELSE instruction, better control your workload and take action appropriate to the result.
- Dynamically modify the behavior of your batch applications, without changing the databases of the scheduler.
- Complete, delete, add, release, modify, and control occurrences or operations in the scheduler plans.
- Kill jobs in STARTED state on IBM Workload Scheduler for z/OS Agents or on distributed workstations that are directly connected to the end-to-end server (OPCMaster). Also kill recovery jobs in EXECUTING state on distributed workstations that are directly connected to the end-to-end server (OPCMaster).
- Dynamically change the dependencies among the operations in the current plan, by adding or deleting predecessor occurrences, and releasing the internal and external successors.
- Release all or some of the external successors of an occurrence.
- Integrate the JSUACT, OPSTAT, SRSTAT, and WSSTAT TSO commands provided by the scheduler that can be issued in accordance with the result of a relational test.
- Update the value of the variables you can use in your jobs.
- Verify or compare a part of a variable value, such as the third character of the value.
- Use a variable value as a condition for scheduling an application.
- Validate a date you supply, then derive from it about 210 variable values, which you can use in your job.

Advantages of OCL

Some of the main advantages of using OCL are as follows:

- It further automates the scheduling process.
- It increases the usability of the PIF.
- It offers more flexibility for managing the workload.
- It facilitates migration from non-IBM scheduling products to IBM Workload Scheduler for z/OS.
- It increases the number of recovery actions offered by the Automatic Job Recovery (AJR) function of the scheduler: the OCL instructions could be executed as part of the recovery actions.

Summary of OCL instructions

The OCL instructions are specified in a data set referenced by the SYSIN DD card. For an example of an OCL job, the EQQYRJCL member of the EQQMISC data set, see “Sample job and procedure” on page 265.

Here is a brief description of the instructions:

ADD Inserts applications or an application group into the scheduler current plan or into the long-term plan.

ADDCOND

Adds a condition definition to an operation in the scheduler current plan.

ADDOP

Inserts an operation into an occurrence in the current plan.

ADDPRED

Adds a predecessor dependency definition to an occurrence in the scheduler current plan.

ADDRES

Adds a special resource to an operation in the scheduler current plan.

ADDSIMP

Adds a single dependency definition to a condition in the scheduler current plan.

ADOPSAI

Specifies system automation information for the operation (command text, automated function, security element, and completion information).

CALL Invokes a REXX procedure.

CHGEXTNAME

Defines or modifies the extended job information associated with the operations of an occurrence in the scheduler current plan. You can also use CHGEXTNAME to delete the extended job information associated with the operations of an occurrence in the scheduler current plan.

CHGJOB

Changes the jobname associated with the operations of an occurrence in the scheduler current plan.

CHGOPSAI

Defines or modifies the system automation information associated with the operations of an occurrence in the scheduler current plan. You can use CHGOPSAI also to delete the system automation information.

CHKAPPL

Verifies the presence of occurrences in the scheduler current plan; when keywords STATUS(E) and ALERT(YES) are set, it checks operations that ended in error and sends to TSO users a message containing the related error information.

CHKDATE

Checks the validity of an input date in the following formats: YYYY/MM/DD, YY/MM/DD, YYYYMMDD, YYMMDD, DD/MM/YYYY, DD/MM/YY, DDMMYYYY, DDMMYY, and assigns to variables values it derives from the input date.

COMPL

Completes occurrences or specific operations of the occurrences in the scheduler current plan.

DEL Deletes an occurrence or an operation from the scheduler current plan or from the long-term plan.

DELCOND

Deletes a condition from an operation in the scheduler current plan.

DELPRED

Deletes a dependency definition from an occurrence in the scheduler current plan with a predecessor operation.

DELRES

Deletes a special resource from an operation in the scheduler current plan.

DELSIMP

Deletes a single dependency definition from a condition in the scheduler current plan.

EXIT Sets the program exit return code.

FORCE

Forces the execution of the application.

GOTO

Moves to a point specified by a LABEL statement within the OCL program.

HOLD

Sets the operations of an occurrence to HOLD status.

IF-THEN-ELSE

Performs a relational test and, depending on the result, executes one of the following instructions: SET, UPD, SETUPD, GOTO, CALL, EXIT, CHKAPPL, ADD, ADDOP, DEL, COMPL, ADDPRED, DELPRED, ADDRES, DELRES, CHGJOB, JSUACT, OPSTAT, SRSTAT, WSSTAT, CHKDATE, RELSUCC, RELOP, FORCE, RELEASE, HOLD, NOP, UNNOP, MODOP, or IF-THEN-ELSE.

JSUACT

Activates or inactivates the job submission function in the z/OS environment, in the distributed environment, or in both environments.

INIT Specifies the ID of the scheduler variable table in which the job variables are described and specifies the scheduler subsystem ID.

KILLJOB

Stops a job that is already running. Applies only to operations running on

IBM Workload Scheduler for z/OS Agents or on distributed workstations that are directly connected to the end-to-end server (OPCMaster).

KILLREC

Stops a recovery job that is already running. Applies only to operations running on distributed workstations that are directly connected to the end-to-end server (OPCMaster).

This action can be taken only on recovery jobs that are in the EXECUTING status, so that their operation number is known. The application number is required (the operation number is optional, but if you do not specify it, all the operations in the application are killed) to identify the operation that is to be killed.

LABEL

Defines labels within the OCL program SYSIN.

MODCOND

Modifies the condition defined for an operation in the scheduler current plan.

MODOP

Modifies the operations details.

NOP Removes an operation from the current plan. When a NOP operation is ready to be started, it is immediately set to C status.

OPSTAT

Changes the status of an operation in the scheduler current plan.

PROMPTN

Specifies that NO is the reply to a recovery prompt issued for an abended operation.

PROMPTY

Specifies that YES is the reply to a recovery prompt issued for an abended operation.

RELEASE

Releases the operations of an occurrence that are in HOLD status.

RELOP

Releases the internal successors of an operation.

RELSUCC

Releases the successors of the occurrence by deleting the external dependency definition.

SET Sets a user variable.

SETUPD

Sets a user variable and updates its default value in an scheduler variable table. This instruction is equivalent to the SET and UPD instructions together.

SRSTAT

Modifies the availability status of a special resource.

UNNOP

Restores an operation from NOP status.

UPD Updates the default value of a user variable in an scheduler variable table.

WSSTAT

Modifies the status of a workstation.

WTO Displays messages on the system console and waits for a reply.

For a full description of all OCL instructions, see “Description of OCL instructions” on page 175.

If the OCL PARMLIB TSOCMD parameter is set to YES, you can also specify a TSO command or a REXX expression other than those listed here. For example:

```
DO X = 1 TO 3;,
  SAY 'HELLO';,
END
Select;,
  when var1 = 'OK' then ADD APPL(TEST01);,
  when var1 = 'KO' then EXIT 16;,
  otherwise nop;,
End
```

For punctuation rules, see “Specifying OCL instructions” on page 171.

Customizing OCL

Follow these steps to customize the OCL function.

1. Customize the initialization parameters in the OCL library member EQQYRPRM. The initialization parameters are as follows:

SUBSYS()

Default subsystem ID for controller. This is overridden by the SUBSYS specified in the EQQYPARM DD card and by the SUBSYS() keyword specified in the OCL instructions (see “Specifying input arrival dates and times” on page 172).

OPCTRK()

Default subsystem ID for tracker.

LUNAME()

LU node name of the server communicating with the controller system.

REMHOST()

The server host name for the program interface TCP/IP session. REMHOST and LUNAME are mutually exclusive.

REMPORT()

The server port number for the program interface TCP/IP session. REMPORT and LUNAME are mutually exclusive.

DEFCONDID()

Default condition identifier; for example, 001.

DEFIAT()

Default Input Arrival Time in HHMM format; for example, 1800.

DEFOPNO()

Default operation number; for example, 001.

DEFPREOPNO()

Default external predecessor operation number; for example, 099. This is used only for external dependencies.

DUBPROC()

Determines the parameter with which the BPX1SDD routine is to be invoked. It can be Y or N.

Y The BPX1SDD routine is invoked by using the DUBPROCESS parameter.

N The BPX1SDD routine is invoked by using the DUBPROCESSDEFER parameter.

SORT()

If you do not specify the occurrence input arrival date and time in the OCL instructions, SORT() determines which application occurrence in the current plan to consider:

MAX Determines the occurrence with the latest input arrival date and time

MIN Determines the occurrence with the earliest input arrival date and time (please refer to the General rules section)

TSOCMD()

The external TSO command and REXX instruction activator. It can be YES or NO.

YES You can specify TSO commands and REXX instructions other than those provided by OCL.

NO You can specify only OCL instructions.

INCVALUE()

Indicates the increasing value for the variable calculation. OCL will add days to or subtract days from an input date up to the INCVALUE value. It will calculate the new date $\pm n$ calendar days, where n is a value from 1 to the value of INCVALUE, which can be in the range 1-30.

This parameter is used only by CHKDATE instruction.

LOOPDEP()

OCL does not add a group of applications if the group contains applications that are mutually dependent. In this case, the operation fails and message EQQCL7QE is issued. Set LOOPDEP to NO if you do not want OCL to check whether the applications in a group are mutually dependent.

The default is YES.

For information about how to specify initialization parameters, see "Specifying the initialization parameters" on page 169.

2. If you want to enable the logging function, allocate the log data set. The log data set must be a sequential data set with LRECL 133 and RECFM FB. For information about the logging function, see "Logging executed instructions" on page 171.
3. Customize all the JCL cards in the OCL library member EQQYRPRC, in accordance with your installation standards.
4. Customize all the JCL cards in the OCL library member EQQYRJCL, in accordance with your installation standards.
5. Specify the OCL instructions in the data set referred to by the SYSIN DD card in the EQQYRJCL member.

Example:

```
//EQQOCL.SYSIN DD *  
CHKAPPL APPL(TEST*) STATUS(E)  
* ...
```

6. Submit the job.

Specifying the initialization parameters

You can use the DEFOPNO and DEFPREOPNO parameters if your application's first and last operations are nonreporting operations. This is illustrated in Figure 6. OCL will use these parameters as default operation numbers.

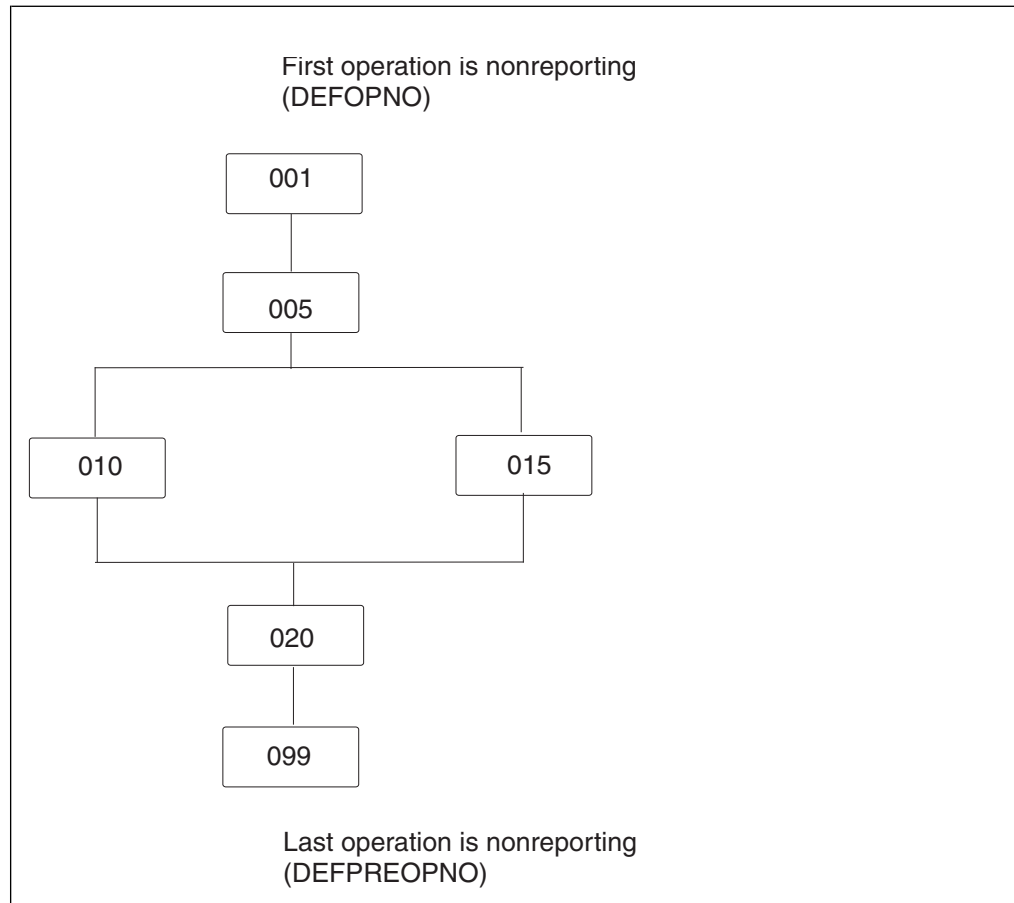


Figure 6. Application flow example

Example 1

If DEFOPNO is 001 and DEFPREOPNO is 099, the instruction:

```
ADDPRED APPL(TEST02) PA(TEST01)
```

adds a predecessor dependency between operation 099 of application TEST01 and operation 001 of application TEST02.

Example 2

Assuming that the application shown in Figure 6 is TEST03, the instruction:

```
RELOP APPL(TEST03) OP(5)
```

deletes the dependency between operation 005 and its internal successors (operations 010 and 015) and defines operation 001 (DEFOPNO) as an internal predecessor of these two operations.

Example 3

The DEFIAT parameter is used when you specify the IADATE parameter in the OCL instruction. For example:

```
COMPL APPL(TEST01) IADATE(=)
```

The COMPL instruction will complete the application occurrence TEST01 with an input arrival date equal to the current date and with an input arrival time equal to the DEFIAT parameter value.

Example 4

The SORT parameter specifies whether OCL is to consider the application occurrence with the latest or the earliest input arrival date and time. For example:

```
COMPL APPL(TEST01)
```

The current plan includes the following occurrences of application TEST01:

```
Input arrival date: 97/12/01 time: 18.00  
Input arrival date: 97/12/01 time: 20.00  
Input arrival date: 97/12/03 time: 18.00
```

If SORT is MAX, OCL selects the occurrence with input arrival 97/12/03 1800. If SORT is MIN, OCL selects the occurrence with input arrival 97/12/01 1800.

The SORT parameter is used only if IA, IADATE, and IATIME keywords are not specified in the OCL instruction.

Obtaining access authorization

Many of the OCL instructions use the scheduler programming interface. You need authorization to use the programming interface requests. If you are not authorized to use the instruction you need, give the relevant access type and RACF resource code to your administrator. Table 122 lists the OCL instructions that access the scheduler resources and the access authorization you need for each.

Note: Use the MODOP instruction as follows:

```
MODOP APPL(applname) IADATE(iadate) IATIME(iatime) OPNO(operationnumber)
```

One of the following parameters must be specified:

WSNAME, ASUB, AJR, TIMEDEP, CLATE, OPIA, and OPDL

Otherwise warning message EQQCL4VW “No parameter was specified” is issued.

Table 122. Access Authorizations

| Instruction | Access Authorization Required |
|---|-------------------------------|
| INIT | None |
| CHKAPPL CHKDATE | Read |
| ADD ADDCOND ADDOP ADDPRED ADDRES ADDSIMP CHGEXTNAME CHGJOB CHGOPSAI COMPL DEL DELCOND DELPRED DELSIMP DELRES FORCE HOLD KILLJOB KILLREC JSUACT MODCOND MODOP NOP OPSTAT RELEASE RELOP RELSUCC SETUPD SRSTAT UNNOP UPD WSSTAT | Update |

OCL libraries can be protected by a security product, such as RACF, which allows only authorized users can run the OCL program. Because OCL instructions can be

issued from within IBM Workload Scheduler for z/OS-scheduled jobs, the product should be authorized to access the OCL libraries.

Use the TSOCMD parameter to avoid having to specify TSO commands or REXX instructions other than those provided by OCL.

Logging executed instructions

For each instruction executed, OCL writes messages in the data sets referred to by the SYSTSPRT DD card and by the OCLLOG DD card. These messages describe the instruction that have been executed by OCL. The data set referred to by the OCLLOG DD card must be allocated with LRECL 133 and RECFM FB and used with disposition MOD.

The logging function requires the OCLLOG DD card to be specified in the OCL procedure. For an example OCL procedure, see “EQQYRPRC sample procedure” on page 266.

Note: Instructions are logged only if the OCLLOG DD card is specified.

An example of the log records follows:

```
07/30 13:41:33 EQQCL00I Instruction : INIT VARTAB(&OADID)
07/30 13:41:34 EQQCL02I INIT instruction executed : RC=0
07/30 13:41:34 EQQCL00I Instruction : SETUPD VAR1 = 'PIPP0'
07/30 13:41:34 EQQCL02I SETUPD instruction executed : RC=0
07/30 13:41:34 EQQCL00I Instruction : COMPL APPL(TEST01)
07/30 13:41:35 EQQCL02E COMPL instruction executed : RC=8
07/30 13:41:35 EQQCL00I Instruction : CHKAPPL APPL(TEST02)
07/30 13:41:35 EQQCL02W CHKAPPL instruction executed : RC=4
07/30 13:41:35 EQQCL00I Instruction : IF RESULT = 8 THEN NOP
```

An example of the messages written in the SYSTSPRT DD card follows:

```
EQQCL01I =====
EQQCL00I Processing: CHKAPPL APPL(TEST01) STATUS(S)
EQQCL0JI Searching for the occurrence TEST01 in CP
EQQCL00I Occurrence found: APPL(TEST01) IA(9709080800) STATUS(S)
EQQCL00I Occurrence found: APPL(TEST01) IA(9709080930) STATUS(S)
EQQCL0KI Total n. of matching occurrences: 2
EQQCL01I =====
EQQCL00I Processing: IF RESULT=0 THEN ADD APPL(TEST02)
EQQCL03I True condition: IF RESULT = 0
EQQCL01I =====
EQQCL00I Processing: ADD APPL(TEST02)
EQQCL0AI The occurrence was successfully added: APPL(TEST02) IA(9709091657
```

Specifying OCL instructions

To correctly specify OCL instructions in your JCL, observe the following rules:

1. When you specify two or more REXX instructions on the same line, separate one from the next by a semicolon followed immediately by one or more blank characters (;). For example:

```
IF VAR1 = 'OK' THEN ADD APPL(TEST01); ELSE GOTO ESCI
```

2. Specify all the instructions in column 1 through column 72.

3. If the instruction continues on the next line, end the first line with a comma (the continuation character). For example:

```
IF RESULT > 0 THEN,
  DEL APPL(TESTAPPL2)
IF SUBSTR(VAR2,3,1) = '0' THEN COMPL APPL(TEST03);,
  ELSE ADD APPL(TEST04)
```

- If you continue a list of keyword values on the next line, end the first line with two consecutive commas (,,). For example:

```
CHKAPPL APPL(TEST01,TEST02,TEST03,TEST04,,
TEST05)
```

- If you need to use IBM Workload Scheduler for z/OS-managed variables, specify the JCL tailoring directive (for example, `//*%OPC SCAN`). These variables will be substituted by the scheduler before it runs the OCL program. For example:

```
//... JOB ...
//*%OPC SCAN
//...
//EQQOCL.SYSIN DD *
if substr(&OYMD1,5,2) = 31 then goto lastdd
:
:
```

In this example, the scheduler substitutes `&OYMD1` with the occurrence input arrival date in `YYMMDD` format.

- Prefix a comment line with an asterisk (*). For example:

```
//EQQOCL.SYSIN DD *
* This is my comment line
:
:
```

- The following instructions store their return code in the variable `RESULT`: `ADD`, `ADDOP`, `ADDPRED`, `ADDRES`, `CALL`, `CHGEXTNAME`, `CHKAPPL`, `CHKDTE`, `CHGJOB`, `CHGOPSAI`, `COMPL`, `DEL`, `DELPRED`, `DELRES`, `FORCE`, `HOLD`, `JSUACT`, `KILLJOB`, `KILLREC`, `MODOP`, `NOP`, `OPSTAT`, `RELEASE`, `RELOP`, `RELSUCC`, `SRSTAT`, `UNNOP` and `WSSTAT`. For example:

```
CHKAPPL APPL(test01,test02)
IF RESULT = 0 then ...
```

- Do not include delimiters, such as parentheses, in descriptive fields, such as `DESC` and `EXTNAME`.

Specifying input arrival dates and times

About this task

In the scheduler, the input arrival time forms part of the key that uniquely identifies each occurrence of the application in the long-term plan and in the current plan; it is not the time that the scheduler attempts to start the occurrence, unless you specify the first operation as time-dependent. The input arrival time of the occurrence is the default input arrival time for operations making up that occurrence. Some of the OCL instructions use the input arrival date and time to uniquely identify the occurrences in the scheduler plans. The keywords that specify the input arrival date and time are:

Table 123. Input Arrival Date and Time Keywords

| Keyword | Abbreviation | Purpose |
|---------|--------------|---|
| IADATE | IAD | Specifies the input arrival date in <code>YYMMDD</code> format |
| IATIME | IAT | Specifies the input arrival time in <code>HHMM</code> format |
| IA | | Specifies the input arrival date and time in <code>YYMMDDHHMM</code> format |

These keywords are optional. You can specify either or both the keywords `IADATE` and `IATIME`, but if you do you cannot then specify the `IA` keyword. OCL does not

check for duplicate keywords in the OCL statements. If a keyword is specified more than once, the values from the last occurrence will be used.

The initialization parameters that affect the calculation of the input arrival date and time if you omit either specification are:

DEFIAT

Specified in the EQQYRPRM member; it determines the default occurrence input arrival time (for example, 1800)

SORT Specified in the EQQYRPRM member or in the INIT instruction. It determines whether OCL is to consider the earliest or the latest date and time of the occurrence in the current plan. For example:

COMPL APPL(TEST01)

The following occurrences of the application TEST01 are in the current plan:

Input arrival date: 971201 time: 18.00
 Input arrival date: 971201 time: 20.00
 Input arrival date: 971203 time: 18.00

If SORT is MAX, OCL selects the occurrence with input arrival 97/12/03 18.00.

If SORT is MIN, OCL selects the occurrence with input arrival 97/12/01 18.00.

The SORT parameter is used only if IADATE, IATIME, and IA keywords are not specified in the OCL instruction.

Table 124 describes how OCL uses the IADATE, IATIME, and IA keywords:

Table 124. How OCL Uses the IADATE, IATIME, and IA Keywords

| Keywords | Input arrival date and time calculated by OCL | Comment |
|-----------------------|---|---|
| IAD(971203) | 9712031800 | Uses the default input arrival time, if it is specified in the DEFIAT parameter (for example, 1800); otherwise it uses the current time (for example, 1015) |
| IAD(971203) IAT(2030) | 9712032030 | |
| IAD(=) | 9712041800 | Uses the current date and the default input arrival time (i.e. 1800), if it is specified in the DEFIAT parameter; otherwise, it uses the current time |
| IAD(=) IAT(=) | 9712041015 | Uses the current date and time |
| IAD(971203) IAT=() | 9712031015 | Uses the current time |
| IAT(=) | 9712041015 | Uses the current date and time |
| IAT(2030) | 9712042030 | Uses the current date |
| IA(9712032030) | 971232030 | |
| IA(=) | 9712031015 | Uses the current date and time |

Note: If no keyword is specified, OCL selects the earliest or the latest occurrence of the application in the scheduler plans, according to the SORT parameter specified in the EQQYRPRM member or in the INIT instruction.

The keywords listed in Table 124 on page 173 can be specified in the following OCL instructions:

| | | | | |
|---------|-----------|---------|---------|---------|
| ADD | ADDCOND | ADDOP | ADDPRED | ADDRES |
| ADDSIMP | CHGEXTNME | CHGJOB | CHKAPPL | COMPL |
| DEL | DELCOND | DELPRED | DELRES | DELSIMP |
| FORCE | HOLD | KILLJOB | KILLREC | MODCOND |
| MODOP | NOP | RELEASE | RELOP | RELSUCC |
| UNNOP | | | | |
| | | | | |

You can use the SUBSYS() keyword to specify the scheduler subsystem name in any of the following:

- The EQQYRPRM member
- The INIT instruction
- The EQQYPARM DD card if it is specified in the OCL procedure

OCL uses the following order to override the scheduler subsystem name:

1. SUBSYS() specified in the data set referenced by the EQQYPARM DD card
2. SUBSYS() specified in an OCL instruction (for example, OPSTAT)
3. SUBSYS() specified in the INIT instruction
4. SUBSYS() specified in the OCL PARMLIB

The SUBSYS() keyword specified in the EQQYPARM overrides all the scheduler subsystem name specifications.

If duplicate keywords are specified in the OCL instructions, OCL takes the rightmost specification. For example:

```
CHKAPPL APPL(TEST01) IAD(=) APPL(TEST05) STATUS(E)
```

In this example, OCL checks for occurrences of application TEST05.

Substituting variables

About this task

IBM Workload Scheduler for z/OS supports the automatic substitution of variables during job setup and at job submission. IBM Workload Scheduler for z/OS also supplies several standard variables, which you can use in your job. You can create your own variables, which are stored in variable tables in the scheduler database.

The scheduler supplies variables that you can use in the OCL instructions. For example:

```
COMPL APPL(TEST02) IAD(&OYMD1) IAT(&OHHMM)
IF &OHHMM > '1930' THEN ADD APPL(TEST01)
```

The scheduler variables are organized into four groups:

- Occurrence-related variables
- Operation-related variables
- Date-related variables

- Dynamic-format variables

For details on the IBM Workload Scheduler for z/OS-supplied variables, refer to the section “Variable Substitution” in *IBM Workload Scheduler for z/OS: Managing the Workload*.

Table 125 lists the most common IBM Workload Scheduler for z/OS-supplied variables.

Table 125. IBM Workload Scheduler for z/OS-supplied Variables

| Variable | Description |
|----------|---|
| OADID | Application ID |
| ODD | Occurrence input arrival day of month in DD format |
| ODDD | Occurrence input arrival day of the year in DDD format |
| ODMY1 | Occurrence input arrival date in DDMMYY format |
| ODMY2 | Occurrence input arrival date in DD/MM/YY format |
| OHHMM | Occurrence input arrival hour and minute in HHMM format |
| OMMY | Occurrence input arrival month and year in MMY format |
| OYMD | Occurrence input arrival date in YYYYMMDD format |
| OYMD1 | Occurrence input arrival date in YYMMDD format |
| OYMD2 | Occurrence input arrival date in YY/MM/DD format |
| OYMD3 | Occurrence input arrival date in YYYY/MM/DD format |
| OJOBNAME | Operation job name |
| OOPNO | Operation number within the occurrence, right-justified and padded with zeros |
| OWSID | Workstation ID for current operation |
| CDD | Current day of month in DD format |
| CDDD | Day number in the current year |
| CHHMM | Current hour and minute in HHMM format |
| CHHYY | Current month within year in MMY format |
| CYMD | Current date in YYYYMMDD format |
| CYYMMDD | Current date in YYMMDD format |

Description of OCL instructions

About this task

This section described each OCL instruction.

ADD

The ADD instruction adds occurrences or a group of occurrences to the current plan or to the long-term plan. It modifies the current plan or long-term plan, depending on the date and time specified or defaulted to. If the date and time being used is after the end of the current plan, the modification will be made to the long-term plan instead of the current plan.

Table 126 on page 176 describes the keywords that can be used.

Table 126. Keywords Used in the Add Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|-----------|-----------------------|-------------------------|---|--------------|------------|
| APPL() | Required ¹ | Yes | The name of the application to be added. APPL() and GROUP() are mutually exclusive. | | |
| GDEPRES() | Optional | No | Group dependencies resolution option (Y or N). If not specified it defaults to "Y". To be used only when the versioning is present. | | |
| GROUP() | Required ¹ | No | The application group name to be added. GROUP() and APPL() are mutually exclusive. | | |
| IADATE() | Optional | No | The input arrival date of the application occurrence. If not specified, it defaults to the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. It defaults to the current time or, if an occurrence of the application already exists in the current plan, to the first minute after the current time. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence or group. It defaults to the current date and time. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| VARTAB() | Optional | No | The IBM Workload Scheduler for z/OS variable table you want to associate with the group or application. | | |

Note: ¹ Mutually exclusive with another keyword.

Examples:

ADD APPL(TEST01)

Application TEST01 will be added to the current plan or to the long-term plan

ADD GROUP(GROUPTEST01)

All the applications associated with the application group GROUPTEST01 will be added to the current plan or to the long-term plan

ADD APPL(TEST01) IA(=)

Application TEST01 will be added to the current plan or to the long-term plan with input arrival date and time that corresponds to the current date and time

ADD APPL(TEST01) IAD(970708) IAT(0900)

Application TEST01 will be added to the current plan or to the long-term plan with input arrival date 970708 and input arrival time 0900

ADD APPL(TEST01,TEST02,TEST03,TEST04,TEST05,TEST06)

Applications TEST01, TEST02, TEST03, TEST04, TEST05, and TEST06 will be added to the current plan or to the long-term plan

Note: The following message is not to be considered an error by OCL:

EQQM018E OCCURRENCE ALREADY EXISTS - NOT ADDED

It is issued when OCL tries to add an occurrence that already exists in the current plan. In this case, OCL tries again to add the occurrence with an input arrival time incremented by 1 minute.

The ADD instruction returns one of the following return codes:

RESULT = 0

Occurrences added to the current plan or to the long-term plan.

RESULT = 4

Occurrences added to the long-term plan, but unable to resolve all external dependencies.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

ADDCOND

The ADDCOND instruction adds a condition to an operation in the current plan. It modifies the current plan, depending on the date and time specified or defaulted to.

Table 127 describes the keywords that can be used.

Table 127. Keywords used in the Addcond Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-------------|-------------------------|---|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | No | The number of the operation to be modified. It defaults to the default operation number specified in the OCL program DEFOPNO. | OP | |

Table 127. Keywords used in the Addcond Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|-----------|-----------------------|-------------------------|--|--------------|------------|
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time specified in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence. occurrence found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| PREOPNO() | Optional | No | The predecessor operation number. It defaults to the default predecessor operation number specified in the DEFPREOPNO parameter of the EQQYRPRM member. | PO | |
| PREAPPL() | Optional | No | The predecessor application name. It defaults to the application name specified in the application occurrence. | PA | |

Table 127. Keywords used in the Addcond Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|-------------|-------------|-------------------------|---|--------------|--------|
| PREIADATE() | Optional | No | The predecessor application input arrival date. It defaults to the date specified in IADATE(). | PIAD | YYMMDD |
| PREIATIME() | Optional | No | The predecessor application time. It defaults to the time specified in IATIME(). | PIAT | HHMM |
| CONDID() | Optional | No | The number of the condition to be inserted. It defaults to the default condition id specified in the OCL program DEFCONDID. | | |
| COUNT() | Optional | No | Condition counter. Use it to define the rule type: 0 = All the condition dependencies in this condition must be true $n > 0$ = At least n out of the condition dependencies in this condition must be true The default is 0. | | |
| DESC() | Optional | No | Descriptive text. | | |
| CHKTYPE() | Required | No | Check type. Possible values are: RC Return code ST Status The default is ST. | | |
| | | | | | |

Table 127. Keywords used in the Addcond Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-------------|-------------------------|--|--------------|--------|
| LOG() | Required | No | Logical operator: GE = Greater than or equal to. Valid only for RC check type. GT = Greater than. Valid only for RC check type. LE = Less than or equal to. Valid only for RC check type. LT = Less than. Valid only for RC check type. EQ = Equal to. NE = Not equal to. Use it to specify conditions on final statuses only. RG = Range. The default is EQ. | | |
| VALRC() | Required | No | Return code, valid only for RC check type. The default is 0000. | | |
| VALRC2() | Required | No | Return code, valid only for RC check type, as second boundary in a range expressed by the RG logical operator. The default is 9999. | | |
| VALST() | Required | No | Condition status, Valid only for ST check type. The default is C. | | |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

The ADDCOND instruction returns one of the following return codes:

RESULT = 0

Predecessor dependency definition added.

RESULT = 4

Predecessor dependency not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

ADDOP

The ADDOP instruction adds an operation into an occurrence in the current plan.

Table 128 on page 181 describes the keywords that can be used.

Table 128. Keywords used in the Addop Instruction

| Keyword | Requirement | List of Values Allowed | Description | Abbreviation | Format |
|------------|-----------------------|------------------------|--|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Required | No | The operation to be added. | OP | |
| WSNAME() | Required | No | The name of the workstation on which the added operation is executed. | WS | |
| ASUB() | Optional | No | Automatic job submission option (Y or N). | | |
| AUTFUNC | Optional | No | Automated function for the system automation integration. It can be up to 8 characters. | | |
| CLNTYPE() | Optional | No | Cleanup type: A Automatic I Immediate M Manual N None The default is N. | | |
| COMMTEXT | Optional ² | No | Command text for the system automation integration. It can be up to 255 characters. | | |
| COMPINFO | Optional | No | Completion information for the system automation integration. It can be up to 64 characters. | | |
| CONDRJOB() | Optional | No | Specifies if the operation might recover a conditional predecessor (Y or N). The default is N. | | |
| CSCRIPT | Optional | No | Specifies if the operation uses a centralized script (Y or N). If the workstation is not fault-tolerant, the default is Y. | | |
| DESC() | Optional | No | Operation descriptive text. If you do not specify DESC(), it defaults to: '* OCL added this oper *' | | |
| DURATION() | Optional ¹ | No | The operation estimated duration. EDUR() and DURATION are mutually exclusive. | | HHMMSS |

Table 128. Keywords used in the Addop Instruction (continued)

| Keyword | Requirement | List of Values Allowed | Description | Abbreviation | Format |
|-----------|-----------------------|------------------------|---|--------------|------------|
| EDUR() | Optional ¹ | No | The operation estimated duration. EDUR() and DURATION are mutually exclusive. | | HHMM |
| EXPJCL | Optional | No | Expanded JCL used (Y or N). If not specified the default is N. | | |
| EXTNAME() | Optional | No | A free-format name for the operation. It can include blanks and special characters for a maximum of 54 characters. Do not include parentheses in the extended name value. | | |
| EXTSE() | Optional | No | The scheduling environment name. The maximum length can be 16 characters. | | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. It defaults to the earliest or latest input arrival date and time of the application occurrence found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| JOBCRT() | Optional | No | Specifies whether the operation is critical or eligible for WLM assistance, if late. Possible values are: P Critical path. W Eligible for WLM assistance. N Not eligible for WLM assistance. This is the default. | | |

Table 128. Keywords used in the Addop Instruction (continued)

| Keyword | Requirement | List of Values Allowed | Description | Abbreviation | Format |
|-----------|-------------|------------------------|---|--------------|------------|
| JOBNAME() | Optional | No | The jobname. If the workstation is a job setup, printer, or computer workstation, the jobname is required. | JOB | |
| JOBPOL() | Optional | No | Specifies the WLM assistance policy to apply, if the job was defined as critical. Possible values are: D Deadline. The job is assisted if it has not completed at deadline time. L Long Duration. The job is assisted if it runs beyond the estimated duration. S Latest Start Time. The job is assisted if it was submitted before its latest start time. C Conditional. An algorithm is used to decide whether to apply the Deadline or Latest Start Time policy. blank The policy set in the OPCOPTS statement is applied. | | |
| MONITOR | Optional | No | Specifies if the operation is monitored by an external product (Y or N). | | |
| OPDL() | Optional | No | Operation deadline. | | YYMMDDHHMM |
| OPIA() | Optional | No | Operation input arrival time. | | YYMMDDHHMM |
| PREOPNO() | Optional | No | The internal predecessor of the added operation. It defaults to the default operation number specified in the DEFOPNO initialization parameter. | PO | |
| SECELEM | Optional | No | Security element for the system automation integration. It can be up to 8 characters. | | |
| TIMEDEP() | Optional | No | Time-dependent option. | | |
| USRSYS() | Optional | No | User sysout needed (Y or N). If not specified the default is N. | | |

Table 128. Keywords used in the Addop Instruction (continued)

| Keyword | Requirement | List of Values Allowed | Description | Abbreviation | Format |
|---|-------------|------------------------|------------------------|--------------|--------|
| WLMSCLS | Optional | No | The WLM service class. | | |
| ¹ Mutually exclusive with another keyword. ² Required for automation workstations. | | | | | |

Examples:

ADDOP APPL(TEST01) OP(30) PO(20) WS(CPU1) JOB(JOB11111)

Operation 020 will be added to the occurrence TEST01 with the latest or earliest input arrival date and time (depending on the RT parameter specified in PARMLIB or in the INIT instruction)

**ADDOP APPL(TEST01) OP(30) WS(CPU1) JOB(JOB11111) TIMEDEP(Y),
OPIA(&OYMD1.2000) OPDL(&OYMD1.2300)**

Operation 30 will be added as a successor of the default operation number (for example, 001) and it will be scheduled at 08.00 p.m.

**ADDOP APPL(TEST01) WS(FTW1) IA(0310160000) DESC(EXTNAME)
PREOPNO(001) JOBNAME(TEST) OPNO(002) EXTNAME(DAILY PAYROLL
JOB)** Operation 002 will be added to occurrence TEST01 with Extended Name DAILY PAYROLL JOB.

**ADDOP APPL(TEST01) WS(FTW1) IA(0310160000) DESC(Extname Blank)
PREOPNO(001) JOBNAME(TEST) OPNO(002) EXTNAME()**

Operation 002 will be added to occurrence TEST01 with Extended Name *blank* .

The EXTNAME keyword is mandatory when the following two conditions coexist:

- When you add an operation that is to be inserted in the symphony file
- When the TWSJOBNAME value is EXTNAME or EXTNOCC.

Note: You cannot add a Job setup operation.

The ADDOP instruction returns one of the following return codes:

RESULT = 0

Operation was successfully added to the occurrence.

RESULT = 4

Not possible to add the operation.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

ADDPRED

The ADDPRED instruction adds a predecessor operation to an occurrence in the current plan or in the long-term plan. It modifies the current plan or long-term plan, depending on the date and time specified or defaulted to. If the date and time being used is after the end of the current plan, then the modification will be made to the long-term plan instead of the current plan.

Table 129 on page 185 describes the keywords that can be used.

Table 129. Keywords used in the Addpred Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|--|--------------|------------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | No | The number of the operation to be modified. It defaults to the default operation number specified in the OCL program DEFOPNO. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time specified in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence. occurrence found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |

Table 129. Keywords used in the Addpred Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-------------|-------------------------|---|--------------|--------|
| PREOPNO() | Optional | No | The predecessor operation number. It defaults to the default predecessor operation number specified in the DEFPREOPNO parameter of the EQQYRPRM member. | PO | |
| PREAPPL() | Optional | No | The predecessor application name. It defaults to the application name specified in the application occurrence. | PA | |
| PREIADATE() | Optional | No | The predecessor application input arrival date. It defaults to the date specified in IADATE(). | PIAD | YYMMDD |
| PREIATIME() | Optional | No | The predecessor application time. It defaults to the time specified in IATIME(). | PIAT | HHMM |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Examples:

ADDPRED APPL(TEST01) OP(40) PA(TEST00)

The default predecessor operation, DEFPREOPNO, of the application TEST00 will be added as a predecessor of the operation 40 of the application TEST01.

ADDPRED APPL(TEST01) IAD(970708) OP(40) PO(90), PA(TEST00)

The operation 90 of the application TEST00 will be added as a predecessor of the operation 40 of application TEST01 with the input arrival date 970708 and the input arrival time that corresponds to the default IA time, DEFIAT.

ADDPRED APPL(TEST01)

The default predecessor operation, DEFPREOPNO, of the application TEST01 will be added as an external predecessor of the default operation number, DEFOPNO, of the same application.

The ADDPRED instruction returns one of the following return codes:

RESULT = 0

Predecessor dependency definition added.

RESULT = 4

Operation not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

ADDRES

The ADDRES instruction adds a special resource to an operation in the current plan.

Table 130 describes the keywords that can be used.

Table 130. Keywords used in the Addres Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|--|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | No | The number of the operation to be modified. It defaults to the default operation number specified in the DEFOPNO parameter of the EQQYRPRM member. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE=() means the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time that is defined in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |

Table 130. Keywords used in the Address Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|------------|-----------------------|-------------------------|--|--------------|------------|
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter specified in the EQQYRPRM member in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| ONERROR | Optional | No | Keep on error: YES, NO, or blank | OE | |
| ONCOMPL | Optional | No | On complete action. Change resource availability: YES, NO, RESET, or blank | ONC | |
| RESNAME() | Required | No | The special resource to be added. | RN | |
| RESUSAGE() | Required | No | The special resource usage: S Shared X Exclusive | RU | |
| ONERROR() | Required | No | The ONERROR option: YES to keep the special resource, NO to free it, in case of operation ended in error. | | |

Note: ¹ Mutually exclusive with another keyword.

Examples:

ADDRESS APPL(TEST01) RN(prova.sr1) RU(X) OE (N)

Adds the special resource PROVA.SR1 with exclusive use to the default operation number, DEFOPNO, of the occurrence TEST01 and KEEP-ON-ERROR set to no.

ADDRESS APPL(TEST02) IAD(&OYMD1) RN(prova.sr2) RU(X) OP(10), OE(Y)

Adds the special resource PROVA.SR2, with exclusive use and

keep-on-error set to YES, to operation 10 of the occurrence TEST01 with the default input arrival time, DEFIAT

The ADDRES instruction returns one of the following return codes:

RESULT = 0

Special resource added.

RESULT = 4

Occurrence or operation not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

ADDSIMP

The ADDSIMP instruction adds condition dependencies to an operation in the current plan. It modifies the current plan, depending on the date and time specified or defaulted to.

Table 131 describes the keywords that can be used.

Table 131. Keywords used in the Addsimp Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------------|-----------------------|--------------------------------|--|---------------------|---------------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | No | The number of the operation to be modified. It defaults to the default operation number specified in the OCL program DEFOPNO. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time specified in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |

Table 131. Keywords used in the Addsimp Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|-------------|-------------|-------------------------|--|--------------|------------|
| IA() | Optional | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence. occurrence found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| PREOPNO() | Optional | No | The predecessor operation number. It defaults to the default predecessor operation number specified in the DEFPREOPNO parameter of the EQQYRPRM member. | PO | |
| PREAPPL() | Optional | No | The predecessor application name. It defaults to the application name specified in the application occurrence. | PA | |
| PREIADATE() | Optional | No | The predecessor application input arrival date. It defaults to the date specified in IADATE(). | PIAD | YYMMDD |
| PREIATIME() | Optional | No | The predecessor application time. It defaults to the time specified in IATIME(). | PIAT | HHMM |
| CONDID() | Optional | No | The number of the condition to be inserted. It defaults to the default condition id specified in the OCL program DEFCONDID. | | |

Table 131. Keywords used in the Addsimp Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-------------|-------------------------|--|--------------|--------|
| CHKTYPE() | Optional | No | Check type. Possible values are: RC Return code ST Status The default is ST. | | |
| LOG() | Optional | No | Logical operator: GE = Greater than or equal to. Valid only for RC check type. GT = Greater than. Valid only for RC check type. LE = Less than or equal to. Valid only for RC check type. LT = Less than. Valid only for RC check type. EQ = Equal to. NE = Not equal to. Use it to specify conditions on final statuses only. RG = Range. The default is EQ. | | |
| VALRC() | Optional | No | Return code, valid only for RC check type. The default is 0000. | | |
| VALRC2() | Optional | No | Return code, valid only for RC check type, as second boundary in a range expressed by the RG logical operator. The default is 9999. | | |
| VALST() | Optional | No | Condition status, Valid only for ST check type. The default is C. | | |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

The ADDSIMP instruction returns one of the following return codes:

RESULT = 0

Predecessor dependency definition added.

RESULT = 4

Predecessor dependency not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

CALL

The CALL instruction invokes a routine. A result returned by the routine is stored in the variable RESULT. The meaning of the RESULT variable value depends on the routine.

Syntax

CALL *routineparameters*

Examples:

```
CALL CHKDATA &DATACONT.  
CALL test1  
IF RESULT > 0 THEN EXIT 99
```

CHGEXTNAME

The CHGEXTNAME instruction defines or modifies the extended job information associated to the operations of an application occurrence. You can delete the extended name by specifying EXTNAME() and EXTSE().

Table 132 describes the keywords that can be used.

Table 132. Keywords used in the Chgextname Instructions

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|--|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Required | No | The number of the operation to be modified. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes that the default input arrival time is specified in the DEFIAT parameter of the EQQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |

Table 132. Keywords used in the Chgextname Instructions (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-----------------------------------|-------------------------|--|--------------|------------|
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |
| EXTNAME() | Required if EXTSE not specified | No | A free-format name for the operation. It can include blanks and special characters. Do not include parentheses in the extended name value. | | |
| EXTSE() | Required if EXTNAME not specified | No | Scheduling Environment name. | | |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Examples:

CHGEXTNAME APPL(TEST01) EXTNAME('DAILY PAYROLL JOB')

This example sets the extended name DAILY PAYROLL JOB for the operations of TEST01 application.

CHGEXTNAME APPL(TEST01) EXTNAME()

This example deletes the extended name DAILY PAYROLL JOB for the operations of TEST01 application.

Do not use CHGEXTNAME to add the extended name to an operation that is to be included in the Symphony file, when the value of TWSJOBNAME is either EXTNAME or EXTNOCC. Use the ADDOP instruction with the EXTNAME keyword instead.

The CHGEXTNAME instruction returns one of the following return codes:

RESULT = 0

Occurrence modified.

RESULT = 4

Occurrence or operations not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

CHGJOB

The CHGJOB instruction changes the jobname associated to the operations of an application occurrence.

This instruction applies only to jobs running in z/OS environments or to jobs running on a fault-tolerant workstation using the centralized script support.

Table 133 describes the keywords that can be used.

Table 133. Keywords used in the Chgjob Instructions

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|--|--------------|------------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | Yes | The number of the operation to be modified. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes that the default input arrival time is specified in the DEFIAT parameter of the EQQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |
| OLDJOB() | Optional | No | The old jobname | OLDJ | |

Table 133. Keywords used in the Chgjob Instructions (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-------------|-------------------------|--------------------------------|--------------|--------|
| NEWJOB() | Required | No | The new jobname | NEWJ | |
| WSNAME() | Optional | No | The operation workstation name | WS | |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Note: If you omit the selection parameters OPNO(), OLDJOB(), or WSNAME(), OCL will change the jobname of all the operations in the occurrence. You cannot change the jobname of a job setup operation without changing the jobname of the dependent computer operation. In this case, do not use the OPNO() or WSNAME() keywords for applications that contain job setup operations.

Examples:

CHGJOB APPL(TEST01) OLDJ(job11111) NEWJ(job22222)

Changes the jobname to all the operations with JOB11111

CHGJOB APPL(TEST01) IAD(&OYMD1), NEWJ(job22222) OP(20)

Changes the jobname to operation 20

CHGJOB APPL(TEST01) IA(=) NEWJ(job22222) WS(CPU1)

Changes the jobname to all the operations at the workstation CPU1

CHGJOB APPL(TEST01) IAD(970708) IAT(1800), NEWJ(job22222) WS(CPU1) OP(20)

Changes the jobname to operation 20 at the workstation CPU1

The CHGJOB instruction returns one of the following return codes:

RESULT = 0

Occurrence modified.

RESULT = 4

Occurrence or operations not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

CHGOPSAI

The CHGOPSAI instruction defines or modifies the system automation information associated with the operations of an application occurrence. The COMMTEXT keyword always must be specified; you can disable AUTFUNC, COMPINFO, and SECELEM by setting them to blank.

Table 134 describes the keywords that can be used.

Table 134. Keywords used in the Chgopsai Instructions

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-------------|-------------------------|---|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |
| | | | | | |

Table 134. Keywords used in the Chgopsai Instructions (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|--|--------------|------------|
| OPNO() | Required | No | The number of the operation to be modified. | OP | |
| AUTFUNC | Optional | No | Automated function for the system automation integration. It can be up to 8 characters. | | |
| COMMTEXT | Optional ² | No | Command text for the system automation integration. It can be up to 255 characters. | | |
| COMPINFO | Optional | No | Completion information for the system automation integration. It can be up to 64 characters. | | |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |

Table 134. Keywords used in the Chgopsai Instructions (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-----------------------|-------------------------|--|--------------|--------|
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes that the default input arrival time is specified in the DEFIAT parameter of the EQQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |
| SECELEM | Optional | No | Security element for the system automation integration. It can be up to 8 characters. | | |
| ¹ Mutually exclusive with another keyword. ² Required, if not previously specified. | | | | | |

Examples:

**CHGOPSAI APPL(TEST01) COMMTEXT(DAILY PAYROLL JOB)
SECELEM(AAA)**

Sets the command text and security element for the operations of application TEST01

CHGOPSAI APPL(TEST01) AUTFUNC()

Deletes the automated function for the operations of application TEST01

The CHGOPSAI instruction returns one of the following return codes:

RESULT = 0

Occurrence modified.

RESULT = 4

Occurrence or operations not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

CHKAPPL

Syntax

CHKAPPL keyword

The CHKAPPL instruction verifies the presence of application occurrences in the current plan. Use the STATUS keyword to verify the application occurrences in a specific status.

If you specify the STATUS keyword as E (Error), you can send a message reporting the operation error code to TSO users.

Table 135 on page 198 describes the keywords that can be used.

Table 135. Keywords used in the Chkappl Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|---|--------------|------------|
| APPL() | Required | Yes | The name of the application to be checked. Wild characters are accepted, for example: APPL(TEST*) or APPL(T%%T*) | | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE() means the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time that is defined in the DEFIAT parameter. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date time of the application occurrence. IA(=) assumes the current date and time. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| STATUS() | Optional | No | The occurrence status can be: C Complete E An operation has ended in error D Deleted P A pending predecessor exists for the occurrence S Started U Undecided W No operations in the occurrence have started | ST | |
| OPNO() | Optional | Yes | The operation number. This keyword has effect only with STATUS(E). | OP | |

Table 135. Keywords used in the Chkappl Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-------------|-------------------------|---|--------------|--------|
| ALERT() | Optional | No | <p>The alert option (YES or NO):</p> <p>ALERT(YES) A message will be sent to the TSO users specified in the USER() keyword. YES is the default.</p> <p>ALERT(NO) No message will be sent.</p> <p>This keyword has effect only with STATUS(E).</p> | | |
| USER() | Optional | Yes | <p>The users who are to receive the alert message. This keyword is required only with ALERT(YES).</p> | | |

Table 135. Keywords used in the Chkappl Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-------------|-------------------------|---|--------------|--------|
| ERRCODE() | Optional | Yes | The operation code can be: CAN CCUN FAIL JCCE JCL JCLI MCP OFxx (xx is the status and extended status) OAUT OJCV OSEQ OSUB OSUF OSUP OSxx (xx is the status and extended status) PCAN <i>nnnn</i> (step return code) Sxxx (system abend code) Uxxx (user abend code in hexadecimal notation) xxxx (user-defined error code) yyyy (distributed agent error code) This keyword has effect only with STATUS(E) | ERR | |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Examples:

CHKAPPL APPL(TEST01)

All the occurrences of application TEST01

CHKAPPL IAD(&OYMD1) APPL(TEST02)

The occurrences of application TEST02 with the input arrival date of the occurrence that executes the OCL instructions and with input arrival time corresponding to the current time

CHKAPPL IA(=) APPL(TEST02)

The occurrences of application TEST02 with the input arrival date and time corresponding to the current date and time

CHKAPPL APPL(TEST02) IAT(1830)

All the occurrences of application TEST02 with the input arrival time 18.30 and with the input arrival date corresponding to the current date

CHKAPPL APPL(TEST01) STATUS(E) ALERT(YES) USER(ibm001,ibm002)

All the occurrences of application TEST02 in Error status; a message will be sent to TSO users IBM001 and IBM002

CHKAPPL APPL(TEST01) STATUS(E) USER(ibm001,ibm002), ERR(JCL,JCLI) OP(10,15,20)

Checks whether operation 10, 15, or 20 ended with error code JCL or JCLI; if any did, a message will be sent to users IBM001 and IBM002

Note: It is not possible to specify a variable (which is not an IBM Workload Scheduler for z/OS variable) in the IA, IAT and IAD parameters of the OCL CHAPPL instruction, as in the following example:

```
SET VAR4=9912241830
CHKAPPL APPL(TEST01) IA(&VAR4)
```

CHKAPPL returns one of the following return codes:

RESULT = 0

Occurrences found in the current plan.

RESULT = 4

Occurrences not found or no operations that ended in error were found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

CHKDATE

The CHKDATE instruction verifies the validity of an input date and derives variables from the input date. It accepts dates in the following formats:

Keyword**Formats****DATE1()**

YYYYMMDD, YYMMDD, YYYY/MM/DD, and YY/MM/DD

DATE2()

DDMMYYYY, DDMMYY, DD/MM/YYYY, and DD/MM/YY

If the date is valid, the CHKDATE instruction provides the value of the following variables that are based on the input date. The value reported in the table is calculated with an input date equal to 1997/09/28.

Table 136. Chkdate Instruction Variables

| Variable | Description | Value |
|----------|--|------------|
| XYMD | Date in YYYYMMDD format | 19970928 |
| XYMD1 | Date in YYMMDD format | 970928 |
| XYMD2 | Date in YY/MM/DD format | 97/09/28 |
| XYMD3 | Date in YYYY/MM/DD format | 1997/09/28 |
| XDMY | Date in DDMMYYYY format | 28091997 |
| XDMY1 | Date in DDMMYY format | 280997 |
| XDMY2 | Date in DD/MM/YY format | 28/09/97 |
| XDMY3 | Date in DD/MM/YYYY format | 28/09/1997 |
| XDAY | Day of the week: 1-7; 1 = Monday, 2 = Tuesday... | 7 |
| XDAYD | Day of the week (MON, TUE, WED, THU, FRI, SAT, or SUN) | SUNDAY |

Table 136. Chkdate Instruction Variables (continued)

| Variable | Description | Value |
|------------------|--|-----------|
| XDD | Day of the month in DD format | 28 |
| XDDD | Day of the year in DDD format (Julian day) | 271 |
| XDDDDY | Date as a Julian date in DDYYYY format | 27197 |
| XDDDDYYY | Date as a Julian date in DDDDDYYY format | 2711997 |
| XDE | Ten-day period in the year | 28 |
| XFDAY | First day of the month: 1 = Monday, 2 = Tuesday... | 1 |
| XFDAYD | First day of the month | MONDAY |
| XFDAYJ | First day of the month in DDD format (Julian day) | 244 |
| XFDAYJ1 | First day of the month in YYYYDDD format | 1997244 |
| XFF | Half of the month: 1 = first half, 2 = second half | 2 |
| XFIRSTDE | First day of a 10-day period in a month | 19970921 |
| XFIRSTQ | First day of the quarter in YYYYMMDD format | 19970901 |
| XLASTDD | Last day of the month in DD format | 30 |
| XLASTDDN | Last day of the next month | 31 |
| XLASTDDP | Last day of the previous month | 31 |
| XLASTDD1 | Last day of the month in YYYYMMDD format | 19970930 |
| XLASTDD2 | Last day of the month in YYMMDD format | 970930 |
| XLASTDD3 | Last day of the month in DDMMYYYY format | 30091997 |
| XLASTDD4 | Last day of the month in DDMMYY format | 300997 |
| XLSTDDJ | Last day of the month in DDD format (Julian day) | 273 |
| XLSTDDJ1 | Last day of the month in YYYYDDD format | 1997273 |
| XMM | Month in MM format | 09 |
| XMMYY | Month and year in MMY format | 0997 |
| XMY | Date in MMYYYY format | 091997 |
| XMMNAME | Name of the month | SEPTEMBER |
| XMMP1 | Previous month in MM format | 08 |
| XMMN1 | Next month in MM format | 10 |
| XQUARTER | Quarter of the year | 3 |
| XTOTWWM | Number of weeks in the month: 1-6 | 5 |
| XWW ¹ | Week of the year | 39 |
| XWWD | Week of the year and day of the week | 397 |
| XWWLAST | Last week of the month: Y or N | N |
| XWWMONTH | Week of the month: 1-6 | 4 |
| XYM | Date in YYMM format | 199709 |
| XYDDDD | Date as a Julian date in YYDDD format | 97271 |
| XYDDDDDD | Date as a Julian date YYYYDDD format | 1997271 |
| XY | Year in YY format | 97 |
| XY | Year in YYYY format | 1997 |
| XYMM | Year and month in YYMM format | 9709 |
| XYYP | Previous year in YY format | 96 |

Table 136. Chkdate Instruction Variables (continued)

| Variable | Description | Value |
|----------|---|----------|
| XYYYYP | Previous year in YYYY format | 1996 |
| XYYMMP | Previous month in YYMM format | 9708 |
| XYYYYMMP | Previous month in YYYYMM format | 199708 |
| XYYN | Next year YY format | 98 |
| XYYYYN | Next year in YYYY format | 1998 |
| XYYMMN | Next month in YYMM format | 9710 |
| XYYYYMMN | Next month in YYYYMM format | 199710 |
| X1MOND | First Monday of the month in YYYYMMDD format | 19970901 |
| X1MONDJ | First Monday of the month in DDD format (Julian day) | 244 |
| X1MONDJ1 | First Monday of the month in YYYYDDD format (Julian day) | 1997244 |
| X2MOND | Second Monday of the month in YYYYMMDD format | 19970908 |
| X2MONDJ | Second Monday of the month in DDD format (Julian day) | 251 |
| X2MONDJ1 | Second Monday of the month in YYYYDDD format (Julian day) | 1997251 |
| X3MOND | Third Monday of the month in YYYYMMDD format | 19970915 |
| X3MONDJ | Third Monday of the month in DDD format (Julian day) | 258 |
| X3MONDJ1 | Third Monday of the month in YYYYDDD format (Julian day) | 1997258 |
| X4MOND | Fourth Monday of the month in YYYYMMDD format | 19970922 |
| X4MONDJ | Fourth Monday of the month in DDD format (Julian day) | 265 |
| X4MONDJ1 | Fourth Monday of the month in YYYYDDD format (Julian day) | 1997265 |
| X5MOND | Fifth Monday of the month in YYYYMMDD format | 19970929 |
| X5MONDJ | Fifth Monday of the month in DDD format (Julian day) | 272 |
| X5MONDJ1 | Fifth Monday of the month in YYYYDDD format (Julian day) | 1997272 |
| XFREEDAY | F = free day, W = work day | F |
| XYMDN1 | Calendar day + 1 in YYYYMMDD format | 19970929 |
| XDDN1 | Calendar day + 1 in DD format | 29 |
| XYMDN1 | Calendar day + 1 in DDMMYYYY format | 29091997 |
| XYMDN2 | Calendar day + 2 in YYYYMMDD format | 19970930 |
| XYMDN3 | Calendar day + 3 in YYYYMMDD format | 19971001 |
| XYMDN4 | Calendar day + 4 in YYYYMMDD format | 19971002 |
| XYMDN5 | Calendar day + 5 in YYYYMMDD format | 19971003 |
| XYMDN6 | Calendar day + 6 in YYYYMMDD format | 19971004 |
| XYMDN7 | Calendar day + 7 in YYYYMMDD format | 19971005 |
| XYMDN8 | Calendar day + 8 in YYYYMMDD format | 19971006 |
| XYMDN9 | Calendar day + 9 in YYYYMMDD format | 19971007 |
| XYMDN10 | Calendar day + 10 in YYYYMMDD format | 19971008 |
| XYMDN11 | Calendar day + 11 in YYYYMMDD format | 19971009 |
| XYMDN12 | Calendar day + 12 in YYYYMMDD format | 19971010 |
| XYMDN13 | Calendar day + 13 in YYYYMMDD format | 19971011 |
| XYMDN14 | Calendar day + 14 in YYYYMMDD format | 19971012 |
| XYMDN15 | Calendar day + 15 in YYYYMMDD format | 19971013 |

Table 136. Chkdate Instruction Variables (continued)

| Variable | Description | Value |
|----------|--------------------------------------|----------|
| XYMDN16 | Calendar day + 16 in YYYYMMDD format | 19971014 |
| XYMDN17 | Calendar day + 17 in YYYYMMDD format | 19971015 |
| XYMDN18 | Calendar day + 18 in YYYYMMDD format | 19971016 |
| XYMDN19 | Calendar day + 19 in YYYYMMDD format | 19971017 |
| XYMDN20 | Calendar day + 20 in YYYYMMDD format | 19971018 |
| XYMDN21 | Calendar day + 21 in YYYYMMDD format | 19971019 |
| XYMDN22 | Calendar day + 22 in YYYYMMDD format | 19971020 |
| XYMDN23 | Calendar day + 23 in YYYYMMDD format | 19971021 |
| XYMDN24 | Calendar day + 24 in YYYYMMDD format | 19971022 |
| XYMDN25 | Calendar day + 25 in YYYYMMDD format | 19971023 |
| XYMDN26 | Calendar day + 26 in YYYYMMDD format | 19971024 |
| XYMDN27 | Calendar day + 27 in YYYYMMDD format | 19971025 |
| XYMDN28 | Calendar day + 28 in YYYYMMDD format | 19971026 |
| XYMDN29 | Calendar day + 29 in YYYYMMDD format | 19971027 |
| XYMDN30 | Calendar day + 30 in YYYYMMDD format | 19971028 |
| XWDDN1 | Work day + 1 in DD format | 29 |
| XWDMYN1 | Work day + 1 in DDMMYYYY format | 29091997 |
| XWYMDN1 | Work day + 1 in YYYYMMDD format | 19970929 |
| XWYMDN2 | Work day + 2 in YYYYMMDD format | 19970930 |
| XWYMDN3 | Work day + 3 in YYYYMMDD format | 19971001 |
| XWYMDN4 | Work day + 4 in YYYYMMDD format | 19971002 |
| XWYMDN5 | Work day + 5 in YYYYMMDD format | 19971003 |
| XWYMDN6 | Work day + 6 in YYYYMMDD format | 19971006 |
| XWYMDN7 | Work day + 7 in YYYYMMDD format | 19971007 |
| XWYMDN8 | Work day + 8 in YYYYMMDD format | 19971008 |
| XWYMDN9 | Work day + 9 in YYYYMMDD format | 19971009 |
| XWYMDN10 | Work day + 10 in YYYYMMDD format | 19971010 |
| XWYMDN11 | Work day + 11 in YYYYMMDD format | 19971013 |
| XWYMDN12 | Work day + 12 in YYYYMMDD format | 19971014 |
| XWYMDN13 | Work day + 13 in YYYYMMDD format | 19971015 |
| XWYMDN14 | Work day + 14 in YYYYMMDD format | 19971016 |
| XWYMDN15 | Work day + 15 in YYYYMMDD format | 19971017 |
| XWYMDN16 | Work day + 16 in YYYYMMDD format | 19971020 |
| XWYMDN17 | Work day + 17 in YYYYMMDD format | 19971021 |
| XWYMDN18 | Work day + 18 in YYYYMMDD format | 19971022 |
| XWYMDN19 | Work day + 19 in YYYYMMDD format | 19971023 |
| XWYMDN20 | Work day + 20 in YYYYMMDD format | 19971042 |
| XWYMDN21 | Work day + 21 in YYYYMMDD format | 19971027 |
| XWYMDN22 | Work day + 22 in YYYYMMDD format | 19971028 |
| XWYMDN23 | Work day + 23 in YYYYMMDD format | 19971029 |

Table 136. Chkdate Instruction Variables (continued)

| Variable | Description | Value |
|----------|--------------------------------------|----------|
| XWYMDN24 | Work day + 24 in YYYYMMDD format | 19971030 |
| XWYMDN25 | Work day + 25 in YYYYMMDD format | 19971031 |
| XWYMDN26 | Work day + 26 in YYYYMMDD format | 19971103 |
| XWYMDN27 | Work day + 27 in YYYYMMDD format | 19971104 |
| XWYMDN28 | Work day + 28 in YYYYMMDD format | 19971105 |
| XWYMDN29 | Work day + 29 in YYYYMMDD format | 19971106 |
| XWYMDN30 | Work day + 30 in YYYYMMDD format | 19971107 |
| XYDDP1 | Calendar day - 1 in DD format | 27 |
| XYMDP1 | Calendar day - 1 in YYYYMMDD format | 19970927 |
| XYMDP2 | Calendar day - 2 in YYYYMMDD format | 19970926 |
| XYMDP3 | Calendar day - 3 in YYYYMMDD format | 19970925 |
| XYMDP4 | Calendar day - 4 in YYYYMMDD format | 19970924 |
| XYMDP5 | Calendar day - 5 in YYYYMMDD format | 19970923 |
| XYMDP6 | Calendar day - 6 in YYYYMMDD format | 19970922 |
| XYMDP7 | Calendar day - 7 in YYYYMMDD format | 19970921 |
| XYMDP8 | Calendar day - 8 in YYYYMMDD format | 19970920 |
| XYMDP9 | Calendar day - 9 in YYYYMMDD format | 19970919 |
| XYMDP10 | Calendar day - 10 in YYYYMMDD format | 19970918 |
| XYMDP11 | Calendar day - 11 in YYYYMMDD format | 19970917 |
| XYMDP12 | Calendar day - 12 in YYYYMMDD format | 19970916 |
| XYMDP13 | Calendar day - 13 in YYYYMMDD format | 19970915 |
| XYMDP14 | Calendar day - 14 in YYYYMMDD format | 19970914 |
| XYMDP15 | Calendar day - 15 in YYYYMMDD format | 19970913 |
| XYMDP16 | Calendar day - 16 in YYYYMMDD format | 19970912 |
| XYMDP17 | Calendar day - 17 in YYYYMMDD format | 19970916 |
| XYMDP18 | Calendar day - 18 in YYYYMMDD format | 19970910 |
| XYMDP19 | Calendar day - 19 in YYYYMMDD format | 19970909 |
| XYMDP20 | Calendar day - 20 in YYYYMMDD format | 19970908 |
| XYMDP21 | Calendar day - 21 in YYYYMMDD format | 19970907 |
| XYMDP22 | Calendar day - 22 in YYYYMMDD format | 19970906 |
| XYMDP23 | Calendar day - 23 in YYYYMMDD format | 19970905 |
| XYMDP24 | Calendar day - 24 in YYYYMMDD format | 19970904 |
| XYMDP25 | Calendar day - 25 in YYYYMMDD format | 19970903 |
| XYMDP26 | Calendar day - 26 in YYYYMMDD format | 19970902 |
| XYMDP27 | Calendar day - 27 in YYYYMMDD format | 19970901 |
| XYMDP28 | Calendar day - 28 in YYYYMMDD format | 19970831 |
| XYMDP29 | Calendar day - 29 in YYYYMMDD format | 19970830 |
| XYMDP30 | Calendar day - 30 in YYYYMMDD format | 19970829 |
| XWDDP1 | Work day - 1 in DD format | 26 |
| XWDMYP1 | Work day - 1 in DDMMYYYY format | 26091997 |

Table 136. Chkdate Instruction Variables (continued)

| Variable | Description | Value |
|----------|----------------------------------|----------|
| XWYMDP1 | Work day - 1 in YYYYMMDD format | 19970926 |
| XWYMDP2 | Work day - 2 in YYYYMMDD format | 19970925 |
| XWYMDP3 | Work day - 3 in YYYYMMDD format | 19970924 |
| XWYMDP4 | Work day - 4 in YYYYMMDD format | 19970923 |
| XWYMDP5 | Work day - 5 in YYYYMMDD format | 19970922 |
| XWYMDP6 | Work day - 6 in YYYYMMDD format | 19970919 |
| XWYMDP7 | Work day - 7 in YYYYMMDD format | 19970918 |
| XWYMDP8 | Work day - 8 in YYYYMMDD format | 19970917 |
| XWYMDP9 | Work day - 9 in YYYYMMDD format | 19970916 |
| XWYMDP10 | Work day - 10 in YYYYMMDD format | 19970915 |
| XWYMDP11 | Work day - 11 in YYYYMMDD format | 19970912 |
| XWYMDP12 | Work day - 12 in YYYYMMDD format | 19970911 |
| XWYMDP13 | Work day - 13 in YYYYMMDD format | 19970910 |
| XWYMDP14 | Work day - 14 in YYYYMMDD format | 19970909 |
| XWYMDP15 | Work day - 15 in YYYYMMDD format | 19970908 |
| XWYMDP16 | Work day - 16 in YYYYMMDD format | 19970905 |
| XWYMDP17 | Work day - 17 in YYYYMMDD format | 19970904 |
| XWYMDP18 | Work day - 18 in YYYYMMDD format | 19970903 |
| XWYMDP19 | Work day - 19 in YYYYMMDD format | 19970902 |
| XWYMDP20 | Work day - 20 in YYYYMMDD format | 19970901 |
| XWYMDP21 | Work day - 21 in YYYYMMDD format | 19970829 |
| XWYMDP22 | Work day - 22 in YYYYMMDD format | 19970828 |
| XWYMDP23 | Work day - 23 in YYYYMMDD format | 19970827 |
| XWYMDP24 | Work day - 24 in YYYYMMDD format | 19970826 |
| XWYMDP25 | Work day - 25 in YYYYMMDD format | 19970825 |
| XWYMDP26 | Work day - 26 in YYYYMMDD format | 19970822 |
| XWYMDP27 | Work day - 27 in YYYYMMDD format | 19970821 |
| XWYMDP28 | Work day - 28 in YYYYMMDD format | 19970820 |
| XWYMDP29 | Work day - 29 in YYYYMMDD format | 19970819 |
| XWDYMD30 | Work day - 30 in YYYYMMDD format | 19970818 |

¹ XWW is the variable updated by CHKDATE to show the week of the year, in terms of week number. It is a fixed calculation that starts the first day of the year and has a fixed length of seven days. It is assumed that week number 1 begins on January 1st, regardless of what day of the week it is.

The IBM Workload Scheduler for z/OS-supplied variable are calculated on the occurrence input arrival date and time, whereas the OCL variables are calculated on an input date, such as a date provided with a promptable variable.

The following table describes the keywords that can be used.

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|-----------|-----------------------|-------------------------|---|--------------|--------|
| DATE1() | Required ¹ | No | The input date in the following formats: YYYYMMDD, YYYY/MM/DD, YYMMDD, or YY/MM/DD. DATE1() and DATE2() are mutually exclusive. | | YYMMDD |
| DATE2() | Required ¹ | No | The input date in the following formats: DDMMYYYY, DD/MM/YYYY, DDMMYY, or DD/MM/YY. DATE2() and DATE1() are mutually exclusive. | | DDMMYY |
| CALNAME() | Optional | No | The scheduler default calendar ID. It defaults to DEFAULT. | CAL | |
| SUBSYS() | Optional | No | The scheduler subsystem name. It overrides the subsystem ID specified in a previous INIT instruction or in the EQQYRPRM member, except when you specify the SUBSYS() keyword in the EQQYPARM DD card. | | |
| MSG() | Optional | No | It can be YES or NO. It specifies whether the program is to display the description and the values of the calculated variables in the OCL SYSOUT. It defaults to YES. | | |

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---|-------------|-------------------------|--|--------------|--------|
| INCVALUE() | Optional | No | This value indicates the decreasing value for the variables calculation. OCL will add days to, or subtract days from, an input date up to the <i>incvalue</i> . (OCL will calculate the new date +/- <i>n</i> work days or calendar days, where <i>n</i> is a value in the range 1- <i>incvalue</i> .) <i>incvalue</i> can be a value in the range 1-30 and overlays the value specified in PARMLIB. | INC | |
| Note: Mutually exclusive with another keyword. | | | | | |

Examples:

```
CHKDATE DATE2(&datac) MSG(NO) INC(15) CAL(calend01)
CHKDATE DATE2(201097) MSG(NO)
```

The CHKDATE instruction can be an extension to the scheduler SETVAR directive. For example, you can use it to perform a double arithmetic calculation on an input date. If you need to calculate a date that corresponds to the occurrence input arrival date plus 2 workdays and plus 20 calendar days, you can use the following OCL instructions:

```
CHKDATE DATE1(&OYMD1) INC(2) MSG(NO)
CHKDATE DATE1(XWYMDN2) INC(20)
SETUPD var1 = XYMDN20
```

The first CHKDATE instruction calculates the occurrence input arrival date plus 2 workdays. The result is stored in the variable XWYMDN2. The second CHKDATE instruction uses the variable XWYMDN2 as the input date to calculate the new date: XWYMDN2 + 20 calendar days. The result is stored in variable XYMDN20, which you can use, for example, to update the default value of a variable in a variable table.

The CHKDATE instruction returns one of the following return codes:

RESULT = 0

Valid date.

RESULT = 8

Invalid date. Refer to the error messages.

Note: When using the CHKDATE instruction, the variable name must be put between single quotation marks. For example:

```
CHKDATE DATE1('XWYMDN2') INC(20)
```

COMPL

The COMPL instruction completes occurrences or operations within an occurrence in the current plan.

Table 137 describes the keywords that can be used.

Table 137. Keywords used in the Compl Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|---|--------------|--------|
| APPL() | Required | No | The name of the application to be completed. | | |
| OPNO() | Optional | Yes | The number of the operation to be completed. If OPNO() is specified then COMPL will complete the operations and not the occurrence. If the operation is the last one in the occurrence, then the occurrence will be completed. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE=() means the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE=() is specified, IATIME() assumes the default input arrival time that is defined in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |

Table 137. Keywords used in the Compl Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-----------------------|-------------------------|---|--------------|------------|
| IA() | Optional ¹ | N | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Note: if you do not specify IADATE(), IATIME(), or IA(), OCL considers only the application occurrences in the current plan that do not have Complete status.

Examples:

COMPL APPL(TEST01,TEST02,TEST03)

Completes the occurrences TEST01, TEST02, and TEST03 with the earliest or latest input arrival date and time, depending on the SORT parameter

COMPL APPL(TEST01) IAD(=)

Completes the occurrence TEST01 with the input arrival date corresponding to the current date and with the default input arrival time

COMPL APPL(TEST01,TEST02,TEST03) IA(=)

Completes the occurrences TEST01, TEST02 and TEST03 with the input arrival date and time corresponding to the current date and time

COMPL APPL(TEST02) IAD(970709) OP(10,40)

Completes operations 10 and 40 of the occurrence TEST02 with the input arrival date 970709 and with the default input arrival time

The COMPL instruction returns one of the following return codes:

RESULT = 0

Occurrences or operations completed

RESULT = 4

Occurrences or operations not found

RESULT = 8

Invalid instruction or PIF problem. See the error messages.

DEL

The DEL instruction deletes an occurrence or an operation from the current plan or from the long-term plan. It modifies the current plan or long-term plan, depending on the date and time specified or defaulted to. If the date and time being used is after the end of the current plan, the modification will be made to the long-term plan instead of the current plan.

Table 138 describes the keywords that can be used.

Table 138. Keywords used in the Del Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|---|--------------|--------|
| APPL() | Required | No | The name of the application to be deleted. | | |
| OPNO() | Optional | Yes | The number of the operation to be deleted. If OPNO() is specified, DEL will delete the operations and not the occurrence. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE=() means the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE=() is specified, IATIME() assumes the default input arrival time that is defined in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |

Table 138. Keywords used in the Del Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-----------------------|-------------------------|---|--------------|------------|
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Examples:

DEL APPL(TEST01)

Deletes the occurrence with the earliest or latest input arrival date and time of the application TEST01 (according to the SORT parameter or the INIT instruction)

DEL APPL(TEST01) IAD(=) IAT(1700)

Deletes the occurrence of application TEST01 with the input arrival date corresponding to the current date and with the input arrival time 1700

DEL APPL(TEST01) IAD(970708)

Deletes the occurrence TEST01 with the input arrival date 970708 and the input arrival time corresponding to the default IA time, DEF

DEL APPL(TEST01) IA(=)

Deletes the occurrence TEST01 with the input arrival date and time corresponding to the current date and time

DEL APPL(TEST01) IA(9707081800)

Deletes the occurrence TEST01 with the input arrival date 970708 and time 1800

DEL APPL(TEST01) OP(10,30)

Deletes operations 10 and 30 of the occurrence TEST01 with the earliest or latest input arrival date and time, depending on the SORT parameter

Note: If the occurrence is associated with an application group, DEL will remove the group definition from the occurrence before deleting the occurrence.

The DEL instruction returns one of the following return codes:

RESULT = 0

Occurrence deleted.

RESULT = 4

Occurrence found but operations not found in the current plan.

RESULT = 8

Occurrence not found in the current plan. Invalid instruction or PIF problem. Refer to the error messages.

DELCOND

The DELCOND instruction adds a condition to an operation in the current plan. It modifies the current plan, depending on the date and time specified or defaulted to.

Table 139 describes the keywords that can be used.

Table 139. Keywords used in the Delcond Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------------|-----------------------|--------------------------------|--|---------------------|---------------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | No | The number of the operation to be modified. It defaults to the default operation number specified in the OCL program DEFOPNO. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time specified in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| | | | | | |

Table 139. Keywords used in the Delcond Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-------------|-------------------------|--|--------------|------------|
| IA() | Optional | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence. occurrence found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| CONDID() | Optional | No | The number of the condition to be inserted. It defaults to the default condition id specified in the OCL program DEFCONDID. | | |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

The DELCOND instruction returns one of the following return codes:

RESULT = 0

Predecessor dependency definition added.

RESULT = 4

Predecessor dependency not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

DELPRED

The DELPRED instruction deletes the external predecessors of an occurrence in the current plan or in the long-term plan. It modifies the current plan or long-term plan, depending on the date and time specified or defaulted to. If the date and time being used is after the end of the current plan, the modification will be made to the long-term plan instead of the current plan.

Table 140 describes the keywords that can be used.

Table 140. Keywords used in the Delpred Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-------------|-------------------------|---|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |

Table 140. Keywords used in the Delpred Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|------------|-----------------------|-------------------------|---|--------------|------------|
| OPNO() | Optional | Yes | The number of the operation to be modified. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE=() means the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time that is defined in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| EXCLOPNO() | Optional | Yes | The operation number that must not be modified. | EOP | |
| EXCLPRE() | Optional | Yes | The predecessor application name that must not be deleted. | EPRE | |

Note: ¹ Mutually exclusive with another keyword.

Examples:

DELPRED APPL(TEST01)

All the predecessor dependencies will be deleted from occurrence TEST01 with the latest or earliest input arrival date and time, depending on the SORT parameter specified in PARMLIB or in the INIT instruction.

DELPRED APPL(TEST01) IAD(970708) OP(40,50)

All the predecessor dependencies will be deleted from operations 40 and 50 of the occurrence TEST01 with the input arrival date 970708 and the input arrival time that corresponds to the current time.

DELPRED APPL(TEST01) IA(=) EXCLOP(20)

All the predecessor dependencies will be deleted from the operations other than operation 20 of the occurrence TEST01 with the input arrival date and time that corresponds to the current date and time.

DELPRED APPL(TEST01) IA(=) EPRE(TEST00)

All the predecessor dependencies other than occurrence TEST00 will be deleted from the occurrence EST01 with the input arrival date and time that corresponds to the current date and time.

The DELPRED instruction returns one of the following return codes:

RESULT = 0

Predecessor dependency definition deleted.

RESULT = 4

Occurrence or operation not found in the current plan or in the long-term plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

DELRES

The DELRES instruction deletes one or more special resources from an operation in the current plan.

Table 141e describes the keywords that can be used.

Table 141. Keywords used in the Delres Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-------------|-------------------------|---|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | Yes | The number of the operation to be modified. It defaults to the default operation number, DEFOPNO, specified in the OCL program. | OP | |

Table 141. Keywords used in the Delres Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|-----------|-----------------------|-------------------------|---|--------------|------------|
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE=() means the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time that is defined in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter specified in the EQQYRPRM member or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| EXCLRES() | Optional | Yes | The special resource that must not be deleted. | ERES | |
| RESNAME() | Optional | No | The special resource to be deleted. Do not include parentheses in the resource name. | RN | |

Note: ¹ Mutually exclusive with another keyword.

Examples:

DELRES APPL(TEST01)

Deletes all the special resources from the operations of the occurrence TEST01 with the latest or earliest input arrival date and time, depending on the SORT parameter specified in the EQQYRPRM member or in the INIT instruction

DELRES APPL(TEST01) IAD(=) OP(10)

Deletes all the special resources from operation 10 of the occurrence TEST01 with the input arrival date corresponding to the current date and with the input arrival time corresponding to the default input arrival time, DEFIAT

DELRES APPL(TEST01) IA(=) OP(10) ERES(PROVA.SR1)

Deletes all the special resources, except for special resource PROVA.SR1, from operation 10 of the occurrence TEST01, with the input arrival date and time corresponding to the current date and time.

The DELRES instruction returns one of the following return codes:

RESULT = 0

Special resource deleted.

RESULT = 4

Operation not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

DELSIMP

The DELSIMP instruction deletes condition dependencies of an operation in the current plan. It modifies the current plan, depending on the date and time specified or defaulted to.

Table 142 describes the keywords that can be used.

Table 142. Keywords used in the Delsimp Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-------------|-------------------------|---|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | No | The number of the operation to be modified. It defaults to the default operation number specified in the OCL program DEFOPNO. | OP | |

Table 142. Keywords used in the Delsimp Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|-----------|-----------------------|-------------------------|--|--------------|------------|
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time specified in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence. occurrence found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| PREOPNO() | Optional | No | The predecessor operation number. It defaults to the default predecessor operation number specified in the DEFPREOPNO parameter of the EQQYRPRM member. | PO | |
| PREAPPL() | Optional | No | The predecessor application name. It defaults to the application name specified in the application occurrence. | PA | |

Table 142. Keywords used in the Delsimp Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|-------------|-------------|-------------------------|--|--------------|--------|
| PREIADATE() | Optional | No | The predecessor application input arrival date. It defaults to the date specified in IADATE(). | PIAD | YYMMDD |
| PREIATIME() | Optional | No | The predecessor application time. It defaults to the time specified in IATIME(). | PIAT | HHMM |
| CONDID() | Optional | No | The number of the condition to be inserted. It defaults to the default condition id specified in the OCL program DEFCONDID. | | |
| CHKTYPE() | Optional | No | Check type. Possible values are: RC Return code ST Status The default is ST. | | |
| LOG() | Optional | No | Logical operator: GE = Greater than or equal to. Valid only for RC check type. GT = Greater than. Valid only for RC check type. LE = Less than or equal to. Valid only for RC check type. LT = Less than. Valid only for RC check type. EQ = Equal to. NE = Not equal to. Use it to specify conditions on final statuses only. RG = Range. The default is EQ. | | |
| VALRC() | Optional | No | Return code, valid only for RC check type. The default is 0000. | | |
| VALRC2() | Optional | No | Return code, valid only for RC check type, as second boundary in a range expressed by the RG logical operator. The default is 9999. | | |

Table 142. Keywords used in the Delsimp Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-------------|-------------------------|---|--------------|--------|
| VALST() | Optional | No | Condition status, Valid only for ST check type. The default is C. | | |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

The DELSIMP instruction returns one of the following return codes:

RESULT = 0

Predecessor dependency definition added.

RESULT = 4

Predecessor dependency not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

EXIT

Syntax

EXIT *result* | *return_code*

The EXIT instruction terminates the program with a specified return code, which you can specify in the instruction. The value can be a number or RESULT. You can use this instruction to set a return code, which you can then test, using the JCL COND or IF parameters, to determine which job steps to execute.

Examples:

```
//SYSIN DD *
* OCL instructions
IF VAR2 = 'KO' THEN EXIT 99
EXIT RESULT
:
/*
//TESTIF IF (RC = 99) THEN
//STEP002 EXEC PGM=MYPROG
//...
//TESTIF ENDIF
```

FORCE

The FORCE instruction forces the execution of an occurrence or of an operation within an occurrence in the current plan.

Table 143 describes the keywords that can be used.

Table 143. Keywords used in the Force Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-------------|-------------------------|--|--------------|--------|
| APPL() | Required | No | The name of the application to be started. | | |

Table 143. Keywords used in the Force Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|---|--------------|------------|
| OPNO() | Optional | Yes | The number of the operation to be started. It defaults to the default operation number, DEFOPNO, specified in the OCL program. ² | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE=() means the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time that is defined in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | N | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |

Table 143. Keywords used in the Force Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---|-------------|-------------------------|-------------|--------------|--------|
| <p>Notes:</p> <ol style="list-style-type: none"> 1. Mutually exclusive with another keyword. 2. In order to start the occurrence from a specific operation number, the program performs the following actions: <ol style="list-style-type: none"> a. Deletes all the special resources associated to the operation b. Deletes all the external predecessors c. Changes the operation status to R (ready), and changes the following automatic options of the operation: <ul style="list-style-type: none"> • Job submission to YES • Time dependent to NO • Operation descriptive text to '* OCL forced this oper *' • Parallel server to 1 • Resource R1 to 0 • Resource R2 to 0 <p>If the change is successful, the return code is 0. (The return code is stored in the RESULT variable.) Otherwise it does the following:</p> <ol style="list-style-type: none"> 1) Deletes the external predecessors of all the internal predecessors of the specified operation 2) Changes again the operation status to R (ready) and the operation automatic options d. Checks whether the occurrence is started. | | | | | |

Examples:

FORCE APPL(TEST01)

Forces the default operation, DEFOPNO, to start

FORCE APPL(TEST01) OP(50)

Forces operation 50 to start

FORCE APPL(TEST01) IA(=)

Forces operation 50 to start, using the current date and time as the input arrival date and time

FORCE APPL(TEST01) IA(9707081800) OP(70)

Forces operation 70 to start, using the input arrival date 970708 and the input arrival time 18.00

FORCE APPL(TEST01) IAD(=)

Forces the default operation, DEFOPNO, to start, using the current date as the input arrival date and the default input arrival time

FORCE APPL(TEST01) IAD(&OYMD1)

Forces the default operation, DEFOPNO, to start, using the input arrival date and time value of variable &OYMD1

FORCE APPL(TEST01) IAD(970708) IAT(&OHHMM) OP(60)

Forces operation 60 to start, using the input arrival date 970708 and the input arrival time value of variable &OHHMM

The FORCE instruction returns one of the following return codes:

RESULT = 0

Occurrence or operations status is started.

RESULT = 4

Occurrence found but operations not found in the current plan.

RESULT = 8

Occurrence not found in the current plan. Invalid instruction or PIF problem. Refer to the error messages.

GOTO

The GOTO instruction transfers control to a point specified by a LABEL instruction within the OCL program.

Syntax

GOTO *label_name*

Example:

```
IF &var1 = 'OK' then GOTO esci
:
:
LABEL esci
:
```

HOLD

The HOLD instruction holds an operation or all the operations of an occurrence in the current plan.

Table 144 describes the keywords that can be used.

Table 144. Keywords used in the Hold Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|--|--------------|--------|
| APPL() | Required | Yes | The name of the application to be modified. | | |
| OPNO() | Optional | Yes | The number of the operation to be modified. If OPNO() is not specified, the HOLD command, MH, will be issued to all the occurrence operations. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |

Table 144. Keywords used in the Hold Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-----------------------|-------------------------|---|--------------|------------|
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes that the default input arrival time is specified in the DEFIAT parameter of the EQQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Example control statements:

HOLD APPL(TEST01)

Sets to HOLD status all operations of occurrence TEST01

HOLD APPL(TEST01) OP(10,30,50)

Sets to hold status operations 10, 30, and 50 of occurrence TEST01

HOLD APPL(TEST01) IAD(=)

Sets to HOLD status all operations of occurrence TEST01 for the current input arrival date and the default input arrival time

HOLD APPL(TEST01) IAD(=) IAT(1800)

Sets to HOLD status all operations of occurrence TEST01 for the current input arrival date and input arrival time 18.00

HOLD APPL(TEST01) IAD(970708)

Sets to HOLD status all operations of occurrence TEST01 for input arrival date 970708 and the default input arrival time

HOLD APPL(TEST01) IA(=)

Sets to HOLD status all operations of occurrence TEST01 for the current input arrival date and time

HOLD APPL(TEST01) IA(9707081801)

Sets to HOLD status all operations of occurrence TEST01 for input arrival date 970708 and input arrival time 18.01

The HOLD instruction returns one of the following return codes:

RESULT = 0

Operations status modified.

RESULT = 4

Occurrence found but operations not found in the current plan.

RESULT = 8

Occurrence not found in the current plan. Invalid instruction or PIF problem. Refer to the error messages.

IF-THEN-ELSE

Syntax

IF *expression* THEN *instruction1*; ELSE *instruction2*

IF-THEN-ELSE conditionally processes an instruction depending on the evaluation of the expression. The expression is evaluated and results in 0 (false) or 1 (true). The instruction after the THEN clause is processed only if the result is 1 (true). If you specify an ELSE clause, the instruction after the ELSE clause is processed only if the result of the evaluation is 0 (false).

Example:

```
IF result > 0 then exit 99; else ADD(TEST01)
```

Note:

1. If the ELSE clause is on the same line as the last instruction of the THEN clause, it must be preceded by a semicolon (;) followed immediately by a blank character.
2. If the ELSE clause is on the line following the last instruction of the THEN clause, that last THEN clause line must end with a semicolon followed immediately by a comma.

Example:

```
if substr(&OYMD1,5,2) = 30 then GOTO DAY30;,  
else EXIT 50
```

The ELSE clause binds to the nearest IF instruction at the same level.

The expression can contain concatenation, comparison, and logical operators. An operator represents an operation, such as addition, to be performed on one or two terms.

Comparison operations

About this task

The comparison operators compare two terms and return the value 1 if the result of the comparison is true, or 0 if the result is false.

The comparison operators and operations are:

= True if the terms are equal (numerically or when padded, and so forth)

\=, \=, /=

True if the terms are not equal (inverse of =)

> Greater than

< Less than

>< Greater than or less than (the same as not equal)

<> Greater than or less than (the same as not equal)

>= Greater than or equal to

\<, \< Not less than

<= Less than or equal to

\>, \> Not greater than

== True if the terms are strictly equal (identical)

\==, \==, /=

True if the terms are not strictly equal (this is the inverse of ==)

>> Strictly greater than

<< Strictly less than

\<< Strictly not less than

\>>, \>>

Strictly not greater than

A character string has the value false if it is 0, and true if it is 1. The logical operators take one or two such values (values other than 0 or 1 are not allowed) and return 0 or 1 as appropriate:

| | | |
|---|-----|----------------------------------|
| & | AND | Returns 1 if both terms are true |
| | OR | Returns 1 if either term is true |

The instruction can be one of the following:

| | | | | |
|------------|--------|----------|--------------|---------|
| ADD | ADDOP | ADDPRED | ADDRES | CALL |
| CHGEXTNAME | CHGJOB | CHGOPSAI | CHKAPPL | CHKDATE |
| COMPL | DEL | DELPRED | DELRES | EXIT |
| FORCE | GOTO | HOLD | IF-THEN-ELSE | INIT |
| JSUACT | LABEL | MODOP | NOP | OPSTAT |
| RELEASE | RELOP | RELSUCC | SET | SETUPD |
| SRSTAT | UNNOP | UPD | WSSTAT | WTO |

Here are some examples:

```
IF RESULT = 8 THEN EXIT 70; ELSE COMPL APPL(TEST01)
IF LEFT(&VAR1,1) = 1 THEN ADD APPL(TEST01)
IF substr(VAR1,5,2) = 'XX' then COMPL APPL(TEST02)
IF VAR4 = VAR1||VAR2||VAR3 THEN UPD VAR4
IF VAR5 > VAR4 THEN SET VAR6 = VAR3 + 2
IF substr(&OYMD1,3,4) = '0612' THEN goto GIU12
IF &OHHMM > '1830' THEN GOTO T1830
```

The IF instruction returns one of the following return codes:

- CC = 0**
Instruction correctly processed
- CC = 8**
Syntax error

INIT

Syntax

INIT VARTAB() | SUBSYS() | SORT()

The INIT instruction overrides the IBM Workload Scheduler for z/OS subsystem name and the SORT parameter specified in the EQQYRPRM member, and specifies the variable table name used by the UPD or SETUPD instructions.

Table 145 describes the keywords that can be used:

Table 145. Keywords used in the Init Instruction

| Keyword | Requirement | Multiple Values Allowed | Description |
|----------|-------------|-------------------------|---|
| VARTAB() | Optional | No | The scheduler variable table to be updated. This keyword is required only by the UPD and SETUPD instructions. |
| SUBSYS() | Optional | No | The scheduler subsystem name. It overrides the subsystem name specified in the EQQYRPRM member, unless you specify the SUBSYS() keyword in the EQQYPARM DD. |
| SORT() | Optional | No | This keyword overrides the SORT parameter specified in the EQQYRPRM member. It determines the application occurrence in the current plan to be selected if you do not specify the input arrival date and time in the OCL instruction. SORT(MAX) determines the occurrence with the latest input arrival date and time. SORT(MIN) determines the occurrence with the earliest input arrival date and time. |

Examples:

```
INIT VARTAB(&OADID) SUBSYS(OPCC)  
INIT SUBSYS(OPCC)  
INIT SORT(MAX)
```

INIT returns one of the following return codes:

- CC = 0**
Instruction correctly processed
- CC = 8**
Invalid instruction. Refer to the error message.

JSUACT

The JSUACT instruction invokes the TSO JSUACT command, which you can use to activate or inactivate the job submission function in the z/OS environment, in the distributed environment, or in both environments.

Table 146 describes the keywords that can be used:

Table 146. Keywords used in the JSUACT Instruction

| Keyword | Requirement | Multiple Values Allowed | Description |
|----------|-------------|-------------------------|--|
| ACT() | Required | No | If you want to activate the job submission function specify YES, otherwise NO. |
| SUBSYS() | Optional | No | The name of the tracker the JSUACT is directed to. This parameter can be four characters in length. The first character must be alphabetic; the remaining characters must be alphanumeric. All lowercase characters are converted to uppercase. If you specify MSTR, the JSUACT command is directed to all scheduler subsystems on the z/OS system where the JSUACT command was issued. Default is OPCA. |
| TRACE() | Optional | No | Event tracing indicator. When a nonzero positive number is specified, a trace entry is created for each event generated by the JSUACT command. The trace record is written to the message log file identified by ddname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 does not generate trace records. |
| TYPE() | Optional | | Indicates whether the job submission must be deactivated. Possible values are: H To deactivate job submission in the z/OS environment. It is the default value. F To deactivate job submission in the distributed environment. B To deactivate job submission in both environments. |

Example:

```
JSUACT ACT(YES) SUBSYS(OPCB) TYPE(H)
```

In this example the JSUACT instruction is used to activate the job submission function in the z/OS environment.

The JSUACT instruction returns one of the following return codes:

RESULT = 0

Job submission function changed

RESULT = 8
Invalid instruction

KILLJOB

The KILLJOB instruction stops a job that is already running. Applies only to operations running on IBM Workload Scheduler for z/OS Agents or on distributed workstations that are directly connected to the end-to-end server (OPCMASTER).

This action can be taken only on STARTED jobs that are in the EXECUTING status, so that their operation number is known. The application number is required.

Table 147 describes the keywords that can be used.

Table 147. Keywords used in the Killjob Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|---|--------------|--------|
| APPL() | Required | Yes | The name of the application to be modified. | | |
| OPNO() | Optional | Yes | The number of the operation to be modified. If OPNO() is not specified, the command will be issued to all the occurrence operations. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes that the default input arrival time is specified in the DEFIAT parameter of the EQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |

Table 147. Keywords used in the Killjob Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-----------------------|-------------------------|---|--------------|------------|
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence occurrences found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Example control statements:

KILLJOB APPL(TEST01)

Kills all operations of occurrence TEST01

KILLJOB APPL(TEST01) OP(10,30,50)

Kills operations 10, 30, and 50 of occurrence TEST01

KILLJOB APPL(TEST01) IAD(=)

Kills all operations of occurrence TEST01 for the current input arrival date and the default input arrival time

KILLJOB APPL(TEST01) IAD(=) IAT(1800)

Kills all operations of occurrence TEST01 for the current input arrival date and input arrival time 18.00

KILLJOB APPL(TEST01) IAD(090709)

Kills all operations of occurrence TEST01 for input arrival date 090709 and the default input arrival time

KILLJOB APPL(TEST01) IA(=)

Kills all operations of occurrence TEST01 for the current input arrival date and time

KILLJOB APPL(TEST01) IA(0907081801)

Kills all operations of occurrence TEST01 for input arrival date 970708 and input arrival time 18.01

The KILLJOB instruction returns one of the following return codes:

RESULT = 0

Operations killed.

RESULT = 4

Occurrence found but operations not found in the current plan.

RESULT = 8

Occurrence not found in the current plan. Invalid instruction or PIF problem. Refer to the error messages.

KILLREC

The KILLREC instruction stops a recovery job that is already running. Applies only to operations running on distributed workstations that are directly connected to the end-to-end server (OPCMaster).

This action can be taken only on recovery jobs that are in the EXECUTING status, so that their operation number is known. The application number is required.

Table 148 describes the keywords that can be used.

Table 148. Keywords used in the Killrec Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|---|--------------|--------|
| APPL() | Required | Yes | The name of the application to be modified. | | |
| OPNO() | Optional | Yes | The number of the operation to be modified. If OPNO() is not specified, the command will be issued to all the occurrence operations. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes that the default input arrival time is specified in the DEFIAT parameter of the EQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |

Table 148. Keywords used in the Killrec Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-----------------------|-------------------------|---|--------------|------------|
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence occurrences found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Example control statements:

KILLREC APPL(TEST01)

Kills all recovery jobs of occurrence TEST01

KILLREC APPL(TEST01) OP(312)

Kills recovery job 312 of occurrence TEST01

KILLREC APPL(TEST01) IAD(=) IAT(1800)

Kills all recovery jobs of occurrence TEST01 for the current input arrival date and input arrival time 18.00

KILLREC APPL(TEST01) IAD(090708)

Kills all recovery jobs of occurrence TEST01 for input arrival date 090708 and the default input arrival time

KILLREC APPL(TEST01) IA(=)

Kills all recovery jobs of occurrence TEST01 for the current input arrival date and time

The KILLREC instruction returns one of the following return codes:

RESULT = 0

Operations killed.

RESULT = 4

Occurrence found but operations not found in the current plan.

RESULT = 8

Occurrence not found in the current plan. Invalid instruction or PIF problem. Refer to the error messages.

LABEL

The LABEL instruction defines a label within the OCL program that is the target of a GOTO instruction.

Syntax

LABEL *label_name*

Example:

```
IF &var1 = 'OK' then GOTO esci
:
:
LABEL esci
:
:
LABEL esci2
```

Note: If OCL processes a LABEL instruction without branching to it in response to a GOTO instruction, the return code is the highest set by any of the OCL instructions.

The LABEL instruction returns one of the following return codes:

LASTRC

Highest return code returned by the program routines if OCL processed the label without branching to it

CC = 8

Label name not specified

MODCOND

The MODCOND instruction adds a condition to an operation in the current plan. It modifies the current plan, depending on the date and time specified or defaulted to.

Table 149 describes the keywords that can be used.

Table 149. Keywords used in the Modcond Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|---|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | No | The number of the operation to be modified. It defaults to the default operation number specified in the OCL program DEFOPNO. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |

Table 149. Keywords used in the Modcond Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|--|--------------|------------|
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time specified in the DEFIAT parameter of the EQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence. occurrence found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| CONDID() | Optional | No | The number of the condition to be inserted. It defaults to the default condition id specified in the OCL program DEFCONDID. | | |
| COUNT() | Optional | No | Condition counter. Use it to define the rule type: 0 = All the condition dependencies in this condition must be true n>0 = At least n out of the condition dependencies in this condition must be true The default is the current value. | | |
| DESC() | Optional | No | Descriptive text. | | |

Table 149. Keywords used in the Modcond Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-------------|-------------------------|-------------|--------------|--------|
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

The MODCOND instruction returns one of the following return codes:

RESULT = 0

Predecessor dependency definition added.

RESULT = 4

Predecessor dependency not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

MODOP

The MODOP instruction changes the operation's details.

Table 150 describes the keywords that can be used to select the operations to be modified.

Table 150. Keywords used in the Modop Instructions

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|------------|-------------|-------------------------|--|--------------|--------|
| APPL() | Required | Yes | The name of the application to be modified. | | |
| CLNTYPE() | Optional | No | Cleanup type: A Automatic I Immediate M Manual N None If not specified, the default is N. | | |
| CONDRJOB() | Optional | No | Specifies if the operation might recover a conditional predecessor (Y or N). The default is N. | | |
| EXPJCL | Optional | No | Expanded JCL used (Y or N). If not specified the default is N. | | |

Table 150. Keywords used in the Modop Instructions (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|--|--------------|------------|
| OPNO() | Optional | Yes | The number of the operation to be modified. If you omit this parameter, the process changes all the operations in the selected occurrence, regardless of any default operation number, DEFOPNO, specified in the OCL program. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time specified in the DEFIAT parameter of the EQQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |

Table 150. Keywords used in the Modop Instructions (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-------------|-------------------------|---|--------------|--------|
| JOBCRT() | Optional | No | Specifies whether the operation is critical or eligible for WLM assistance, if late. Possible values are: P Critical path. W Eligible for WLM assistance. N Not eligible for WLM assistance. This is the default. | | |
| JOBPOL() | Optional | No | Specifies the WLM assistance policy to apply, if the job was defined as critical. Possible values are: D Deadline. The job is assisted if it has not completed at deadline time. L Long Duration. The job is assisted if it runs beyond the estimated duration. S Latest Start Time. The job is assisted if it was submitted before its latest start time. C Conditional. An algorithm is used to decide whether to apply the Deadline or Latest Start Time policy. blank The policy set in the OPCOPTS statement is applied. | | |
| MONITOR | Optional | No | Specifies if the operation is monitored by an external product (Y or N). | | |
| | | | | | |

Table 150. Keywords used in the Modop Instructions (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-------------|-------------------------|---|--------------|--------|
| USRSYS() | Optional | No | User sysout needed (Y or N). If not specified the default is N. | | |
| WLMSCLS | Optional | No | The WLM service class. | | |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Table 151 describes the operation details that can be modified.

Table 151. Operations Details that can be modified

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|-----------|-----------------------|-------------------------|--|--------------|------------|
| JOBNAME() | Optional | No | The job name | JOB | |
| WSNAME() | Optional | No | The workstation name | WS | |
| DESC() | Optional | No | The operation descriptive text | | |
| EDUR() | Optional ¹ | No | The operation estimated duration. EDUR() and DURATION() are mutually exclusive. | | HHMM |
| ASUB() | Optional | No | The automatic job submission option (Y or N) | | |
| AJR() | Optional | No | The hold/release option (Y or N) | | |
| TIMEDP() | Optional | No | The time dependent option (Y or N) | | |
| CLATE() | Optional | No | The cancel-if-late option (Y or N) | | |
| OPIA() | Optional | No | The operation input arrival date and time | | YYMMDDHHMM |
| OPDL() | Optional | No | The operation deadline | | YYMMDDHHMM |
| PSUSE() | Optional | No | The number of parallel servers required by the operation | | |
| R1USE() | Optional | No | A value in the range 0–99, indicating the maximum capacity of workstation resource 1 | | |

Table 151. Operations Details that can be modified (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|------------|-----------------------|-------------------------|--|--------------|--------|
| R2USE() | Optional | No | A value in the range 0–99, indicating the maximum capacity of workstation resource 2 | | |
| DURATION() | Optional ¹ | No | The operation estimated duration. EDUR() and DURATION() are mutually exclusive. | | HHMMSS |

Note: ¹ Mutually exclusive with another keyword.

Examples:

MODOP APPL(TEST01) OP(10) TIMEDEP(N)

Changes the time dependent option for operation 10

**MODOP APPL(TEST01) IAD(&OYMD1), JOB(job22222) OP(20)
OPIA(&OYMD1.1600)**

Changes the operation input arrival time

The MODOP instruction returns one of the following return codes:

RESULT = 0

Occurrence modified.

RESULT = 4

Occurrence found but operations not found in the current plan, or the operation input arrival or deadline are not valid, but are accepted. Refer to the warning message.

RESULT = 8

Occurrence not found in the current plan. Invalid instruction or PIF problem. Refer to the error messages.

Note: Using the MODOP instruction as follows:

MODOP APPL(APPLNAME) IADATE(IADATE) IATIME(IATIME) OPNO(OPNO)

one of the following parameters must be specified:

- AJR
- ASUB
- CLATE
- CLNTYPE
- CONDRJOB
- DESC
- DURATION
- EDUR
- EXPJCL
- JOBCRT
- JOBNAME
- JOBPOL
- MONITOR
- OPDL

- OPIA
- PSUSE
- R1USE
- R2USE
- TIMEDEP
- USRSYS
- WLMSCLS
- WSNAME

Otherwise the message, EQQCL4VW NO PARAMETER WAS ISSUED, message is issued.

NOP

The NOP instruction removes an operation that is already in the current plan. When a NOP operation is ready to be started, IBM Workload Scheduler for z/OS immediately sets it to C status. The operation is not submitted and successor operations are eligible to start. The NOP instruction can be issued for any operation that has status A, R, *, W, or, for computer workstations with automatic reporting only, C. NOP operations are identified by the N extended status code. If you want to restore the operation, use the UNNOP instruction. The NOP instruction is not supported in operations running on fault-tolerant workstations.

Table 152 describes the keywords that can be used.

Table 152. Keywords used in the Nop Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|--|--------------|--------|
| APPL() | Required | Yes | The name of the application to be modified. | | |
| OPNO() | Optional | Yes | The number of the operation to be modified. If OPNO() is not specified, the NOP command, NP, will be issued to all the occurrence operations. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes that the default input arrival time is specified in the DEFIAT parameter of the EQQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |

Table 152. Keywords used in the Nop Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-----------------------|-------------------------|--|--------------|------------|
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |

Note: ¹ Mutually exclusive with another keyword.

Example control statements:

NOP APPL(TEST01)

Sets all the operations of occurrence TEST01 in NOP status

NOP APPL(TEST01) OP(10,30,50)

Sets operations 10, 30, and 50 of occurrence TEST01 in NOP status

NOP instruction returns one of the following return codes:

RESULT = 0

Operations status modified.

RESULT = 4

Occurrence found but operations not found in the current plan.

RESULT = 8

Occurrence not found in the current plan. Invalid instruction or PIF problem. Refer to the error messages.

OPSTAT

The OPSTAT instruction invokes the TSO command OPSTAT, which you can use to change the status of an operation at any workstation, except workstations that have the nonreporting attribute. Events generated by OPSTAT are matched against operations on the ready list. Events received for operations in waiting, W, or complete, C, status are ignored. Jobs and tasks that are running are always allowed to finish. If there is more than one operation at the workstation, you can optionally specify the ADID, IA, OPNUM, or JOBNAME parameters to identify the particular operation whose status is to be changed.

Table 153 shows the keywords that can be specified.

Table 153. Keywords used in the Opstat Instruction

| Keyword | Requirement | Description | Abbreviation | Format |
|---------|-------------|--|--------------|--------|
| ADID() | Required | The application identifier of the operation whose status you want to change. | A | |

Table 153. Keywords used in the Opstat Instruction (continued)

| Keyword | Requirement | Description | Abbreviation | Format |
|-------------|-------------|--|--------------|--------|
| CLASS() | Optional | For a printer workstation, specifies the printer SYSOUT class of the operation whose status you want to change. | | |
| DURATION() | Optional | If you are specifying STATUS(C) to set the operation status to complete, you can optionally specify a duration for the completed operation. You specify the duration in hours and minutes, in the format hhmm. | | HHMM |
| FORM() | | For a printer workstation, specifies the printer FORM name of the operation whose status you want to change. | | |
| OPNUM() | Optional | The operation number of the operation whose status you want to change. | | |
| ERRORCODE() | Optional | If you are specifying STATUS(E) to set the operation status to ended-in-error, you can optionally specify an error code for the operation. The error code can be any 4 characters. | | |
| EVDATE() | Optional | The date of this operation status event. You can use the EVDATE parameter to indicate to the scheduler that the operation changed status at a time other than the current date. If you do not specify this parameter, the operation is considered to have changed status on the date the scheduler processed the OPSTAT command. | | YYMMDD |
| EVTIME() | Optional | The time of this operation status event. You can use the EVTIME parameter to indicate to the scheduler that the operation changed status at a time other than the current time. If you do not specify this parameter, the operation is considered to have changed status at the time the scheduler processed the OPSTAT command. | | HHMM |

Table 153. Keywords used in the Opstat Instruction (continued)

| Keyword | Requirement | Description | Abbreviation | Format |
|-----------|-------------|---|--------------|------------|
| IA() | Optional | The input arrival date and time of the occurrence that contains the operation whose status you want to change. It defaults to the current date and current time or to the current date and default input arrival time, if the default is specified in the DEFIAT parameter of the EQYRPRM member. | | YYMMDDHHMM |
| JOBNAME() | Optional | The job name associated with the operation whose status you want to change. | J | |
| NUMJOB() | Optional | Use this optional parameter to specify a job number for an operation. Specify a number in the range 0–99999. The scheduler builds a job number in the format USR#####, padding the number with zeros on the left if you specify fewer than 5 digits. | | |
| STATUS | Optional | The operation status that you want to set. You can change the operation status to C (operation completed successfully). Note: You cannot use the OPSTAT command to change an operation from status W to status C, because predecessor jobs might not have completed. | ST | |
| SUBSYS() | Optional | The name of the tracker subsystem to which the OPSTAT instruction is directed. It defaults to the subsystem name specified in the OCL OPTRK initialization parameter. | | |
| TOKEN() | Optional | The token assigned for the operation whose status you want to change. A token is automatically assigned for operations started on workstations that specify a user-defined destination ID. The token can be used to uniquely identify the operation. | | |
| | | | | |

Table 153. Keywords used in the Opstat Instruction (continued)

| Keyword | Requirement | Description | Abbreviation | Format |
|----------|-------------|---|--------------|--------|
| TRACE() | Optional | Event tracing indicator. When a positive number is specified, a trace entry is created for each event generated by the OPSTAT command. The trace record is written to the message log file identified by ddname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 will not generate trace records. | | |
| WSNAME() | Optional | The name of the workstation for which you are reporting the status of an operation. | W | |

The SUBSYS keyword defaults to the IBM Workload Scheduler for z/OS subsystem name (tracker) specified in the OCL SUBSYS initialization parameter

Example:

```
OPSTAT W(BDEC) ST(C) J(DNCD3000) A(ACLMSDLY)
```

In this example an operation for application ACLMSDLY at workstation BDEC is reported as completed.

The OPSTAT instruction returns one of the following return codes:

RESULT = 0

Operation status modified

RESULT = 8

Invalid instruction

PROMPTN

The PROMPTN instruction specifies that NO is the reply to a recovery prompt issued for an abended operation.

Table 154 describes the keywords that can be used.

Table 154. Keywords used in the PROMPTN Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-------------|-------------------------|---|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | Yes | The number of the operation to be modified. | OP | |

Table 154. Keywords used in the PROMPTN Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-----------------------|-------------------------|---|--------------|------------|
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes that the default input arrival time is specified in the DEFIAT parameter of the EQQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence occurrences found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Examples:

PROMPTN APPL(TEST01)

Specifies that NO is the reply to a recovery prompt issued for any of the operations of application occurrence TEST01

PROMPTN APPL(TEST01) OP(10,30,50)

Specifies that NO is the reply to a recovery prompt issued for operations 10, 30, and 50 of application occurrence TEST01

PROMPTN APPL(TEST01) IAD(=)

Specifies that NO is the reply to a recovery prompt issued for any of the

operations of application occurrence TEST01 for the current input arrival date and the default input arrival time

PROMPTN APPL(TEST01) IAD(=) IAT(1800)

Specifies that NO is the reply to a recovery prompt issued for any of the operations of application occurrence TEST01 for the current input arrival date and input arrival time 18.00

PROMPTN APPL(TEST01) IAD(970708)

Specifies that NO is the reply to a recovery prompt issued for any of the operations of application occurrence TEST01 for the input arrival date 970708 and the default input arrival time

PROMPTN APPL(TEST01) IA(=) OP(30)

Specifies that that NO is the reply to a recovery prompt issued for operation 30 of application occurrence TEST01 for the current input arrival date and time

PROMPTN APPL(TEST01) IA(9707081801)

Specifies that NO is the reply to a recovery prompt issued for any of the operations of application occurrence TEST01 for the input arrival date 970708 and the input arrival time 18.01

The PROMPTN instruction returns one of the following return codes:

RESULT = 0

Operations status modified.

RESULT = 4

Occurrence or operations not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

PROMPTY

The PROMPTY instruction specifies that YES is the reply to a recovery prompt issued for an abended operation.

Table 155 describes the keywords that can be used.

Table 155. Keywords used in the PROMPTY Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-------------|-------------------------|---|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | Yes | The number of the operation to be modified. | OP | |
| | | | | | |

Table 155. Keywords used in the PROMPTY Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-----------------------|-------------------------|---|--------------|------------|
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes that the default input arrival time is specified in the DEFIAT parameter of the EQQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Examples:

PROMPTY APPL(TEST01)

Specifies that YES is the reply to a recovery prompt issued for any of the operations of application occurrence TEST01 that ended in error with return code different from FAIL.

PROMPTY APPL(TEST01) OP(10,30,50)

Specifies that YES is the reply to a recovery prompt issued for operations 10, 30, and 50 of application occurrence TEST01 if they ended in error with return code different from FAIL.

PROMPTY APPL(TEST01) IAD(=)

Specifies that YES is the reply to a recovery prompt issued for any operation of application occurrence TEST01, for the current input arrival date and the default input arrival time, that ended in error with return code different from FAIL.

PROMPTY APPL(TEST01) IAD(=) IAT(1800)

Specifies that YES is the reply to a recovery prompt issued for any operation of application occurrence TEST01, for the current input arrival date and input arrival time 18.00, that ended in error with return code different from FAIL.

PROMPTY APPL(TEST01) IAD(970708)

Specifies that YES is the reply to a recovery prompt issued for any operation of application occurrence TEST01, for the input arrival date 970708 and the default input arrival time, that ended in error with return code different from FAIL.

PROMPTY APPL(TEST01) IA(=) OP(30)

Specifies that YES is the reply to a recovery prompt issued for operation 30 of application occurrence TEST01, for the current input arrival date and time, if it ended in error with return code different from FAIL.

PROMPTY APPL(TEST01) IA(9707081801)

Specifies that YES is the reply to a recovery prompt issued for any operation of application occurrence TEST01, for the input arrival date 970708 and the input arrival time 18.01, that ended in error with return code different from FAIL.

The PROMPTY instruction returns one of the following return codes:

RESULT = 0

Operations status modified.

RESULT = 4

Occurrence or operations not found in the current plan.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

RELEASE

The RELEASE instruction releases an operation or all the operations of an occurrence that are in HOLD status in the current plan.

Table 156 describes the keywords that can be used.

Table 156. Keyword used in the Release Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-------------|-------------------------|---|--------------|--------|
| APPL() | Required | Yes | The name of the application to be modified. | | |

Table 156. Keyword used in the Release Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|---|--------------|------------|
| OPNO() | Optional | Yes | The number of the operation to be modified. If OPNO() is not specified, the RELEASE command, MR, will be issued to all the occurrence operations. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes that the default input arrival time is specified in the DEFIAT parameter of the EQQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrence occurrences found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |

Table 156. Keyword used in the Release Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-------------|-------------------------|-------------|--------------|--------|
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Examples:

RELEASE APPL(TEST01)

Releases from HOLD status all operations of application occurrence TEST01

RELEASE APPL(TEST01) OP(10,30,50)

Releases from HOLD status operations 10, 30, and 50 of application occurrence TEST01

RELEASE APPL(TEST01) IAD(=)

Releases from HOLD status all operations of application occurrence TEST01 for the current input arrival date and the default input arrival time

RELEASE APPL(TEST01) IAD(=) IAT(1800)

Releases from HOLD status all operations of application occurrence TEST01 for the current input arrival date and input arrival time 18.00

RELEASE APPL(TEST01) IAD(970708)

Releases from HOLD status all operations of application occurrence TEST01 for the input arrival date 970708 and the default input arrival time

RELEASE APPL(TEST01) IA(=) OP(30)

Releases from HOLD status operation 30 of application occurrence TEST01 for the current input arrival date and time

RELEASE APPL(TEST01) IA(9707081801)

Releases from HOLD status all operations of application occurrence TEST01 for the input arrival date 970708 and the input arrival time 18.01

The RELEASE instruction returns one of the following return codes:

RESULT = 0

Operations status modified.

RESULT = 4

Occurrence found but operations not found in the current plan.

RESULT = 8

Occurrence not found in the current plan. Invalid instruction or PIF problem. Refer to the error messages.

RELOP

The RELOP instruction releases an internal successors of an operation within an occurrence in the current plan.

Table 157 on page 252 describes the keywords that can be used.

Table 157. Keywords used in the Relop Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|------------|-----------------------|-------------------------|--|--------------|------------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | Yes | The number of the operation to be modified. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE=() means the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE=() is specified, IATIME() assumes the default input arrival time that is defined in the DEFIAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA=() assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| EXCLISUC() | Optional | Yes | The operation successor that must not be released. | EISUC | |

Note: ¹ Mutually exclusive with another keyword.

Note: Before deleting the dependency between the OPNO() and internal successor operation, the program links the successor operation to the default first operation within the application, which is specified in the DEOPNO variable.

Examples:

RELOP APPL(TEST01) OP(30)

All the internal successor dependencies will be deleted from operation 30 of the occurrence TEST01 with the earliest or latest input arrival date and time, depending on the SORT parameter.

RELOP OP(30) APPL(TEST01) EISUC(60)

All the internal successor dependencies other than operation 60 will be deleted from operation 30 of the occurrence TEST01 with the earliest or latest input arrival date and time, depending on the SORT parameter.

RELOP APPL(TEST01) OP(30) IAD(970708) IAT(1800)

All the internal successor dependencies will be deleted from operation 30 of occurrence TEST01 with the input arrival date 970708 and time 1800.

RELOP APPL(TEST01) EISUC(50,60) OP(30)

All the internal successor dependencies other than 50 and 60 will be deleted from operation 30 of occurrence TEST01 with the earliest or latest input arrival date and time depending on the SORT parameter.

The RELOP instruction returns one of the following return codes:

RESULT = 0

The internal successors have been released.

RESULT = 4

Internal successors not released.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

RELSUCC

The RELSUCC instruction releases the external successors of an occurrence in the current plan.

Table 158 describes the keywords that can be used.

Table 158. Keywords used in the Relsucc Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|---------|-------------|-------------------------|---|--------------|--------|
| APPL() | Required | No | The name of the application to be modified. | | |
| OPNO() | Optional | Yes | The number of the operation to be modified. OPNO() and EXCLOP() are mutually exclusive. | OP | |

Table 158. Keywords used in the Relsucc Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-----------------------|-------------------------|---|--------------|------------|
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE=() means the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME() | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes the default input arrival time that is defined in the DEFAT parameter of the EQQYRPRM member. IATIME() and IA() are mutually exclusive. | IAT | HHMM |
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter specified in PARMLIB or in the INIT instruction. IA() cannot be used with IADATE() or IATIME(). | | YYMMDDHHMM |
| EXCLSUC() | Optional | Yes | The application successor that must not be released. | ESUC | |
| EXCLOP() | Optional ¹ | Yes | The operation that must not be modified. EXCLOP() and OPNO() are mutually exclusive. | EOP | |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Examples:

RELSUCC APPL(TEST01)

All the successor dependencies will be deleted from occurrence TEST01 with the earliest or latest input arrival date and time, depending on the SORT parameter

RELSUCC APPL(TEST01) IAD(970708) IAT(1800)

All the successor dependencies will be deleted from occurrence TEST01 with the input arrival date 970708 and time 1800

RELSUCC APPL(TEST01) IAD(&OYMD1) EOP(30,35)

All the successor dependencies will be deleted from all the operations, other than 30 and 35, of occurrence TEST01 with the occurrence input arrival date and with the input arrival time corresponding to the default IA time, DEFIAT

**RELSUCC APPL(TEST01) IAD(970708) OP(40,50,60,70,,80)
ESUC(TEST03,TEST04)**

All the successor dependencies other than TEST03 and TEST04 will be deleted from operations 40, 50, 60, 70, and 80 of the occurrence TEST01 with the input arrival date 970708 and with the default input arrival time, DEFIAT

The RELSUCC instruction returns one of the following return codes:

RESULT = 0

The successors have been released.

RESULT = 4

Successors not released.

RESULT = 8

Invalid instruction or PIF problem. Refer to the error messages.

SET

SET assigns a value to a variable, which can be used in the OCL program.

Syntax

SET variable = expression

The expression can contain arithmetic and concatenation operators and built-in functions.

Concatenation operators

The concatenation operators combine two strings to form one string by appending the second string to the right-hand end of the first string. The concatenation might occur with or without an intervening blank. The concatenation operators are:

(blank)

Concatenate terms with one blank in between.

||

Concatenate without an intervening blank.

Built-In Functions

The expression can also contain REXX built-in functions: SUBSTR, LEFT, RIGHT, and OVERLAY. Here are some examples:

```

SET VAR1 = VAR1 !! VAR2 !! 'XX'
 (!! is the concatenation character)
SET VAR2 = SUBSTR(&OYMD1,5,2) + 1
SET VAR3 = RIGHT(VAR1,2,'0')
SET VAR4 = LEFT(VAR4,3)
SET VAR5 = OVERLAY('X',VAR4,5)

```

Syntax of SUBSTR built-in function

SUBSTR(string,n,length,pad)

The SUBSTR built-in function returns the substring of string that begins at the *n*th character and is of length *length*, padded if necessary with the character *pad*. *n* must be a positive whole number. If *n* is greater than LENGTH(*string*), only padding characters are returned. If you omit *length*, the rest of the string is returned. The default padding character is a blank. Here are some examples:

```

SET VAR1 = SUBSTR('abc',2)          ->  VAR1 = 'bc'
SET VAR1 = SUBSTR('abc',2,4)        ->  VAR1 = 'bc '
SET VAR1 = SUBSTR('abc',2,6,'.')    ->  VAR1 = 'bc....'

```

Syntax of RIGHT built-in function

RIGHT(string,length,pad)

The RIGHT built-in function returns a string of length *length*, containing the rightmost *length* characters of *string*. The string returned is padded with *pad* characters, or truncated, on the left, as necessary. The default padding character is a blank. *length* must be a positive whole number or zero.

Here are some examples:

```

SET VAR1 = RIGHT('abc d',8)         ->  VAR1 = ' abc d'
SET VAR1 = RIGHT('abc def',5)       ->  VAR1 = 'c def'
SET VAR1 = RIGHT('12',5,'0')        ->  VAR1 = '00012'

```

Syntax of LEFT built-in function

LEFT(string,length,pad)

The LEFT built-in function returns a string of length *length*, containing the leftmost *length* characters of *string*. The string returned is padded with *pad* characters, or truncated, on the right as necessary. The default padding character is a blank. *length* must be a positive whole number or zero. The LEFT function is exactly equivalent to SUBSTR(*string*,1,*length*,*pad*)

Here are some examples:

```

SET VAR1 = LEFT('abc d',8)           ->  VAR1 = 'abc d '
SET VAR1 = LEFT('abc d',8,'.')       ->  VAR1 = 'abc d...'
SET VAR1 = LEFT('abc def',7)         ->  VAR1 = 'abc de'

```

Syntax of OVERLAY built-in function

OVERLAY(new,target,n,length,pad)

The OVERLAY built-in function returns the string *target*, which, starting at the *n*th character, is overlaid with the string *new*, padded or truncated to length *length*. (The overlay might extend beyond the end of the original target string.) If you specify *length*, it must be a positive whole number or zero. The default value for

length is the length of *new*. If *n* is greater than the length of the target string, padding is added to the left of the *new* string. The default padding character is a blank, and the default value for *n* is 1. If you specify *n*, it must be a positive whole number.

Here are some examples. Assume that VAR1 value is 'ABCDEFGH'.

```
SET VAR1 = OVERLAY(' ',VAR1,3)      -> VAR1 = 'AB DEF GH'  
SET VAR1 = OVERLAY('. ',VAR1,3,2)   -> VAR1 = 'AB. EFGH'  
SET VAR1 = OVERLAY('qq',VAR1)      -> VAR1 = 'qqCDEFGH'  
SET VAR1 = OVERLAY('qq',VAR1,4)     -> VAR1 = 'ABCqqFGH'  
SET VAR1 = OVERLAY('123',VAR1,5,6,'+') -> VAR1 = 'ABCD123+++'
```

The SET instruction returns one of the following return codes:

CC = 0

Instruction correctly processed

CC = 8

Invalid instruction. See the error message.

SETUPD

Syntax

SETUPD expression

The SETUPD instruction sets the value of a user variable and updates its default value in a variable table. The new default value can be used jobs in an application occurrence that use that variable.

The variable table must be specified by the INIT instruction.

SETUPD is equivalent to both the SET and UPD instructions.

The SETUPD instruction uses the EQQPIFT program, which is in the EQQPFIJFV member of the scheduler sample library. You need to specify the CARDIN DD card in the OCL procedure EQQYRPRC.

Examples:

```
SETUPD VAR1 = '00000'  
SETUPD VAR1 = SUBSTR(&OYMD1,5,2) + 1
```

According to the previous example, enclose within single quotes any input string or character when specified as value of a user variable.

The SETUPD instruction returns one of the following return codes:

CC = 0

Instruction correctly processed

CC = 4

Variable does not exist in the variable table; it is added

CC = 8

Invalid instruction. See the error messages.

SRSTAT

The SRSTAT instruction invokes the TSO command SRSTAT, which you can use to change the overriding (global) availability, quantity, and deviation of a special resource. You can use it to prevent operations from allocating a particular resource, or to request the ETT function to add an occurrence to the current plan.

Table 159 shows the keywords that can be specified.

Table 159. Keywords used in the Srstat Instruction

| Keyword | Requirement | Description | Abbreviation |
|-----------------|-------------|--|--------------|
| 'resource name' | Required | The name of the resource whose availability you want to change. This parameter must be contained within single quotation marks and can be up to 44 characters in length. | |
| AVAIL() | Optional | It can be one of the following: YES Indicates that the availability status of the resource is to be set to YES. NO Indicates that the availability status of the resource is to be set to NO. RESET Sets the overriding availability to blank, so that the interval or default value is used. KEEP The default, does not change the availability status. | |

Table 159. Keywords used in the Srstat Instruction (continued)

| Keyword | Requirement | Description | Abbreviation |
|-------------|-------------|--|--------------|
| CREATE() | Optional | <p>It can be one of the following:</p> <p>NO If NO is specified, the resource is not added to the current plan of the receiving subsystem if it does not exist in the database. CREATE(NO) does not have any effect. If the resource does not exist in the database, a new resource is created.</p> <p>YES If YES is specified or defaulted, and the DYNAMICADD keyword of the RESOPTS initialization statement is set to YES or EVENT, IBM Workload Scheduler for z/OS adds the resource to the current plan of the receiving IBM Workload Scheduler for z/OS subsystem if the resource is not in the database. It uses these values:</p> <p>Text Blank.</p> <p>Specres group ID Blank.</p> <p>Hiperbatch No.</p> <p>Used for Control.</p> <p>On error Blank. If an error occurs, IBM Workload Scheduler for z/OS uses the value specified in the operation details or, if that is also blank, the value of the ONERROR keyword of RESOPTS.</p> <p>Overriding availability, quantity, and deviation The value specified by SRSTAT, or blank.</p> <p>Default quantity 1. The default quantity is automatically increased if contention occurs.</p> <p>Default availability Yes.</p> <p>Intervals No intervals are created.</p> <p>Workstations * (All workstations can allocate the resource).</p> | CRE |
| DEVIATION() | Optional | <p>It can be one of the following:</p> <p><i>amount</i> If you want to change the deviation, specify a deviation, which is an amount to be added to (positive number) or subtracted from (negative number) the current quantity. A specified amount can be in the range of -999999 through +999999.</p> <p>KEEP The default, does not alter the deviation.</p> <p>RESET Sets the deviation to zero.</p> | DEV |

Table 159. Keywords used in the Srstat Instruction (continued)

| Keyword | Requirement | Description | Abbreviation |
|------------|-------------|---|--------------|
| LIFESPAN() | Optional | <p>(<i>interval</i>, <i>new_availability_value</i>)</p> <p><i>interval</i> The interval of time, in minutes. It can be from 0 to 99999.</p> <p><i>new_availability_values</i> The value to which the global availability is changed after interval expiration. It can be YES, NO, or RESET</p> | |
| QUANTITY() | Optional | <p>It can be one of the following:</p> <p><i>amount</i> If you want to change the overriding (global) quantity, specify the amount in the range 1–999999.</p> <p>RESET Sets the overriding quantity to blank. so that the interval or default value is used.</p> <p>KEEP The default, does not alter the quantity.</p> | Q |
| SUBSYS() | Optional | The name of the tracker subsystem to which the SRSTAT instruction is directed. It defaults to the subsystem name specified in the OCL OPTRK initialization parameter. | |
| TRACE() | Optional | Event tracing indicator. When a positive number is specified, a trace entry is created for each event generated by the SRSTAT command. The trace record is written to the message log file identified by ddname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 will not generate trace records. | |
| | | | |

Example:

```
SRSTAT 'IMS.DATA.BASE' SUBSYS(OPCB) AVAIL(YES)
```

In this example the availability status of the resource IMS™.DATA.BASE is changed to YES.

The SRSTAT instruction returns one of the following return codes:

RESULT = 0

Special resource availability modified

RESULT = 8

Invalid instruction

UNNOP

The UNNOP instruction restores an operation from the NOP status. This instruction is not supported in operations running on fault-tolerant workstations.

Table 160 describes the keywords that can be used.

Table 160. Keywords used in the Unnop Instruction

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|----------|-----------------------|-------------------------|---|--------------|--------|
| APPL() | Required | Yes | The name of the application to be modified. | | |
| OPNO() | Optional | Yes | The number of the operation to be modified. If OPNO() is not specified, the UNNOP command, UN, will be issued to all the occurrence operations. | OP | |
| IADATE() | Optional ¹ | No | The input arrival date of the application occurrence. IADATE(=) assumes the current date. IADATE() and IA() are mutually exclusive. | IAD | YYMMDD |
| IATIME | Optional ¹ | No | The input arrival time of the application occurrence. If IADATE(=) is specified, IATIME() assumes that the default input arrival time is specified in the DEFIAT parameter of the EQYRPRM member. IADATE() and IA() are mutually exclusive. | IAT | HHMM |

Table 160. Keywords used in the UNNOP Instruction (continued)

| Keyword | Requirement | Multiple Values Allowed | Description | Abbreviation | Format |
|--|-----------------------|-------------------------|--|--------------|------------|
| IA() | Optional ¹ | No | The input arrival date and time of the application occurrence. IA(=) assumes the current date and time. It defaults to the earliest or latest input arrival date and time of the application occurrences found in the current plan, depending on the SORT parameter. IA() cannot be used together with IADATE() or IATIME(). | | YYMMDDHHMM |
| Note: ¹ Mutually exclusive with another keyword. | | | | | |

Example control statements:

UNNOP APPL(TEST01)

Restores all the operations of occurrence TEST01 from the NOP status

UNNOP APPL(TEST01) OP(10,30,50)

Restore operations 10, 30, and 50 of occurrence TEST01 from the NOP status

The UNNOP instruction returns one of the following return codes:

RESULT = 0

Operations status modified.

RESULT = 4

Occurrence found but operations not found in the current plan.

RESULT = 8

Occurrence not found in the current plan. Invalid instruction or PIF problem. Refer to the error messages.

UPD

The UPD instruction changes the default value of a user variable in a variable table of the scheduler. The new default value can be used by jobs in an application occurrence that use that variable. The variable table must be identified by the INIT instruction.

Syntax

UPD *variable*

The UPD instruction uses the EQQPIFT program, which is supplied with IBM Workload Scheduler for z/OS and is in the EQQPFIJV member of the scheduler sample library. You need to specify the CARDIN DD card in the OCL procedure EQQYRPRC.

If the variable exists in the variable table, the UPD instruction changes the default value; otherwise it adds the variable and the variable's default value to the variable table.

Example:

```
UPD VAR2
```

Enclose within single quotes any input string or character when specified as value of a user variable.

The UPD instruction returns one of the following return codes:

CC = 0

Instruction correctly processed

CC = 4

Variable did not exist in the variable table; it has been added

CC = 8

Invalid instruction. See the error message.

WSSTAT

The WSSTAT instruction invokes the TSO command WSSTAT, which you can use to change the status of a workstation in the current plan. You can also establish or close the connection of a workstation to the network. The status information is communicated to the controller to indicate a workstation as active, off-line, or failed. When you use the WSSTAT instruction to report a workstation status of offline or failed, you can optionally define restart and routing options for the workload defined on the workstation. You can also change the domain manager of a workstation.

Table 161 shows the keywords that can be used.

Table 161. Keywords used in the Wsstat Instruction

| Keyword | Requirement | Description |
|-----------|-------------|---|
| ALTWS() | Optional | The alternate workstation name. When the workstation status is set to offline or failed, you can specify the alternate workstation where reroutable operations are to be started. |
| CMD() | Optional | When you want to change the workstation status, you can specify one of the following for CMD(): L Link P Stop S Start U Unlink |
| MANAGES() | Optional | When you want to modify the domain manager of a workstation, you can specify the new domain manager with MANAGES(). |

Table 161. Keywords used in the Wsstat Instruction (continued)

| Keyword | Requirement | Description |
|------------|-------------|--|
| REROUTE() | Optional | When the workstation status is set to offline or failed, you can specify one of the following for REROUTE(): R To reroute operations to the alternate workstation L To leave the operations at the inactive workstation |
| STARTOPR() | Optional | When the workstation status is set to offline or failed, you can specify what the scheduler is to do with operations that are currently in started status at the workstation: R Restart operations automatically on the alternate workstation. E Set all started operations to ended-in-error status L Leave the operations in started status. |
| STATUS() | Required | The status you want to report for the workstation: A Active O Offline F Failed |
| SUBSYS() | Optional | The name of the tracker subsystem to which the WSSTAT instruction is directed. It defaults to the subsystem name specified in the OCL OPTRK initialization parameter. In the end-to-end with fault tolerance capabilities environment, when you set the CMD keyword, you must specify the controller name. |
| TRACE() | Optional | Event tracing indicator. When a nonzero positive number is specified, a trace entry is created for each event generated by the WSSTAT command. The trace record is written to the message log file identified by dname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 will not generate trace records. |
| WSNAME() | Required | The name of the workstation to be updated. |

Examples:

```
WSSTAT WSNAME(AS4H) STATUS(O) START(R)
```

In this example, the status of workstation AS4H is set to offline. Started operations will be restarted on the alternate workstation.

```
WSSTAT WSNAME(AS4H) SUBSYS(OPCC) MANAGES(DOMAIN1)
```

In this example, the workstation AS4H changes its domain manager to DOMAIN1

The WSSTAT instruction returns one of the following return codes:

RESULT = 0

Workstation status modified

RESULT = 8

Invalid instruction

Note: If you try to set the status of a fault-tolerant workstation to Failed, the WSSTAT instruction returns a RESULT=0, the message EQQE112E is issued in the controller Mlog and the workstation status is left as it was before. This happens because that status is not allowed for a fault-tolerant workstation.

WTO

The WTO instruction displays messages on the system console and waits for a reply.

Example:

```
WTO Reply YES, SI, OK or press enter to continue
```

Note: OCL uses the IPOWTO program, which is provided as a sample program with IBM Workload Scheduler for z/OS. The message to be displayed by this program is written in member WTOIN in the EQQMISC data set. You therefore need not specify the WTOIN DD card in the OCL procedure.

The WTO instruction returns one of the following return codes:

CC = 0

Valid reply

CC = 8

Reply is not YES, SI, OK, or enter

Requirements

The requirements for OCL are as follows:

- IBM Workload Scheduler for z/OS
- IBM Library for SAA REXX/370 Version 1 Release 3
- IPOWTO program provided in the EQQOCWTO member of the IBM Workload Scheduler for z/OS sample library.
- EQQPIFT program provided in the EQQPFIJV member of the scheduler sample library

Note: All libraries referred to in the STEPLIB DD card in the OCL procedure must be APF-authorized.

Sample job and procedure

This section provides an example of a job, EQQYRJCL, and an example of a procedure, EQQYRPRC.

EQQYRJCL sample job

```
//A JOB CARD ACCORDING TO YOUR INSTALLATION STANDARDS IS REQUIRED
//*
//* THIS JOB RUNS THE EQQOCL PROGRAM.
//*
//* THERE IS ONE STEP IN THIS JOB:
//*
//* EQQOCL:    INVOKES THE EQQOCL REXX COMPILED PROGRAM
//*
//* IN ORDER TO USE THIS JOB SUCCESSFULLY YOU SHOULD MODIFY IT
//* AS FOLLOWS:
//* 1. REPLACE THE JOBCARD WITH A VALID JOBCARD FOR YOUR
//*    INSTALLATION
//* 2. LOCATE DATA SET NAMES BEGINNING WITH OPCA.INST AND REPLACE
//*    WITH DATA SET NAMES VALID FOR YOUR INSTALLATION.
//* 3. WRITE YOUR OCL INSTRUCTIONS IN THE SYSIN CARD
//*
//* MOREOVER YOU SHOULD CONSIDER THE FOLLOWING INSTRUCTIONS:
//* 4. UPDATE THE EQQYRPRC PROCEDURE IN THE SCHEDULER SAMPLE LIBRARY
//*    ACCORDING TO THE INSTRUCTIONS GIVEN IN ITS PROLOG
```

```

/** 5. THE SAMPLE PROGRAM EQQRXSTG DELIVERED WITH THE SCHEDULER MUST BE
/** AVAILABLE
/** 6. IF THE WTO INSTRUCTION IS USED, THE IPOWTO SAMPLE PROGRAM
/** DELIVERED WITH IBM Workload Scheduler for z/OS MUST BE AVAILABLE
/** 7. IF THE UPD INSTRUCTION IS USED, THE EQPIFT SAMPLE PROGRAM
/** DELIVERED WITH THE SCHEDULER IN THE EQPIFJV MEMBER OF SAMPLE LIBRARY
/** MUST BE AVAILABLE
/**
/** NOTE THAT THIS JOB ASSUMES THAT THE SCHEDULER HAS BEEN INSTALLED
/** AND THAT THE SMP ACCEPT FUNCTION HAS BEEN PERFORMED.
/**
/**/MYJCLLIB JCLLIB ORDER=OPCA.INST.SEQQSAMP
/**
/**EQOCL EXEC EQYRPRC
/**SYSPRINT DD SYSOUT=*,DCB=(RECFM=FB,LRECL=133,BLKSIZ=1330)
/**SYSTSPRT DD SYSOUT=*
/**EQOCL.SYSIN DD *
* SPECIFY YOUR OCL INSTRUCTIONS
* ...

```

EQYRPRC sample procedure

```

/**EQYRPRC PROC
/*******
/**
/** THIS PROCEDURE IS USED BY EQYRJCL SAMPLE THAT RUNS THE SCHEDULER
/** CONTROL LANGUAGE
/**
/** IN ORDER TO USE THIS JOB SUCCESSFULLY YOU SHOULD MODIFY IT
/** AS FOLLOWS:
/** 1. LOCATE DATA SET NAMES BEGINNING WITH OPCA.INST AND REPLACE WITH
/** DATA SET NAMES VALID FOR YOUR INSTALLATION.
/**
/** MOREOVER YOU SHOULD CONSIDER THE FOLLOWING INSTRUCTIONS:
/** 3. MAKE SURE THAT THE REXX/370 RUNTIME LIBRARIES ARE AVAILABLE TO
/** TSO/E (IRXCPTM TABLE IS DEFINED)
/** 4. THE SAMPLE PROGRAM EQQRXSTG DELIVERED WITH THE SCHEDULER MUST BE
/** AVAILABLE
/** 5. IF THE WTO INSTRUCTION IS USED, MAKE SURE THAT THE
/** SAMPLE PROGRAM EQQCWTO DELIVERED WITH IBM WORKLOAD SCHEDULER
/** FOR Z/OS IS AVAILABLE
/** 6. IF THE UPD INSTRUCTION IS USED, MAKE SURE THAT THE PL/I RUNTIME
/** LIBRARIES ARE AVAILABLE, AND THAT THE EQPIFT SAMPLE PROGRAM
/** (CONTAINED IN EQPIFJV SAMPLE) IS AVAILABLE.
/**
/** REQUIRED DATA SETS:
/**
/** STEPLIB = THE SCHEDULER LOAD LIBRARY
/** SYSEXEC = OCL MODULE
/** OCLPARM = OCL PARAMETER LIBRARY
/** OCLMLIB = OCL MESSAGE LIBRARY
/** OCLLOG = OCL LOG FILE (IF NEEDED); IT MUST BE LRECL 133 AND
/** ALLOCATED WITH DISP=MOD
/** EQMLIB = THE SCHEDULER MESSAGE LIBRARY
/** EQMLOG = THE SCHEDULER MESSAGE LOG
/** EQYPARM = THE SCHEDULER PARAMETERS FOR PIF REQUESTS
/**
/*******
/**
/**EQOCL EXEC PGM=IKJEFT01,PARM='EQOCL'
/**STEPLIB DD DISP=SHR,DSN=OPCA.INST.SEQQLMDO <== CHANGE
/** DD DISP=SHR,DSN=PLI_RUNTIME_LIBRARY <== CHANGE
/**OCLLOG DD DISP=MOD,DSN=OCL_LOG_FILE <== CHANGE
/**OCLPARM DD DISP=SHR,DSN=OPCA.INST.SEQQSAMP(EQYRPRM) <== CHANGE
/**OCLMLIB DD DISP=SHR,DSN=OPCA.INST.SEQQSAMP(EQYRMSG) <== CHANGE
/**SYSEXEC DD DISP=SHR,DSN=OPCA.INST.SEQQMISC <== CHANGE
/**SYSPRINT DD SYSOUT=*,DCB=(RECFM=FB,LRECL=133,BLKSIZ=1330)

```

```

//CARDIN DD UNIT=SYSDA,SPACE=(TRK,(20,200)),          <== CHANGE
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD DUMMY
//EQQLIB DD DISP=SHR,DSN=OPCA.INST.SEQMSG0           <== CHANGE
//EQQPARM DD DISP=SHR,DSN=OPCA.INST.PARM(INIT)       <== CHANGE
//EQQLOG DD SYSOUT=*
//EQQDUMP DD SYSOUT=*

```

Messages

Message format

OCL messages have the format: *EQQCLxxCtext* where:

EQQCL

The message prefix.

xx The message identifier, which is an alphanumeric value in the range 0–9, A–Z: 00,01,02–09, 0A, 0B–0Y, 10, 11–19, 1A...

C The message severity code, which can have one of the following values:

I Information message. Processing continues and, in almost all cases, no action is required of the user.

W Warning message. Processing continues and, in most cases, no action is required of the user.

E Error message. Processing terminates; action is required of the user.

text The message text, which can vary in length. The text can contain message variables that are substituted at run time or when the message is issued. In this book, message variables are shown in italic text.

The OCL messages are written to the OCL output data set, SYSTSPRT, and in the OCL message log data set, if it is allocated in the OCL procedure. The message log data set name is specified in the OCLLOG DD card. The format of an OCL message printed in the message log data set is:

```
mm/dd HH.MM.SS message
```

where:

mm/dd The current month (*mm*) and day (*dd*).

HH.MM.SS

The current time of day:

HH The number of hours, starting from midnight. It is a number in the range 00–23.

MM The number of minutes in the current hour. It is a number in the range 00–59.

SS The number of seconds in the current minute. It is a number in the range 00–59.

message

The message identifier and text (see the description of OCL messages earlier in this section).

Here is an example of messages in the OCL message log data set:

```

07/30 13:41:33 EQQCL00I Instruction : INIT VARTAB(&OADID)
07/30 13:41:34 EQQCL02I INIT instruction executed : RC=0
07/30 13:41:34 EQQCL00I Instruction : SETUPD VAR1 = 'PIPP0'
07/30 13:41:34 EQQCL02I SETUPD instruction executed : RC=0
07/30 13:41:34 EQQCL00I Instruction : COMPL APPL(TEST01)
07/30 13:41:35 EQQCL02E COMPL instruction executed : RC=8
07/30 13:41:35 EQQCL00I Instruction : CHKAPPL APPL(TEST02)
07/30 13:41:35 EQQCL02W CHKAPPL instruction executed : RC=4
07/30 13:41:35 EQQCL00I Instruction : IF RESULT = 8 THEN NOP

```

Here is an example of the messages written in the SYSTSPRT DD card:

```

EQQCL01I =====
EQQCL00I Processing: CHKAPPL APPL(TEST01) STATUS(S)
EQQCL0JI Searching for the occurrence TEST01 in CP
EQQCL00I Occurrence found: APPL(TEST01) IA(9709080800) STATUS(S)
EQQCL00I Occurrence found: APPL(TEST01) IA(9709080930) STATUS(S)
EQQCL0KI Total n. of matching occurrences: 2
EQQCL01I =====
EQQCL00I Processing: IF RESULT=0 THEN ADD APPL(TEST02)
EQQCL03I True condition: IF RESULT = 0
EQQCL01I =====
EQQCL00I Processing: ADD APPL(TEST02)
EQQCL0AI The occurrence was successfully added: APPL(TEST02) IA(9709091657)

```

For explanations of all OCL messages, refer to “EQQCLnnn Messages” in *Messages and Codes*.

Part 3. Appendixes

Appendix A. Program interface record format

This appendix describes the fields of the data records handled by the program interface communication routine, EQQYCOM.

These formats are used when information is retrieved by EQQYCOM and provided to the user-written program, and when information is provided by the user program to EQQYCOM to be written to IBM Workload Scheduler for z/OS databases or data sets.

Note: For a correct interpretation of the fields described as "Tod clock at last update", see "TOD fields."

TOD fields

All field in TOD format contain the time-of-day clock value and are set automatically from the system when a replace or insert request is issued. This data is represented in a binary counter corresponding to a 64-bits unsigned integer and its value is implemented every 2-12 microseconds (clock unit) starting from 1st January 1900 with the cycle of the clock of approximately 143 years. In order to understand better the content of this field, please refer to the two tables below:

Table 162. Clock value setting at the start of different years

| YEAR | CLOCK SETTING (HEX NOTATION) | | | |
|------|------------------------------|------|------|------|
| 1900 | 0000 | 0000 | 0000 | 0000 |
| 1976 | 8853 | BAF0 | B400 | 0000 |
| 1980 | 8F80 | 9FD3 | 2200 | 0000 |
| 1984 | 96AD | 84B5 | 9000 | 0000 |
| 1988 | 9DDA | 6997 | FE00 | 0000 |
| 1992 | A507 | 4E7A | 6C00 | 0000 |
| 1996 | AC34 | 335C | DA00 | 0000 |
| 2000 | B361 | 183F | 4800 | 0000 |

Table 163. Clock value setting at different time interval

| INTERVAL | CLOCK UNIT (HEX ROTATION) | | | |
|----------------|---------------------------|------|------|------|
| 1 microsec. | | | | 1000 |
| 1 millisec. | | | 3E | 8000 |
| 1 second | | | F424 | 0000 |
| 1 minute | | 39 | 3870 | 0000 |
| 1 hour | | D69 | 3A40 | 0000 |
| 1 day | 1 | 41DD | 7600 | 0000 |
| 365 days | 1CA | E8C1 | 3E00 | 0000 |
| 366 days | 1CC | 2A9E | B400 | 0000 |
| 1.461 days (*) | 72C | E4E2 | 6E00 | 1000 |

Application description (resource codes AD, ADCOM)

An application description record can contain these segments:

ADCOM

Common segment. Only one common segment must appear as the first segment in each record.

ADAPD

Application dependency segment.

ADCIV

Interval definition for conditional external predecessor segment.

ADDEP

Dependency segment.

ADCNC

Condition segment.

ADCNS

Condition dependency segment.

ADEXT

Extended name segment

ADOP

Operation segment.

ADRE Remote job information segment

ADRUN

Run cycle segment.

ADSAI

Operation system automation information segment.

ADSR Special resource segment.

ADUSF

User field segment.

ADVDD

Variable duration and deadline.

ADXIV

Interval definition for external predecessor.

Note: For a correct interpretation of the fields described as "TOD clock at last update", see "TOD fields" on page 271.

ADAPD - Application dependency segment

The application dependency part of an Application Description.

Table 164. ADAPD Control Block

| Offsets | | | | | |
|---------|-----|-----------|-----|-----------|--------------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | | ADAPD | APPLICATION DEPENDENCY SECTION OF AD |
| 0 | (0) | CHARACTER | 16 | ADAPDADID | APPLICATION PREDECESSOR <blank> |

Table 164. ADAPD Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|-------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 16 | (10) | CHARACTER | 4 | ADAPDWSID | WORKSTATION NAME |
| 20 | (14) | SIGNED | 4 | ADAPDOPNO | OPERATION NUMBER |
| 24 | (18) | CHARACTER | 4 | * | FREE |
| 28 | (1C) | CHARACTER | 50 | ADAPDDESC | DESCRIPTION |
| 78 | (4E) | CHARACTER | 1 | ADAPDLTP | LTP REPORT PRINT OPTION A C |
| 79 | (4F) | CHARACTER | 1 | ADAPDVERS | RECORD VERSION NUMBER = 1 |
| 80 | (50) | UNSIGNED | 1 | ADAPDFLAG | FLAGS |
| 81 | (51) | CHARACTER | 1 | ADAPDCSEL | RESOLUTION CRITERIA C S R A |
| 82 | (52) | CHARACTER | 1 | * | FREE |
| 83 | (53) | CHARACTER | 1 | ADAPDIVTYPE | INTERVAL TYPE R A (RELATIVE ABSOLUTE) |
| 84 | (54) | CHARACTER | 1 | ADAPDIVFWHE | FROM WHEN B A (BEFORE AFTER) |
| 85 | (55) | CHARACTER | 3 | ADAPDIVFHHH | FROM HOURS HHH (ONLY RELATIVE INTERVAL) |
| 88 | (56) | CHARACTER | 2 | ADAPDIVFHH | FROM HOURS HHH (ONLY ABSOLUTE INTERVAL) |
| 90 | (5A) | CHARACTER | 2 | ADAPDIVFMM | FROM MINUTES MM |
| 92 | (5C) | CHARACTER | 1 | ADAPDIVFD | FROM DAYS (ONLY ABSOLUTE INTERVAL) |
| 93 | (5D) | CHARACTER | 1 | ADAPDIVTWHE | TO WHEN B A (BEFORE AFTER) |
| 94 | (5E) | CHARACTER | 3 | ADAPDIVTHHH | TO HOURS HHH (ONLY RELATIVE INTERVAL) |
| 97 | (61) | CHARACTER | 2 | ADAPDIVTHH | TO HOURS HHH (ONLY ABSOLUTE INTERVAL) |
| 99 | (63) | CHARACTER | 2 | ADAPDIVTMM | TO MINUTES MM |
| 101 | (65) | CHARACTER | 1 | ADAPDIVTD | TO DAYS (ONLY ABSOLUTE INTERVAL) |
| 102 | (66) | CHARACTER | 2 | * | FREE |

ADCIV - Interval definition for conditional external predecessor segment

The interval definition for a conditional external predecessor. Used when ADCNS ADCNSCCSEL has value R or A (only one ADCIV per ADCNS can be used, but the same ADCIV can be used by more ADCNS segments if they refer to the same external predecessor application and operation).

Table 165. ADCIV Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 47 | ADCIV | CONDITIONAL EXTERNAL PREDECESSOR INTERVAL |
| 0 | (0) | CHARACTER | 16 | ADCIVADID | PREDECESSOR APPLICATION NAME |
| 16 | (10) | SIGNED | 4 | ADCIVCID | PREDECESSOR CONDITION ID |
| 20 | (14) | SIGNED | 4 | ADCIVOPNO | PREDECESSOR OPERATION NUMBER |
| 24 | (18) | SIGNED | 4 | ADCIVOWNOP | OWNING OPERATION |
| 28 | (1C) | CHARACTER | 1 | ADCIVTYPE | INTERVAL TYPE R/A (RELATIVE/ABSOLUTE) |
| 29 | (1D) | CHARACTER | 1 | ADCIVFWHE | FROM WHEN B/A (BEFORE/AFTER) |
| 30 | (1E) | CHARACTER | 3 | ADCIVFHHH | FROM HOURS HHH (ONLY RELATIVE INTERVAL) |
| 33 | (21) | CHARACTER | 2 | ADCIVFHH | FROM HOURS HH (ONLY ABSOLUTE INTERVAL) |
| 35 | (23) | CHARACTER | 2 | ADCIVFMM | FROM MINUTES MM |
| 37 | (25) | CHARACTER | 1 | ADCIVFD | FROM DAYS (ONLY ABSOLUTE INTERVAL) |
| 38 | (26) | CHARACTER | 1 | ADCIVTWHE | TO WHEN B/A (BEFORE/AFTER) |
| 39 | (27) | CHARACTER | 3 | ADCIVTHHH | TO HOURS HHH (ONLY RELATIVE INTERVAL) |
| 42 | (2A) | CHARACTER | 2 | ADCIVTHH | TO HOURS HH (ONLY ABSOLUTE INTERVAL) |
| 44 | (2C) | CHARACTER | 2 | ADCIVTMM | TO MINUTES MM |
| 46 | (2E) | CHARACTER | 1 | ADCIVTD | TO DAYS (ONLY ABSOLUTE INTERVAL) |

ADCOM - Common segment

The common part of an application description.

The reserved fields marked by an * in the name column should be treated as record data. Their value should be preserved when a record is updated and set to zero when a new segment is created.

Table 166. ADCOM Control Block

| Offsets | | | | | |
|---------|-----|-----------|-----|-------|----------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 192 | ADCOM | COMMON SECTION OF AD |
| 0 | (0) | CHARACTER | 23 | ADKEY | KEY |
| 0 | (0) | CHARACTER | 16 | ADID | APPLICATION ID |

Table 166. ADCOM Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|---|
| Dec | Hex | Type | Len | Name | Description |
| 16 | (10) | CHARACTER | 1 | ADSTAT | APPLICATION STATUS A = ACTIVE, P = PENDING |
| 17 | (11) | CHARACTER | 6 | ADTO | VALID-TO DATE |
| 23 | (17) | CHARACTER | 1 | * | RESERVED |
| 24 | (18) | CHARACTER | 1 | ADTYPE | APPLICATION TYPE A = APPLICATION, G = GROUP DEF. |
| 25 | (19) | CHARACTER | 1 | ADMONITOR | MONITOR AD |
| 26 | (1A) | CHARACTER | 6 | ADFROM | VALID-FROM DATE |
| 32 | (20) | CHARACTER | 24 | ADDESC | DESCRIPTIVE TEXT |
| 56 | (38) | CHARACTER | 8 | ADGROUP | AUTHORITY GROUP NAME |
| 64 | (40) | CHARACTER | 16 | ADOWNER | OWNER ID |
| 80 | (50) | CHARACTER | 24 | ADODESC | OWNER DESCRIPTION |
| 104 | (68) | SIGNED | 4 | ADPRIOR | PRIORITY |
| 108 | (6C) | CHARACTER | 16 | ADCAL | CALENDAR |
| 124 | (7C) | CHARACTER | 6 | ADLDATE | DATE LAST UPDATED |
| 130 | (82) | CHARACTER | 4 | ADLTIME | TIME LAST UPDATED |
| 134 | (86) | CHARACTER | 8 | ADLUSER | USERID OF LAST UPDATER |
| 142 | (8E) | UNSIGNED | 1 | ADCOMVERS | RECORD VERSION NUMBER |
| 143 | (8F) | CHARACTER | 16 | ADGROUPID | GROUP DEFINITION ID |
| 159 | (9F) | CHARACTER | 1 | * | RESERVED |
| 160 | (A0) | CHARACTER | 8 | ADLUTS | TOD CLOCK AT LAST UPDATE |
| 168 | (A8) | SIGNED | 4 | ADDSM | DEADLINE SMOOTHING FACTOR |
| 172 | (AC) | SIGNED | 4 | ADDLIM | DEADLINE FEEDBACK LIMIT |
| 176 | (B0) | CHARACTER | 16 | * | RESERVED |

ADDEP - Dependency segment

The dependency part of an application description.

Table 167. ADDEP Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|--------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 96 | ADDEP | DEPENDENCY SECTION OF AD |
| 0 | (0) | CHARACTER | 16 | ADDEPADID | EXTERNAL PREDECESSOR <BLANK> |
| 16 | (10) | CHARACTER | 4 | ADDEPWSID | WORKSTATION NAME |
| 20 | (14) | SIGNED | 4 | ADDEPOPNO | OPERATION NUMBER |

Table 167. ADDEP Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|------------|-----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 24 | (18) | SIGNED | 4 | ADDEPOWNOP | OWNING OP (THE SUCCESSOR) |
| 28 | (1C) | SIGNED | 4 | ADDEPTPT | TRANSPORT TIME IN MINUTES |
| 32 | (20) | CHARACTER | 50 | ADDEPDESC | DESCRIPTION |
| 82 | (52) | CHARACTER | 1 | ADDEPLTP | LTP REPORT PRINT OPTION A C |
| 83 | (53) | UNSIGNED | 1 | ADDEPVERS | RECORD VERSION NUMBER=1 |
| 84 | (54) | CHARACTER | 8 | ADDEPJOB | JOBNAME (NOT ALWAYS SET) |
| 92 | (5C) | CHARACTER | 1 | ADDEPFLAG | FLAGS |
| 93 | (5D) | CHARACTER | 1 | ADDEPCSEL | RESOLUTION CRITERIA C/S/R/A |
| 94 | (5E) | CHARACTER | 1 | ADDEPXMAND | IS MANDATORY N/P/C |
| 95 | (5F) | CHARACTER | 1 | * | FREE |

ADCNC - Condition segment

An operation condition.

| Offsets | Type | Length | Name | Description |
|---------|----------------|--------|-------------|--|
| 0 | (0) STRUCTURE | 56 | ADCNC | AD OPERATION CONDITION |
| 0 | (0) SIGNED | 4 | ADCNCOWNID | OWNING AD OPERATION |
| 4 | (4) SIGNED | 4 | ADCNCID | CONDITION ID |
| 8 | (8) SIGNED | 4 | ADCNCSIMPNO | NUMBER OF CONDITION DEPENDENCIES |
| 12 | (C) CHARACTER | 1 | * | NOT USED |
| 13 | (D) UNSIGNED | 1 | ADCNCVERS | VERSION |
| 14 | (E) CHARACTER | 2 | * | FREE |
| 16 | (10) SIGNED | 4 | ADCNCCOUNT | RULE TYPE: 0 = ALL N>0 = AT LEAST N OF |
| 20 | (14) CHARACTER | 24 | ADCNCDESC | OPERATION DESCRIPTION |
| 44 | (2C) CHARACTER | 12 | * | FREE |

ADCNS - Condition dependency segment

An operation condition dependency.

| Offsets | Type | Length | Name | Description |
|---------|----------------|--------|--------------|--|
| 0 | (0) STRUCTURE | 74 | ADCNS | AD OPERATION CONDITION DEPENDENCY |
| 0 | (0) SIGNED | 4 | ADCNSOWNID | OWNING AD OPERATION |
| 4 | (4) SIGNED | 4 | ADCNSID | CONDITION ID |
| 8 | (8) CHARACTER | 24 | ADCNSPREDID | PREDECESSOR ID: |
| 8 | (8) CHARACTER | 16 | ADCNSPREAD | |
| 24 | (18) CHARACTER | 8 | ADCNSPREOP | |
| 24 | (18) CHARACTER | 4 | ADCNSPREWSID | |
| 28 | (1C) SIGNED | 4 | ADCNSPREOPNO | |
| 32 | (20) CHARACTER | 1 | ADCNSDEPTYP | DEPENDENCY TYPE: I: INTERNAL E: EXTERNAL |
| 33 | (21) CHARACTER | 2 | ADCNSPRETYP | CHECK TYPE: RC: RETURN CODE ST: STATUS |
| 35 | (23) CHARACTER | 2 | ADCNSPRELOG | LOGICAL OPERATOR TYPE: GE: >= GREATER EQUAL GT: > GREATER LE: <= LESS EQUAL LT: < LESS |

| | | | | | |
|----|------|-----------|---|-------------|--------------------------------|
| 37 | (25) | CHARACTER | 4 | ADCNSVALRC | EQ: = EQUAL |
| 41 | (29) | CHARACTER | 4 | ADCNSVALRC2 | RG: = RANGE |
| 45 | (2D) | CHARACTER | 1 | ADCNSVALST | RC VALUE |
| | | | | | RC2 VALUE (FOR RANGE) |
| | | | | | ST VALUE: |
| | | | | | S: STARTED |
| | | | | | C: COMPLETED |
| | | | | | X: SUPPRESSED BY CONDITION |
| | | | | | E: ERROR |
| 46 | (2E) | CHARACTER | 8 | ADCNSPROC | STEP NAME |
| 54 | (36) | CHARACTER | 8 | ADCNSSTEP | PROCEDURE INVOCATION STEP NAME |
| 62 | (3E) | UNSIGNED | 1 | ADCNSVERS | VERSION |
| 63 | (3F) | CHARACTER | 1 | ADCNSCCSEL | RESOLUTION CRITERIA C/S/R/A |
| 64 | (40) | CHARACTER | 1 | * | RESERVED |
| 65 | (41) | CHARACTER | 9 | * | FREE |

ADEXT - Extended name segment

The extended name of an operation.

Table 168. ADEXT Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-------------|--------------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 100 | ADEXT | EXTENDED INFORMATION OF AD OPERATION |
| 0 | (0) | CHARACTER | 54 | ADEXTNAME | EXTENDED NAME |
| 54 | (36) | UNSIGNED | 1 | ADEXTVERS | RECORD VERSION NUMBER = 2 |
| 55 | (37) | CHARACTER | 1 | * | RESERVED |
| 56 | (38) | SIGNED | 4 | ADEXTOWNOP | OWNING OP NUMBER |
| 60 | (3C) | CHARACTER | 16 | ADEXTSENAME | SCHEDULING ENVIRONMENT NAME |
| 76 | (4C) | CHARACTER | 24 | * | RESERVED |

ADKEY - Key segment

The program interface LIST request with the ADKEY resource code lets you get a short version of the ADCOM segment consisting of only the application description key fields. The name of this segment is ADKEY and it contains only the first three fields of the ADCOM segment: ADID, ADSTAT, and ADTO.

ADOP - Operation segment

The operation part of an application description.

Note: Certain values are used to show a default or that the field has no value:

ADOPSM = -1

The default should be used.

ADOPLIM = -1

The default should be used.

ADOPHRC = -1

The field is not set.

ADOPHRC = -1

The field is not set.

Table 169. ADOP Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 160 | ADOP | OPERATION OF AN AD |
| 0 | (0) | CHARACTER | 4 | ADOPWSID | WORKSTATION |
| 4 | (4) | SIGNED | 4 | ADOPNO | OPERATION NUMBER |
| 8 | (8) | CHARACTER | 8 | ADOPJN | JOBNAME |
| 16 | (10) | CHARACTER | 24 | ADOPDESC | OPERATION DESCRIPTION |
| 40 | (28) | SIGNED | 4 | ADOPDUR | DURATION IN MINUTES |
| 44 | (2C) | SIGNED | 4 | ADOPSM | SMOOTHING FACTOR (OR -1) |
| 48 | (30) | SIGNED | 4 | ADOPLIM | LIMIT FOR FEEDBACK (OR -1) |
| 52 | (34) | SIGNED | 4 | ADOPHRC | HIGHEST OK RC (OR -1) |
| 56 | (38) | SIGNED | 4 | ADOPSTD | RELATIVE DAY INPUT ARRIVAL |
| 60 | (3C) | CHARACTER | 4 | ADOPSTT | INPUT ARRIVAL TIME |
| 64 | (40) | SIGNED | 4 | ADOPDD | RELATIVE DAY DEADLINE |
| 68 | (44) | CHARACTER | 4 | ADOPDT | DEADLINE TIME |
| 72 | (48) | SIGNED | 4 | ADOP#R1 | NUMBER OF R1 RESOURCES REQUIRED |
| 76 | (4C) | SIGNED | 4 | ADOP#R2 | NUMBER OF R2 RESOURCES REQUIRED |
| 80 | (50) | SIGNED | 4 | ADOP#PS | NUMBER OF SERVERS USED |
| 84 | (54) | CHARACTER | 1 | ADOPJCL | JOB CLASS |
| 85 | (55) | CHARACTER | 1 | ADOPPCL | PRINT CLASS |
| 86 | (56) | CHARACTER | 8 | ADOPFOR | FORM NUMBER |
| 94 | (5E) | CHARACTER | 1 | ADOPSUB | AUTOMATIC SUBMIT Y N |
| 95 | (5F) | CHARACTER | 1 | ADOPAJR | AUTOMATIC CPU RELEASE Y N |
| 96 | (60) | CHARACTER | 1 | ADOPCAN | CANCEL IF LATE TIME Y N |
| 97 | (61) | CHARACTER | 1 | ADOPTIM | SUBMIT JOB ON TIME Y N |
| 98 | (62) | CHARACTER | 1 | ADOPAEC | AUTOMATIC ERROR COMPL Y N |
| 99 | (63) | UNSIGNED | 1 | ADOPVERS | RECORD VERSION NUMBER = 2 |
| 100 | (64) | CHARACTER | 1 | ADOPWTO | DEADLINE WTO Y N |
| 101 | (65) | CHARACTER | 1 | ADOPRES | RESTARTABLE Y N <BLANK> |
| 102 | (66) | CHARACTER | 1 | ADOPRER | REROUTEABLE Y N <BLANK> |
| 103 | (67) | CHARACTER | 1 | ADOPCM | RESTART AND CLEANUP A=AUTOMATIC I=IMMEDIATE M=MANUAL N=NONE |
| 104 | (68) | CHARACTER | 8 | ADOPWSINFO | WORKSTATION INFO |

Table 169. ADOP Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|--------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 104 | (68) | CHARACTER | 1 | ADOPWSISET | INFO AVAILABLE Y N |
| 105 | (69) | CHARACTER | 1 | ADOPWSTYPE | TYPE G C P |
| 106 | (6A) | CHARACTER | 1 | ADOPWSREP | REPORTING ATTRIBUTE A S C N |
| 107 | (6B) | CHARACTER | 1 | ADOPWSSUBT | SUBTYPE JCL, STC, WTO, none J S W blank |
| 108 | (6C) | CHARACTER | 4 | * | RESERVED |
| 112 | (70) | CHARACTER | 1 | ADOPJCRT | (WLM) CRITICAL JOB |
| 113 | (71) | CHARACTER | 1 | ADOPJPOL | (WLM) LATE JOB POLICY |
| 114 | (72) | CHARACTER | 1 | ADOPUSRSYS | USER SYSOUT NEEDED |
| 115 | (73) | CHARACTER | 1 | ADOPEXPJCL | EXPANDED JCL NEEDED |
| 116 | (74) | SIGNED | 4 | ADOPDURI | DURATION IN 100TH OF SEC |
| 120 | (78) | CHARACTER | 1 | ADOPMON | OPERATION MONITORED |
| 121 | (79) | CHARACTER | 1 | ADOPCENSCR | CENTRALIZED SCRIPT |
| 122 | (7A) | CHARACTER | 1 | ADOPUSEEXT | USE ADEXTNAME FIELD |
| 123 | (7B) | CHARACTER | 1 | ADOPUSESE | USE ADEXTSE FIELD |
| 124 | (7C) | CHARACTER | 1 | ADOPUSESA | USE SYSTEM AUTOMATION Y N |
| 125 | (7D) | CHARACTER | 8 | ADOPWLMCLASS | WLM SERVICE CLASS |
| 133 | (85) | CHARACTER | 1 | ADOPCONDRJOB | CONDITIONAL RECOVERY JOB |
| 134 | (86) | CHARACTER | 1 | ADOPNOP | NOP JOB |
| 135 | (87) | CHARACTER | 1 | ADOPMH | MANUALLY HOLD JOB |
| 136 | (88) | CHARACTER | 24 | * | RESERVED |

ADRE - Remote job information segment

A segment containing the remote job information.

Table 170. ADRE Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-------------|---------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 96 | ADRE | |
| 0 | (0) | CHARACTER | 16 | ADRE_JSNAME | ADID OR JOB STREAM NAME |
| 16 | (10) | UNSIGNED | 1 | ADRE_VERS | RECORD VERSION NUMBER = 1 |
| 17 | (11) | CHARACTER | 1 | ADRE_COMPL | COMPLETE ON FAILED BIND |
| 18 | (12) | CHARACTER | 2 | * | RESERVED |
| 20 | (14) | SIGNED | 4 | ADRE_OPNO | OPERATION NUMBER |

Table 170. ADRE Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|--------------|------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 24 | (18) | CHARACTER | 16 | ADRE_JSWS | JOB STREAM WORKSTATION |
| 40 | (28) | CHARACTER | 40 | ADRE_JOBNAME | JOB NAME |
| 80 | (50) | SIGNED | 4 | ADRE_OWNOP | OWNING OP NUMBER |
| 84 | (54) | CHARACTER | 12 | * | RESERVED |
| | | | | | |

ADRUN - Run cycle segment

The run cycle part of an application description. A run cycle is based either on offsets or on rules. The segment contains the fixed part plus either run cycle offsets or a rule definition.

Type Required input.

For run cycles based on offsets, type is:

- N** Normal run cycle that identifies times and days when the application runs.
- X** Negative run cycle that identifies times and days when the application does NOT run. If you specify a particular day and time as a negative run cycle, no occurrences of the application are generated for that day and time, regardless of what is generated by a normal or regular run cycle. Run cycles are used in conjunction; negative run cycles are used to suppress run days generated by normal or regular run cycles.

For run cycles based on rules, type is:

- R** Regular run cycle that identifies times and days when the application runs.
- E** Exclusion run cycle that identifies times and days when the application does NOT run. If you specify a particular day and time as an exclusion run cycle, no occurrences of the application are generated for that day and time, regardless of what is generated by a regular or normal run cycle. Run cycles are used in conjunction; exclusion run cycles are used to suppress run days generated by regular or normal run cycles.

Free day rule

Required input for all run cycles, which indicates how run days are treated:

- E** Free days excluded; only work days are taken into account
- 1** Free days included; run on the nearest day *before* the free day
- 2** Free days included; run on the nearest day *after* the free day
- 3** Free days included; run *on* the free day
- 4** Free days included; do *not* run at all.

Note: ADRIADALL is the start of either run cycle offsets or a rule. EQQPIFAD sample shows how to handle it.

Table 171. ADRUN Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|---------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 228 | ADRUN | RUNCYCLE SECTION |
| 0 | (0) | CHARACTER | 8 | ADRPER | PERIOD NAME |
| 8 | (8) | CHARACTER | 6 | ADRVALF | RUN CYCLE VALID-FROM |
| 14 | (E) | CHARACTER | 6 | ADRVALT | RUN CYCLE VALID-TO |
| 20 | (14) | CHARACTER | 50 | ADRUNDESC | RUN CYCLE DESCRIPTION |
| 70 | (46) | CHARACTER | 1 | ADRUNRULE | RULE FOR WORK/FREE DAYS |
| 71 | (47) | CHARACTER | 1 | ADRTYPE | PERIOD BASED (N X) RULE BASED (R E) |
| 72 | (48) | SIGNED | 4 | ADRIAD(24) | OFFSETS (START DAYS WITHIN PERIOD) |
| 168 | (A8) | CHARACTER | 4 | ADRIAT | INPUT ARRIVAL TIME |
| 172 | (AC) | SIGNED | 4 | ADRDD | DEADLINE DAY RELATIVE TO START |
| 176 | (B0) | CHARACTER | 4 | ADRDT | DEADLINE TIME |
| 180 | (B4) | UNSIGNED | 1 | ADRUNVERS | RECORD VERSION NUMBER=1 |
| 181 | (B5) | CHARACTER | 16 | ADRJV TAB | JCL VARIABLE TABLE |
| 197 | (C5) | CHARACTER | 1 | ADRSHTYPE | SHIFT TYPE (W/D or blank) |
| 198 | (C6) | SIGNED | 2 | ADRINPOS | NUMBER OF POSITIVE RUN CYCLE OFFSETS |
| 200 | (C8) | SIGNED | 2 | ADRINNEG | NUMBER OF NEGATIVE RUN CYCLE OFFSETS |
| 202 | (CA) | SIGNED | 2 | ADRIRDLEN | RULE DEFINITION LENGTH |
| 204 | (CC) | CHARACTER | 4 | ADRREPEATEVRY | REPEAT EVERY |
| 208 | (D0) | CHARACTER | 4 | ADRREPEATENDT | REPEAT END TIME |
| 212 | (D4) | SIGNED | 4 | ADRSHIFT | SHIFT VALUE (-999 to 999) |
| 216 | (D8) | CHARACTER | 12 | * | RESERVED |
| 228 | (E4) | CHARACTER | * | ADRIADALL | START OF RUN CYCLE OFFSETS OR A RULE |

Table 172. Run Cycle Offsets

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|--|
| Dec | Hex | Type | Len | Name | Description |
| 228 | (E4) | STRUCTURE | * | ADRIADALL | START OF RUN CYCLE OFFSETS |
| 228 | (E4) | SIGNED | 4 | ADRIA OFF | ARRAY OF RUN CYCLE OFFSETS (LENGTH=(ADRINPOS+ADRINNEG)*4) |

Run cycle offsets are an array of positive fullwords. ADRINPOS and ADRINNEG identify the number of entries in the array. The positive offsets are first.

If the total number of offsets is 24 or less, the offsets are also found in the ADRIAD array. ADRIAD is an array of 24 integer values that specify the start days within the period. Each nonzero value defines a day that the run cycle selects; that is, when the application runs if ADRTYPE is N, or does not run if ADRTYPE is X. The first day of the period is specified by 1 and the last day by -1. The first zero value ends the array.

Table 173. Rule Definition

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|---------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 228 | (E4) | STRUCTURE | * | ADRIADALL | RULE DEFINITION |
| 228 | (E4) | SIGNED | 4 | ADRULEL | RULE LENGTH (ADRULEL + ADRULET) |
| 232 | (E8) | CHARACTER | * | ADRULET | RULE TEXT |

For a rule-based run cycle, ADRIRDLEN identifies the length of the rule definition. The ADRIADALL structure contains a fullword copy of ADRIRDLEN (ADRULEL), which is followed by the rule text. ADRULEL must specify the same length as ADRIRDLEN. You can insert comments or extra blanks when creating a rule, but these characters are not saved in the AD database. The syntax of the rule text is the same as for the ADRULE control statement used by the batch loader. See IBM Workload Scheduler for z/OS Managing the Workload for more information.

Here is an example of a rule definition, which selects the third day in each month:

```
ADRULEL 33 (X'21')
ADRULET 'ADRULE ONLY(3) DAY(DAY) MONTH'
```

Note: Note that the ADOP segment is enlarged by 32 characters. This will not affect current program interface applications until in a future release, when the reserved field becomes used for operation data.

ADSAI - Operation system automation information segment

System automation information.

Note: This segment exists for system automation operations only.

Table 174. ADSAI Control Block

| Offsets | | | | | |
|---------|-----|-----------|-----|---------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 352 | ADSAI | SYSTEM AUTOMATION INFO FOR AD OPERATION |
| 0 | (0) | CHARACTER | 256 | ADSAICOMMTEXT | SYSTEM AUTOMATION OPERATION COMMAND TEXT |
| 0 | (0) | CHARACTER | 64 | ADSAICOMMTEX1 | SYSTEM AUTOMATION OPERATION COMMAND TEXT, ROW 1 |

Table 174. ADSAI Control Block (continued)

| Offsets | | | | | |
|---------|-------|-----------|-----|---------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 64 | (40) | CHARACTER | 64 | ADSAICOMMTEX2 | SYSTEM AUTOMATION OPERATION COMMAND TEXT, ROW 2 |
| 128 | (80) | CHARACTER | 64 | ADSAICOMMTEX3 | SYSTEM AUTOMATION OPERATION COMMAND TEXT, ROW 3 |
| 192 | (C0) | CHARACTER | 63 | ADSAICOMMTEX4 | SYSTEM AUTOMATION OPERATION COMMAND TEXT, ROW 4 |
| 255 | (FF) | CHARACTER | 1 | ADSAIFILLER | RESERVED |
| 256 | (100) | CHARACTER | 8 | ADSAIAUTOOPER | SYSTEM AUTOMATED OPERATOR |
| 264 | (108) | CHARACTER | 8 | ADSAISECELEM | SYSTEM AUTOMATION SECURITY ELEMENT |
| 272 | (110) | CHARACTER | 64 | ADSAICOMPINFO | SYSTEM AUTOMATION COMPLETION INFORMATION |
| 336 | (150) | CHARACTER | 4 | * | RESERVED |
| 340 | (154) | SIGNED | 4 | ADSAIOWNOP | OWNING OPERATION NUMBER |
| 344 | (158) | CHARACTER | 8 | * | RESERVED |

ADSR - Special resource segment

The special resource part of an application description.

Table 175. ADSR Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|--|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 64 | ADSR | SPECIAL RESOURCE SECTION |
| 0 | (0) | CHARACTER | 44 | ADSRN | SPECIAL RESOURCE NAME |
| 44 | (2C) | SIGNED | 4 | ADSROWNOP | OWNING OPERATION NUMBER |
| 48 | (30) | CHARACTER | 1 | ADSRRT | S = SHARED, X = EXCLUSIVE |
| 49 | (31) | UNSIGNED | 1 | ADSRVERS | RECORD VERSION NUMBER = 2 |
| 50 | (32) | CHARACTER | 1 | ADSRONER | KEEP ON ERROR (Y N blank) |
| 51 | (33) | CHARACTER | 1 | * | FREE |
| 52 | (34) | SIGNED | 4 | ADSRAMNT | QUANTITY REQUIRED. THE VALUE 0 MEANS THE TOTAL QUANTITY OF SPECIAL RESOURCE. |
| 56 | (38) | CHARACTER | 1 | ADSRVACO | ON COMPLETE (Y N R blank) |
| 57 | (39) | CHARACTER | 7 | * | RESERVED |

ADUSF - User field segment

An operation user field.

Table 176. ADUSF Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|------------|-------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 84 | ADUSF | AD OPERATION USER FIELD |
| 0 | (0) | SIGNED | 4 | ADUSFOWNID | OWNING AD OPERATION |
| 4 | (4) | CHARACTER | 16 | ADUSFNAME | USER FIELD NAME |
| 20 | (14) | CHARACTER | 54 | ADUSFVALUE | USER FIELD VALUE |
| 74 | (4A) | CHARACTER | 2 | * | NOT USED |
| 76 | (4C) | UNSIGNED | 1 | ADUSFVERS | VERSION |
| 77 | (4D) | CHARACTER | 7 | * | NOT USED |

ADVDD - Variable Duration and Deadline

An operation variable duration and deadline.

Table 177. ADVDD Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 27 | ADVDD | AD OPERATION VARIABLE DURATION AND DEADLINE |
| 0 | (0) | SIGNED | 4 | ADVDDOWNID | OWNING AD OPERATION |
| 4 | (4) | SIGNED | 4 | ADVDDDUR | VARIABLE DURATION |
| 8 | (8) | CHARACTER | 8 | ADVDDDEAD | VARIABLE DEADLINE |
| 8 | (8) | SIGNED | 4 | ADVDDDEADD | VARIABLE DEADLINE RELATIVE DAY |
| 12 | (C) | CHARACTER | 4 | ADVDDDEADT | VARIABLE DEADLINE TIME |
| 16 | (10) | CHARACTER | 8 | ADVDDRG | RUN CYCLE GROUP NAME |
| 24 | (18) | CHARACTER | 1 | ADVDDNOP | NOP JOB (Y, N, or blank) |
| 25 | (19) | CHARACTER | 1 | ADVDDMH | MANUALLY HOLD JOB (Y, N, or blank) |
| 26 | (1A) | CHARACTER | 1 | ADVDDCRJ | CRITICAL JOB (N, P, W, or blank) |

ADXIV - Interval definition for external predecessor segment

The interval definition for an external predecessor. Used when ADDEP ADDEPCSEL has value R or A (only one ADXIV per ADDEP can be used).

Table 178. ADXIV Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 47 | ADXIV | EXTERNAL PREDECESSOR INTERVAL |
| 0 | (0) | CHARACTER | 16 | ADXIVADID | PREDECESSOR APPLICATION NAME |
| 16 | (10) | CHARACTER | 4 | ADXIVWSID | PREDECESSOR WORKSTATION NAME |
| 20 | (14) | SIGNED | 4 | ADXIVOPNO | PREDECESSOR OPERATION NUMBER |
| 24 | (18) | SIGNED | 4 | ADXIVOWNOP | OWNING OPERATION NUMBER |
| 28 | (1C) | CHARACTER | 1 | ADXIVTYPE | INTERVAL TYPE R/A (RELATIVE/ ABSOLUTE) |
| 29 | (1D) | CHARACTER | 1 | ADXIVFWHE | FROM WHEN B/A (BEFORE/AFTER) |
| 30 | (1E) | CHARACTER | 3 | ADXIVFHHH | FROM HOURS HHH (ONLY RELATIVE INTERVAL) |
| 33 | (21) | CHARACTER | 2 | ADXIVFHH | FROM HOURS HH (ONLY ABSOLUTE INTERVAL) |
| 35 | (23) | CHARACTER | 2 | ADXIVFMM | FROM MINUTES MM |
| 37 | (25) | CHARACTER | 1 | ADXIVFD | FROM DAYS (ONLY ABSOLUTE INTERVAL) |
| 38 | (26) | CHARACTER | 1 | ADXIVTWHE | TO WHEN B/A (BEFORE/AFTER) |
| 39 | (27) | CHARACTER | 3 | ADXIVTHHH | TO HOURS HHH (ONLY RELATIVE INTERVAL) |
| 42 | (2A) | CHARACTER | 2 | ADXIVTHH | TO HOURS HH (ONLY ABSOLUTE INTERVAL) |
| 44 | (2C) | CHARACTER | 2 | ADXIVTMM | TO MINUTES MM |
| 46 | (2E) | CHARACTER | 1 | ADXIVTD | TO DAYS (ONLY ABSOLUTE INTERVAL) |

All workstations closed (resource code AWSCL)

There is no common segment. One segment exists for each interval when all workstations are closed.

AWSCL - All workstations closed interval segment

Description of an interval when all workstations are closed.

Table 179. AWSCL Control Block

| Offsets | | | | | |
|---------|-----|-----------|-----|---------|--------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 80 | AWSCL | WS CLOSED INTERVAL |
| 0 | (0) | CHARACTER | 6 | AWCKEY | UNIQUE IDENTIFIER |
| 0 | (0) | CHARACTER | 6 | AWCDATE | DATE |
| 6 | (6) | CHARACTER | 4 | AWCFROM | FROM TIME |

Table 179. AWSCL Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|----------|-----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 10 | (A) | CHARACTER | 4 | AWCTO | TO TIME |
| 14 | (E) | CHARACTER | 30 | AWCDESC | DESCRIPTION CLOSED INTERVAL |
| 44 | (2C) | UNSIGNED | 1 | AWCVERS | VERSION OF RECORD=1 |
| 45 | (2D) | CHARACTER | 6 | AWCLDATE | DATE LAST UPDATED |
| 51 | (33) | CHARACTER | 4 | AWCLTIME | TIME LAST UPDATED |
| 55 | (37) | CHARACTER | 8 | AWCLUSER | USERID OF LAST UPDATER |
| 63 | (3F) | CHARACTER | 1 | * | RESERVED |
| 64 | (40) | CHARACTER | 8 | AWCLLUTS | TOD CLOCK AT LAST UPDATE |
| 72 | (48) | CHARACTER | 8 | * | RESERVED |

Calendar (resource codes CL, CLCOM)

Each calendar record can contain these segments:

CLCOM

Common segment. Only one common segment must appear as the first segment in each record.

CLSD Specific date segment.

CLWD

Specific day of week segment.

Note: For a correct interpretation of the fields described as "Tod clock at last update", see "TOD fields" on page 271.

CLCOM - Common segment

Common description of a calendar.

Table 180. CLCOM Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|---------|----------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 96 | CLCOM | |
| 0 | (0) | CHARACTER | 16 | CLKEY | UNIQUE IDENTIFIER |
| 0 | (0) | CHARACTER | 16 | CLNAME | CALENDER NAME |
| 16 | (10) | SIGNED | 4 | CLDAYS | NUMBER OF SPECIFIC AND WEEK DAYS |
| 20 | (14) | CHARACTER | 4 | CLSHIFT | END TIME OF A SHIFT |
| 24 | (18) | CHARACTER | 30 | CLDESC | DESCRIPTION |
| 54 | (36) | UNSIGNED | 1 | CLVERS | VERSION OF RECORD=1 |
| 55 | (37) | CHARACTER | 6 | CLLDATE | DATE LAST UPDATED |

Table 180. CLCOM Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|---------|--------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 61 | (3D) | CHARACTER | 4 | CLLTIME | TIME LAST UPDATED |
| 65 | (41) | CHARACTER | 8 | CLLUSER | USER ID OF LAST UPDATER |
| 73 | (49) | CHARACTER | 7 | * | RESERVED |
| 80 | (50) | CHARACTER | 8 | CLLUTS | TOD CLOCK AT LAST UPDATE |
| 88 | (58) | CHARACTER | 8 | * | RESERVED |
| | | | | | |

CLSD - Specific date segment

Calendar description: a specific date.

Day status can be:

W Work

F Free

Table 181. CLSD Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|-------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 48 | CLSD | |
| 0 | (0) | CHARACTER | 6 | CLSDDATE | SPECIFIC DATE |
| 6 | (6) | CHARACTER | 2 | * | RESERVED |
| 8 | (8) | CHARACTER | 1 | CLSDSTAT | STATUS, WORK OR FREE |
| 9 | (9) | CHARACTER | 30 | CLSDDESC | DESCRIPTION OF THE DATE |
| 39 | (27) | CHARACTER | 9 | * | RESERVED |
| | | | | | |

CLWD - Weekday segment

Calendar description: a weekday.

A weekday can be:

MONDAY

TUESDAY

WEDNESDAY

THURSDAY

FRIDAY

SATURDAY

SUNDAY

Note: WEDNESDAY is actually stored as WEDNESDA.

Table 182. CLWD Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|-------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 48 | CLWD | |
| 0 | (0) | CHARACTER | 8 | CLWDDAY | WEEK DAY |
| 8 | (8) | CHARACTER | 1 | CLWDSTAT | STATUS, WORK OR FREE |
| 9 | (9) | CHARACTER | 30 | CLWDDESC | DESCRIPTION OF THE DATE |
| 39 | (27) | CHARACTER | 9 | * | RESERVED |
| | | | | | |

Current plan condition (resource codes CPCOND, CPCONDCO)

The current plan condition record can contain these segments:

CPCOND

Common segment. Only one CPCOND must be provided.

CPSIMP

Conditional dependency segment.

CPCOND - Condition segment

Current plan operation condition.

| Offsets | Type | Length | Name | Description |
|---------|------|-----------|------|---|
| 0 | (0) | STRUCTURE | 134 | CPCONDCO CURRENT PLAN OPERATION CONDITION - KEY FIELDS: ----- |
| 0 | (0) | CHARACTER | 16 | CPCOADI APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | CPCOIA APPLICATION INPUT ARRIVAL |
| 16 | (10) | CHARACTER | 6 | CPCOIAD MODIFIED IF IA IS MODIFIED |
| 22 | (16) | CHARACTER | 4 | CPCOIAT ELSE ORIGINAL FROM PLAN |
| 26 | (1A) | SIGNED | 4 | CPCOOPNO OPERATION NUMBER |
| 30 | (1E) | SIGNED | 4 | CPCOCID CONDITION ID ----- |
| 34 | (22) | CHARACTER | 24 | CPCODESC CONDITION DESCRIPTION |
| 58 | (3A) | CHARACTER | 1 | * |
| 59 | (3B) | CHARACTER | 1 | * |
| 60 | (3C) | SIGNED | 4 | CPCO#SIMP NUMBER OF CONDITION DEPENDENCIES |
| 64 | (40) | SIGNED | 4 | CPCOCOUNT RULE TYPE: 0 = ALL N>0 = AT LEAST N OF |
| 68 | (44) | CHARACTER | 1 | CPCOVALUE FINAL CONDITION STATUS: U: UNDECIDED T: TRUE F: FALSE |
| 69 | (45) | UNSIGNED | 1 | CPCOVERS VERSION |
| 70 | (46) | CHARACTER | 1 | CPCOXST COND EXTENDED STATUS |
| 71 | (47) | CHARACTER | 63 | * |

CPSIMP - Condition dependency segment

Current plan operation condition dependency.

| Offsets | Type | Length | Name | Description |
|---------|------|-----------|------|--|
| 0 | (0) | STRUCTURE | 85 | CPSIMP CURRENT PLAN OPERATION CONDITION DEPENDENCY KEY FIELDS: ----- |
| 0 | (0) | CHARACTER | 16 | CPSIPREADI APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | CPSIPREIA APPLICATION INPUT ARRIVAL |
| 16 | (10) | CHARACTER | 6 | CPSIPREIAD MODIFIED IF IA IS MODIFIED |
| 22 | (16) | CHARACTER | 4 | CPSIPREIAT ELSE ORIGINAL FROM PLAN |
| 26 | (1A) | SIGNED | 4 | CPSIPREOPNO OPERATION NUMBER |
| 30 | (1E) | CHARACTER | 2 | CPSITYP CHECK TYPE: RC OR ST |
| 32 | (20) | CHARACTER | 2 | CPSILOG OPERATOR: GE, GT, LE, LT, EQ, |

| | | | | | |
|-------|------|-----------|---|--------------|------------------------------------|
| 34 | (22) | CHARACTER | 4 | CPSIVALRC | NE, RG RC VALUE |
| 38 | (26) | CHARACTER | 4 | CPSIVALRC2 | RC2 VALUE |
| 42 | (2A) | CHARACTER | 1 | CPSIVALST | ST VALUE |
| ----- | | | | | |
| 43 | (2B) | CHARACTER | 1 | CPSILVAL | CONDITION DEPENDENCY STATUS: U T F |
| 44 | (2C) | UNSIGNED | 1 | CPSIVERS | VERSION |
| 45 | (2D) | CHARACTER | 1 | CPSIREMOVED | CONDITION DEPENDENCY REMOVED: Y/N |
| 46 | (2E) | CHARACTER | 1 | CPSISTEPMISS | MISSING STEP END INFORMATION: Y/N |
| 47 | (2F) | CHARACTER | 8 | CPSISTEP | PROCEDURE INVOCATION STEP NAME |
| 55 | (37) | CHARACTER | 8 | CPSIPSTEP | STEP NAME |
| 63 | (3F) | CHARACTER | 8 | CPSIJOBNAME | JOB NAME |
| 71 | (47) | CHARACTER | 4 | CPSIWSNAME | WS NAME |
| 75 | (4B) | CHARACTER | 1 | CPSINWSTAT | NEW STATUS: T F |
| 76 | (4C) | CHARACTER | 9 | * | FREE |

Current plan occurrence (resource code CPOC, CPOCCOM)

The current plan occurrence record consists always of the following segment:

CPOC Current plan occurrence common segment. Only one common segment must exist.

It can optionally consist of the following segments:

CPOCPRE

Occurrence predecessor segment.

CPOCSUC

Occurrence successor segment.

CPOC - Current plan occurrence segment

Current plan occurrence.

Note:

1. Minutes are the unit of duration.
2. Y and N are the indicator values.
3. Actual arrival, CPOCAA, for manually completed occurrences is blank, if no operations have started.

ADDING FUNCTION

Blank The daily plan batch program

A Automatic recovery

D Dialog (Modify Current Plan dialog)

E ETT, event-triggered tracking

P PIF, program interface

Table 183. CPOC Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|---------|----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 428 | CPOC | CURRENT PLAN OCCURRENCE |
| 0 | (0) | CHARACTER | 16 | CPOCADI | APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | CPOCIA | INPUT ARRIVAL |
| 16 | (10) | CHARACTER | 6 | CPOCIAD | MODIFIED IF IA IS MODIFIED |
| 22 | (16) | CHARACTER | 4 | CPOCIAT | ELSE ORIGINAL FROM PLAN |

Table 183. CPOC Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|-------------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 26 | (1A) | CHARACTER | 8 | CPOCGRP | AUTHORITY GROUP |
| 34 | (22) | CHARACTER | 10 | CPOCIAO | INPUT ARRIVAL FROM LTP |
| 34 | (22) | CHARACTER | 6 | CPOCIAOD | DATE |
| 40 | (28) | CHARACTER | 4 | CPOCIAOT | TIME |
| 44 | (2C) | CHARACTER | 24 | CPOCDESC | DESCRIPTIVE TEXT |
| 68 | (44) | CHARACTER | 16 | CPOCOID | OWNER ID |
| 84 | (54) | CHARACTER | 24 | CPOCODES | OWNER DESCRIPTION |
| 108 | (6C) | CHARACTER | 10 | CPOCDL | DEADLINE |
| 108 | (6C) | CHARACTER | 6 | CPOCDLD | DATE |
| 114 | (72) | CHARACTER | 4 | CPOCDLT | TIME |
| 118 | (76) | CHARACTER | 10 | CPOCAA | ACTUAL ARRIVAL |
| 118 | (76) | CHARACTER | 6 | CPOCAAD | IF ARRIVED |
| 124 | (7C) | CHARACTER | 4 | CPOCAAT | ELSE BLANKS |
| 128 | (80) | CHARACTER | 10 | CPOCAC | ACTUAL COMPLETION |
| 128 | (80) | CHARACTER | 6 | CPOCACD | IF COMPLETED |
| 134 | (86) | CHARACTER | 4 | CPOCACT | ELSE BLANKS |
| 138 | (8A) | CHARACTER | 4 | CPOCERR | OCCURRENCE ERROR CODE |
| 142 | (8E) | CHARACTER | 1 | CPOCST | OCCURRENCE STATUS |
| 143 | (8F) | CHARACTER | 1 | CPOCRER | RERUN REQUESTED (Y N) |
| 144 | (90) | CHARACTER | 1 | CPOCADDED | ADDED TO CURRENT PLAN (Y N) |
| 145 | (91) | CHARACTER | 1 | CPOCLATE | LATEST OUT PASSED (Y N) |
| 146 | (92) | CHARACTER | 1 | CPOCADDF | ADDING FUNCTION (E D P A) |
| 147 | (93) | CHARACTER | 1 | CPOCMON | MONITORING FLAG |
| 148 | (94) | SIGNED | 4 | CPOCPRI | PRIORITY |
| 152 | (98) | SIGNED | 4 | CPOC#OP | NUMBER OF OPERATIONS IN OCCURRENCE |
| 156 | (9C) | SIGNED | 4 | CPOCOPC | NUMBER OF OPERATIONS COMPLETED |
| 160 | (A0) | SIGNED | 4 | CPOC#ER | NUMBER OF OPERATIONS ENDED IN ERROR |
| 164 | (A4) | SIGNED | 4 | CPOC#UN | NUMBER OF OPERATIONS UNDECIDED |
| 168 | (A8) | SIGNED | 4 | CPOC#ST | NUMBER OF OPERATIONS STARTED |
| 172 | (AC) | SIGNED | 4 | CPOCRDU | REMAINING DUR CRITICAL PATH |
| 176 | (B0) | SIGNED | 4 | CPOCROP | REMAINING OPS CRITICAL PATH |
| 180 | (B4) | CHARACTER | 4 | CPOCCWS | WSNAME OF 1ST CRITICAL OP |

Table 183. CPOC Control Block (continued)

| Offsets | | | | | |
|---------|-------|-----------|-----|--------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 184 | (B8) | SIGNED | 4 | CPOCCOP | OP NO. OF 1ST CRITICAL OP |
| 188 | (BC) | UNSIGNED | 1 | CPOCVERS | VERSION NUMBER=1 |
| 189 | (BD) | CHARACTER | 16 | CPOCJVT | JCL VARIABLE TABLE |
| 205 | (CD) | CHARACTER | 1 | * | RESERVED NOT ADD |
| 206 | (CE) | CHARACTER | 16 | CPGROUPID | GROUP DEFINITION ID |
| 222 | (DE) | CHARACTER | 16 | CPOCCAL | CALENDAR NAME |
| 238 | (EE) | CHARACTER | 2 | * | RESERVED |
| 240 | (F0) | UNSIGNED | 4 | CPOCRDUI | REMAIN. DUR CRIT. PATH SEC |
| 244 | (F4) | CHARACTER | 4 | * | RESERVED |
| 248 | (F8) | CHARACTER | 8 | CPOCOCTO | OCCURRENCE TOKEN |
| 256 | (100) | CHARACTER | 10 | CPOCCLO | FIRST CRITICAL OP LATEST OUT |
| 256 | (100) | CHARACTER | 6 | CPOCCLOD | DATE |
| 262 | (106) | SIGNED | 4 | CPOCCLOT | TIME IN 100TH OF SEC. |
| 266 | (10A) | CHARACTER | 44 | CPOCETTCRIT | ETT CRITERIA |
| 310 | (136) | CHARACTER | 1 | CPOCETTTYP | ETT TYPE: J OR R |
| 311 | (137) | CHARACTER | 8 | CPOCETTJOB | ETT JOB NAME |
| 319 | (13F) | CHARACTER | 8 | CPOCETTJID | ETT JOB ID |
| 327 | (147) | CHARACTER | 35 | CPOCETTGROOT | ETT GDG ROOT |
| 362 | (16A) | CHARACTER | 44 | CPOCETTEVNAM | COMPLETE ETT EVENT NAME |
| 406 | (196) | CHARACTER | 8 | CPOCETTGEN | ETT GDG GENERATION |
| 414 | (19E) | CHARACTER | 6 | * | RESERVED |
| 420 | (1A4) | CHARACTER | 8 | CPOCRUNC | RUN CYCLE THAT GENERATED THE OCCURRENCE |

CPOCPRE - Occurrence predecessor segment

Current plan occurrence predecessor.

Table 184. CPOCPRE Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|------------|----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | CHARACTER | 16 | CPOCPREADI | APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | CPOCPREIA | INPUT ARRIVAL |
| 16 | (10) | CHARACTER | 6 | CPOCPREIAD | MODIFIED IF IA IS MODIFIED |
| 22 | (16) | CHARACTER | 4 | CPOCPREIAT | ELSE ORIGINAL FROM PLAN |

Table 184. CPOCPRE Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|-----------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 26 | (1A) | SIGNED | 4 | CPOCPRENO | OPERATION NUMBER |
| 30 | (1E) | CHARACTER | 1 | CPOCPRECO | PREDECESSOR COMPLETED (Y N) |
| 31 | (1F) | CHARACTER | 1 | CPOCPRENRR | PRED. WS WAS NONREPORTING |
| 32 | (20) | SIGNED | 4 | CPOCPRETT | TRANSPORT TIME |
| 36 | (24) | CHARACTER | 1 | CPOCPREND | PENDING PRED |
| 37 | (25) | UNSIGNED | 1 | CPOCPREVERS | VERSION NUMBER=1 |
| 38 | (26) | CHARACTER | 8 | CPOCPREJN | PREDECESSOR JOB NAME |
| 46 | (2E) | CHARACTER | 1 | CPOCPREST | PREDECESSOR STATUS |
| 47 | (2F) | CHARACTER | 1 | CPOCPMATC | PREDECESSOR RESOLUTION CRITERIA: BLANK (MANUALLY CHOSEN) C (CLOSEST PRECEDING) S (SAME DAY) A (ABSOLUTE INTERVAL) R (RELATIVE INTERVAL) |
| 48 | (30) | SIGNED | 4 | CPOCPRECRITPATH | PREDECESSOR OF AN OPERATION BELONGING TO A CRITICAL PATH |
| 52 | (34) | CHARACTER | 21 | * | RESERVED PER MAND PEND |
| 73 | (49) | CHARACTER | 7 | * | RESERVED |
| 80 | (50) | CHARACTER | 4 | * | RESERVED |

CPOCSUC - Occurrence successor segment

Current plan occurrence successor.

Table 185. CPOCSUC Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-------------|----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 48 | CPOCSUC | OPERATION SUCCESSOR |
| 0 | (0) | CHARACTER | 16 | CPOCSUCADI | APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | CPOCSUCIA | INPUT ARRIVAL |
| 16 | (10) | CHARACTER | 6 | CPOCSUCIAD | MODIFIED IF IA IS MODIFIED |
| 22 | (16) | CHARACTER | 4 | CPOCSUCIAT | ELSE ORIGINAL FROM PLAN |
| 26 | (1A) | SIGNED | 4 | CPOCSUCNO | OPERATION NUMBER |
| 30 | (1E) | CHARACTER | 1 | CPOCSUCCR | ON CRITICAL PATH (Y N) |
| 31 | (1F) | UNSIGNED | 1 | CPOCSUCVERS | VERSION NUMBER=1 |
| 32 | (20) | CHARACTER | 8 | CPOCSUCJN | SUCCESSOR JOB NAME |
| 40 | (28) | CHARACTER | 1 | CPOCSUCST | SUCCESSOR STATUS |

Table 185. CPOCSUC Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|------|-------------|
| Dec | Hex | Type | Len | Name | Description |
| 41 | (29) | CHARACTER | 7 | * | RESERVED |

Current plan operation (resource codes CPOP, CPOPCOM)

The current plan operation record can contain these segments:

CPCPR

Conditional predecessor segment.

CPCSU

Conditional successor segment.

CPEXT

Operation extended name segment.

CPOP Common segment. Only one CPOP, but it must be provided.

CPPRE

Predecessor segment.

CPREND

Distributed remote job info segment.

CPRENZ

z/OS remote job info segment.

CPSAI

Operation system automation information segment.

CPSUC

Successor segment.

CPSR Special resource segment.

CPREC

Operation recovery segment.

CPCPR - Conditional predecessor segment

Current plan operation conditional predecessor.

| Offsets | Type | Length | Name | Description |
|---------|----------------|--------|-------------|--|
| 0 | (0) STRUCTURE | 60 | CPCPRE | OPERATION CONDITIONAL PREDECESSOR |
| 0 | (0) CHARACTER | 16 | CPCPREADI | APPLICATION ID |
| 16 | (10) CHARACTER | 10 | CPCPREIA | INPUT ARRIVAL, |
| 16 | (10) CHARACTER | 6 | CPCPREIAD | MODIFIED IF IA IS MODIFIED |
| 22 | (16) CHARACTER | 4 | CPCPREIAT | ELSE ORIGINAL FROM PLAN |
| 26 | (1A) SIGNED | 4 | CPCPRENO | OPERATION NUMBER |
| 30 | (1E) SIGNED | 4 | CPCPRE_CID | CONDITION ID |
| 34 | (22) CHARACTER | 1 | CPCPRECO | PREDECESSOR COMPLETED (Y/N) |
| 35 | (23) CHARACTER | 1 | CPCPRENR | --PRED. WS WAS NON-REPORTING |
| 36 | (24) SIGNED | 4 | CPCPRETT | --TRANSPORT TIME (MIN) |
| 40 | (28) CHARACTER | 1 | CPCPREND | PENDING PRED. OCCURRENCE |
| 41 | (29) UNSIGNED | 1 | CPCPREVERS | VERSION NUMBER |
| 42 | (2A) CHARACTER | 8 | CPCPREJN | JOB NAME |
| 50 | (32) CHARACTER | 1 | CPCPREST | PREDECESSOR STATUS |
| 51 | (33) CHARACTER | 1 | CPCPMATC | PREDECESSOR RESOLUTION CRITERIA: BLANK (MANUALLY CHOSEN) C (CLOSEST PRECEDING) S (SAME DAY) A (ABSOLUTE INTERVAL) R (RELATIVE INTERVAL) |
| 52 | (34) SIGNED | 4 | CPCPRECPATH | --CRITICAL PREDECESSOR |
| 56 | (38) CHARACTER | 4 | * | FREE |

CPCSU - Conditional successor segment

Current plan operation conditional successor.

| Offsets | Type | Length | Name | Description |
|---------|----------------|--------|------------|---------------------------------|
| 0 | (0) STRUCTURE | 52 | CPCSUC | OPERATION CONDITIONAL SUCCESSOR |
| 0 | (0) CHARACTER | 16 | CPCSUCADI | APPLICATION ID |
| 16 | (10) CHARACTER | 10 | CPCSUCIA | INPUT ARRIVAL, |
| 16 | (10) CHARACTER | 6 | CPCSUCIAD | MODIFIED IF IA IS MODIFIED |
| 22 | (16) CHARACTER | 4 | CPCSUCIAT | ELSE ORIGINAL FROM PLAN |
| 26 | (1A) SIGNED | 4 | CPCSUCNO | OPERATION NUMBER |
| 30 | (1E) SIGNED | 4 | CPCSUC_CID | CONDITION ID |
| 34 | (22) CHARACTER | 1 | CPCSUC_C̄R | -- ON CRITICAL PATH |
| 35 | (23) UNSIGNED | 1 | CPCSUCVERS | VERSION |
| 36 | (24) CHARACTER | 8 | CPCSUCJN | JOB NAME |
| 44 | (2C) CHARACTER | 1 | CPCSUCST | SUCCESSOR STATUS |
| 45 | (2D) CHARACTER | 7 | * | |

CPEXT - Operation extended name segment

Operation extended name.

Table 186. CPEXT Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-------------|-------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 100 | CPEXT | EXTENDED INFO OF CP OPERATION |
| 0 | (0) | CHARACTER | 54 | CPEXTNAME | EXTENDED NAME |
| 54 | (36) | UNSIGNED | 1 | CPEXTVERS | RECORD VERSION NUMBER |
| 55 | (37) | CHARACTER | 1 | * | RESERVED |
| 56 | (38) | SIGNED | 4 | CPEXTOWNOP | OWNING OP NUMBER |
| 60 | (3C) | CHARACTER | 16 | CPEXTSENAME | SCHEDULING ENVIRONMENT NAME |
| 76 | (4C) | CHARACTER | 24 | * | RESERVED |

CPOP - Common segment

Current plan operation.

Note: Operation status codes:

- A** Waiting for input to arrive
 - C** Completed
 - E** Ended in error
 - I** Interrupted
 - R** Ready
 - S** Started
 - U** Undecided
 - W** Waiting
 - Y** Completed by NOERROR processing
 - *** Ready with at least one predecessor completed on a nonreporting workstation
- Minutes are the unit of duration.
 - Y and N are the indicator values.
 - SMF reader date formats are 00YYDDDF for the 20th century, and 01YYDDDF for the 21st century.

Table 187. CPOP Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|-----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 440 | CPOPCOM | CURRENT PLAN OPERATION |
| 0 | (0) | CHARACTER | 16 | CPOPADI | APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | CPOPIA | APPLICATION INPUT ARRIVAL |
| 16 | (10) | CHARACTER | 6 | CPOPIAD | MODIFIED, IF IA IS MODIFIED |
| 22 | (16) | CHARACTER | 4 | CPOPIAT | ELSE ORIGINAL FROM PLAN |
| 26 | (1A) | SIGNED | 4 | CPOPNO | OPERATION NUMBER |
| 30 | (1E) | CHARACTER | 8 | CPOPGRP | AUTHORITY GROUP |
| 38 | (26) | CHARACTER | 24 | CPOPDESC | DESCRIPTIVE TEXT |
| 62 | (3E) | CHARACTER | 8 | CPOPJBN | OP OS JOBNAME BLANK |
| 70 | (46) | CHARACTER | 8 | CPOPJES | JOB ID |
| 78 | (4E) | CHARACTER | 4 | CPOPWSN | WORKSTATION NAME |
| 82 | (52) | CHARACTER | 8 | CPOPFRM | FORM NUMBER BLANK |
| 90 | (5A) | CHARACTER | 10 | CPOPSP | PLANNED START |
| 90 | (5A) | CHARACTER | 6 | CPOPSPD | DATE BLANK |
| 96 | (60) | CHARACTER | 4 | CPOPST | TIME BLANK |
| 100 | (64) | CHARACTER | 10 | CPOPPE | PLANNED END |
| 100 | (64) | CHARACTER | 6 | CPOPPEP | DATE BLANK |
| 106 | (6A) | CHARACTER | 4 | CPOPPEP | TIME BLANK |
| 110 | (6E) | CHARACTER | 10 | CPOPOI | OPERATION INPUT ARRIVAL |
| 110 | (6E) | CHARACTER | 6 | CPOPOID | DATE BLANK |
| 116 | (74) | CHARACTER | 4 | CPOPOIT | TIME BLANK |
| 120 | (78) | CHARACTER | 10 | CPOPOD | OPERATION DEADLINE |
| 120 | (78) | CHARACTER | 6 | CPOPODD | DATE BLANK |
| 126 | (7E) | CHARACTER | 4 | CPOPODT | TIME BLANK |
| 130 | (82) | CHARACTER | 10 | CPOPLO | LATEST OUT FOR OP |
| 130 | (82) | CHARACTER | 6 | CPOPLOD | DATE |
| 136 | (88) | CHARACTER | 4 | CPOPLOT | TIME |
| 140 | (8C) | CHARACTER | 10 | CPOPAS | ACTUAL START |
| 140 | (8C) | CHARACTER | 6 | CPOPASD | DATE BLANK |
| 146 | (92) | CHARACTER | 4 | CPOPAST | TIME BLANK |
| 150 | (96) | CHARACTER | 10 | CPOPAA | ACTUAL ARRIVAL |
| 150 | (96) | CHARACTER | 6 | CPOPAAAD | DATE BLANK |
| 156 | (9C) | CHARACTER | 4 | CPOPAAAT | TIME BLANK |

Table 187. CPOP Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|----------|--|
| Dec | Hex | Type | Len | Name | Description |
| 160 | (A0) | CHARACTER | 10 | CPOPIS | INTERMED.START, IF INTERRUPTED |
| 160 | (A0) | CHARACTER | 6 | CPOPISD | DATE BLANK |
| 166 | (A6) | CHARACTER | 4 | CPOPIST | TIME BLANK |
| 170 | (AA) | CHARACTER | 10 | CPOPAE | ACTUAL END |
| 170 | (AA) | CHARACTER | 6 | CPOPAED | DATE BLANK |
| 176 | (B0) | CHARACTER | 4 | CPOPAET | TIME BLANK |
| 180 | (B4) | CHARACTER | 4 | CPOPEDU | ESTIMATED DURATION |
| 180 | (B4) | CHARACTER | 2 | CPOPEDH | ESTIMATED DURATION HOURS HH |
| 182 | (B6) | CHARACTER | 2 | CPOPEDM | ESTIMATED DURATION MINS MM |
| 184 | (B8) | CHARACTER | 6 | CPOPADU | ACTUAL DURATION |
| 184 | (B8) | CHARACTER | 4 | CPOPADH | EST. DURATION HRS HHHH BLANK |
| 188 | (BC) | CHARACTER | 2 | CPOPADM | EST. DURATION MINS MM BLANK |
| 190 | (BE) | CHARACTER | 1 | CPOPST | CURRENT STATUS |
| 191 | (BF) | CHARACTER | 4 | CPOPERR | ERROR CODE |
| 195 | (C3) | CHARACTER | 1 | CPOPXST | EXTENDED STATUS |
| 196 | (C4) | SIGNED | 4 | CPOP#PS | NUMBER OF PARALLEL SERVERS REQUIRED |
| 200 | (C8) | SIGNED | 4 | CPOP#R1 | WS RESOURCES REQUIRED |
| 204 | (CC) | SIGNED | 4 | CPOP#R2 | WS RESOURCES REQUIRED |
| 208 | (D0) | SIGNED | 4 | CPOPRI | PRIORITY |
| 212 | (D4) | SIGNED | 4 | CPOP#SU | NUMBER OF SUCCESSORS |
| 216 | (D8) | SIGNED | 4 | CPOP#PR | NUMBER OF PREDECESSORS |
| 220 | (DC) | SIGNED | 4 | CPOP#PC | NUMBER OF COMPLETED PREDECESSORS |
| 224 | (E0) | SIGNED | 4 | CPOP#SR | NUMBER OF SPECIAL RESOURCES |
| 228 | (E4) | SIGNED | 4 | CPOPPTT | TRANSPORT TIME IF PRED, MIN |
| 232 | (E8) | SIGNED | 4 | CPOPRDD | SMF READER DATE (00YYDDDF or 01YYDDDF) |
| 236 | (EC) | SIGNED | 4 | CPOPRDT | SMF READER TIME (1/100 SEC) |
| 240 | (F0) | CHARACTER | 1 | CPOPJCL | JOB CLASS, SYSOUT CLASS BLANK |
| 241 | (F1) | CHARACTER | 1 | CPOPAEC | AUTO ERROR COMPLETION (Y N) |
| 242 | (F2) | CHARACTER | 1 | CPOPASUB | AUTO JOB SUBMISSION(Y N) |
| 243 | (F3) | CHARACTER | 1 | CPOPAJR | AUTO HOLD/RELEASE (Y N) |
| 244 | (F4) | CHARACTER | 1 | CPOPTIME | TIME JOB (Y N) |

Table 187. CPOP Control Block (continued)

| Offsets | | | | | |
|---------|-------|-----------|-----|------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 245 | (F5) | CHARACTER | 1 | CPOPCLATE | CANCEL IF LATE (Y N) |
| 246 | (F6) | CHARACTER | 8 | CPOPMCPUP | TIME OF LAST MCP UPDATE |
| 254 | (FE) | CHARACTER | 1 | CPOPCPTH | ON CRITICAL PATH (F Y N) |
| 255 | (FF) | CHARACTER | 1 | CPOPLATE | LATEST OUT PASSED (Y) |
| 256 | (100) | CHARACTER | 1 | CPOPURG | URGENT (Y) |
| 257 | (101) | CHARACTER | 1 | CPOPJST | JOB STATUS (H Q N) |
| 258 | (102) | CHARACTER | 1 | CPOPPREP | JCL PREPARATION OP. (Y N) |
| 259 | (103) | CHARACTER | 1 | CPOPOIST | OP INSTR EXIST (Y N +) |
| 260 | (104) | SIGNED | 4 | CPOPHRC | HIGHEST OK RETURN CODE |
| 264 | (108) | UNSIGNED | 1 | CPOPVERS | VERSION NUMBER=1 |
| 264 | (108) | STRUCTURE | 16 | * | AS SELECTION ARGUMENT |
| 264 | (108) | CHARACTER | 16 | CPOPSOWN | OWNER ID |
| 265 | (109) | CHARACTER | 1 | CPOPPWTO | DEADLINE WTO Y N |
| 266 | (10A) | CHARACTER | 1 | CPOPRES | RESTARTABLE Y N <BLANK> |
| 267 | (10B) | CHARACTER | 1 | CPOPRER | REROUTABLE Y N <BLANK> |
| 268 | (10C) | CHARACTER | 1 | CPOPHRCS | HIGHEST RC SET Y N <BLANK> |
| 269 | (10D) | CHARACTER | 1 | CPOPMHLD | MANUALLY HELD OP Y N <BLANK> |
| 270 | (10E) | CHARACTER | 1 | CPOPNOP | NOPED OPERATION Y N <BLANK> |
| 271 | (10F) | CHARACTER | 1 | CPOPCATM | RESTART AND CLEANUP A=AUTOM., I=IMMED., M=MANUAL, N=NONE |
| 272 | (110) | CHARACTER | 16 | CPOPUDA | USER DATA |
| 288 | (120) | CHARACTER | 4 | CPOPCMDS | OPERATION COMMANDS |
| 288 | (120) | CHARACTER | 2 | CPOPCMD | OPERATION COMMAND |
| 290 | (122) | CHARACTER | 2 | * | RESERVED |
| 292 | (124) | CHARACTER | 1 | CPOPCSTA | CLEANUP STATUS |
| 293 | (125) | CHARACTER | 8 | CPOPWSINFO | WORKSTATION INFORMATION |
| 293 | (125) | CHARACTER | 1 | CPOPWSISET | INFO AVAILABLE Y N |
| 294 | (126) | CHARACTER | 1 | CPOPWSTYPE | TYPE G C P |
| 295 | (127) | CHARACTER | 1 | CPOPWSREP | REPORTING ATTRIBUTE A S C N |
| 296 | (128) | CHARACTER | 1 | CPOPWSSUBT | SUBTYPE JCL, STC, WTO, NONE J S W T A BLANK |
| 297 | (129) | CHARACTER | 1 | CPOPWSSTAT | STATUS A F O U <BLANK> |

Table 187. CPOP Control Block (continued)

| Offsets | | | | | |
|---------|-------|-----------|-----|---------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 298 | (12A) | CHARACTER | 1 | CPOPWSRRM | REROUTE MODE Y N |
| 299 | (12B) | CHARACTER | 2 | * | RESERVED |
| 301 | (12D) | CHARACTER | 1 | CPOPJCRT | WORKLOAD MONITOR CRITICAL JOB |
| 302 | (12E) | CHARACTER | 1 | CPOPJPOL | WORKLOAD MONITOR LATE JOB POLICY |
| 303 | (12F) | CHARACTER | 1 | CPOPDPREM | REMOVABLE BY DP |
| 304 | (130) | SIGNED | 4 | CPOPEDUI | ESTIMATED DUR. IN 100th OF SEC. |
| 308 | (134) | UNSIGNED | 4 | CPOPADUI | ACTUAL DUR.IN 100th OF SEC. |
| 312 | (138) | SIGNED | 4 | CPOPPSTI | PLAN. START TIME IN 100th OF SEC. |
| 316 | (13C) | SIGNED | 4 | CPOPPETI | PLAN. END TIME IN 100th OF SEC. |
| 320 | (140) | SIGNED | 4 | CPOPLOTI | LATEST OUT TIME IN 100th OF SEC. |
| 324 | (144) | SIGNED | 4 | CPOPASTI | ACTUAL START TIME IN 100th OF SEC. |
| 328 | (148) | SIGNED | 4 | CPOPAATI | ACTUAL ARR. TIME IN 100th OF SEC. |
| 332 | (14C) | SIGNED | 4 | CPOPISTI | INT. START TIME IN 100th OF SEC. |
| 336 | (150) | SIGNED | 4 | CPOPAETI | ACTUAL END TIME IN 100th OF SEC. |
| 340 | (154) | CHARACTER | 1 | CPOPEXPJCL | EXPANDED JCL NEEDED |
| 341 | (155) | CHARACTER | 1 | CPOPUSRSYS | USER SYSOUT NEEDED |
| 342 | (156) | CHARACTER | 8 | CPOPOCTO | OCCURRENCE TOKEN |
| 350 | (15E) | CHARACTER | 1 | CPOPMON | MONITORING FLAG |
| 351 | (15F) | CHARACTER | 1 | CPOPCENSCR | CENTRALIZED SCRIPT |
| 352 | (160) | SIGNED | 4 | CPOPNLVL | MAX NESTING LEVEL |
| 356 | (164) | CHARACTER | 1 | CPOPRECIS | Y IF CPREC SEGMENT EXISTS |
| 357 | (165) | CHARACTER | 1 | CPOPTWSJNM | RULE USED FOR JOBNAME IN SYMPHONY |
| 358 | (166) | CHARACTER | 1 | CPOPINSYM | JOB IN SYMPHONY (N S Y) |
| 359 | (167) | CHARACTER | 1 | CPOPDELAY | STARTED ON WAIT WORKSTATION (Y N) |
| 360 | (168) | SIGNED | 4 | CPOPSCRITPATH | BELONGING TO CRITICAL PATH |
| 364 | (16C) | CHARACTER | 8 | CPOPWLMCLASS | WLM SERVICE CLASS |
| 372 | (174) | CHARACTER | 1 | CPOPWAITSE | WAITING FOR SCHEDULING ENVIRONMENT (N S Y) |
| 373 | (175) | CHARACTER | 8 | CPOPVRTDEST | SUBMISSION DESTINATION |
| 381 | (17D) | CHARACTER | 8 | CPOPEXECDEST | EXECUTION DESTINATION |
| 389 | (185) | CHARACTER | 1 | CPOPCONDJRJOB | CONDITIONAL RECOVERY JOB |

Table 187. CPOP Control Block (continued)

| Offsets | | | | | |
|---------|-------|-----------|-----|--------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 390 | (186) | CHARACTER | 1 | CPOPUNEXPRC | UNEXPECTED RC (Y N) |
| 391 | (187) | CHARACTER | 1 | CPOPSHADOW | SHADOW JOB (Y N) |
| 392 | (188) | SIGNED | 4 | CPOPFTRC | FTA WS NUMERIC RC |
| 396 | (18C) | SIGNED | 4 | CPOP#CPROP | NUMBER OF CONDITIONAL PREDECESSORS |
| 400 | (190) | SIGNED | 4 | CPOP#CSUOP | NUMBER OF CONDITIONAL SUCCESSORS |
| 404 | (194) | SIGNED | 4 | CPOP#CONDTOT | NUMBER OF CONDITIONS |
| 408 | (198) | SIGNED | 4 | CPOP#COND_T | NUMBER OF TRUE CONDITIONS |
| 412 | (19C) | SIGNED | 4 | CPOP#COND_F | NUMBER OF FALSE CONDITIONS |
| 416 | (1A0) | SIGNED | 4 | CPOP#PX | NUMBER OF PREDECESSORS IN X STATUS |
| 420 | (1A4) | CHARACTER | 4 | CPOPORIGRC | ORIGINAL RETURN CODE |
| 424 | (1A8) | CHARACTER | 1 | CPOPBNDDST | BIND STATUS FOR SHADOW JOBS. POSSIBLE VALUES FOR CPOPBNDDST ARE: <ul style="list-style-type: none"> • P: BIND SENT • J: SENDING BIND • B: BIND ERROR • I: BIND OK |
| 425 | (1A9) | CHARACTER | 1 | CPOPWPEND | WAITING PENDING PREDECESSORS (Y/N) |
| 426 | (1AA) | CHARACTER | 1 | CPOPWMPEND | WAITING MANDATORY PENDING PREDECESSORS (Y/N) |
| 427 | (1AB) | CHARACTER | 1 | CPOPWMPPEND | WAITING MANDATORY OR PENDING PREDECESSORS (Y/N) |
| 428 | (1AC) | CHARACTER | 4 | * | FREE |
| 432 | (1B0) | CHARACTER | 8 | CPOPTRUNC | RUN CYCLE ASSOCIATED WITH THE DURATION AND DEADLINE |

CPOPSRU - Special resource usage segment

Current plan operation special resource use.

When retrieving information about the operations waiting for a certain resource (LIST CPOPSRU with argument LISTTYPE=WAITQ) or those having a certain resource allocated (LIST CPOPSRU with argument LISTTYPE=INUSE) the information about each operation is shown in the segment.

Table 188. CPOPSRU Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|---------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 96 | CPOPSRU | CP OPERATION, SR USAGE |
| 0 | (0) | CHARACTER | 16 | CPOPUADI | APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | CPOPUAIA | APPLICATION INPUT ARRIVAL |
| 16 | (10) | CHARACTER | 6 | CPOPUAID | MODIFIED IF IA IS MODIFIED |
| 22 | (16) | CHARACTER | 4 | CPOPUAIA | ELSE ORIGINAL FROM PLAN |
| 26 | (1A) | SIGNED | 4 | CPOPUNO | OPERATION NUMBER |
| 30 | (1E) | CHARACTER | 8 | CPOPUJBN | OP OS JOBNAME BLANK |
| 38 | (26) | CHARACTER | 4 | CPOPUWSN | WS NAME |
| 42 | (2A) | CHARACTER | 10 | CPOPULO | LATEST OUT |
| 42 | (2A) | CHARACTER | 6 | CPOPULOD | DATE, BLANK IF IN-USE LIST |
| 48 | (30) | CHARACTER | 4 | CPOPULOT | TIME, BLANK IF IN-USE LIST |
| 52 | (34) | CHARACTER | 10 | CPOPUAS | ACTUAL START |
| 52 | (34) | CHARACTER | 6 | CPOPUASD | DATE, BLANK IF WAIT QUEUE |
| 58 | (3A) | CHARACTER | 4 | CPOPUAST | TIME, BLANK IF WAIT QUEUE |
| 62 | (3E) | CHARACTER | 4 | CPOPUEU | ESTIMATED DURATION |
| 62 | (3E) | CHARACTER | 2 | CPOPUEUH | EST DUR HH |
| 64 | (40) | CHARACTER | 2 | CPOPUEUM | EST DUR MM |
| 66 | (42) | CHARACTER | 1 | CPOPUST | CURRENT STATE |
| 67 | (43) | UNSIGNED | 1 | CPOPUVRS | VERSION |
| 68 | (44) | SIGNED | 4 | CPOPUPRI | PRIORITY |
| 72 | (48) | SIGNED | 4 | CPOPUSRQ | SR QUANTITY USED/NEEDED |
| 76 | (4C) | CHARACTER | 8 | CPOPUWRS | REASON FOR WAIT FOR SR |
| 84 | (54) | CHARACTER | 1 | CPOPUSRU | SR ALLOCATION TYPE |
| 85 | (55) | CHARACTER | 3 | * | RESERVED |
| 88 | (58) | SIGNED | 4 | CPOPUEUI | ESTIMATED DUR. IN 100th OF SEC. |
| 92 | (5C) | CHARACTER | 4 | * | RESERVED |

CPPRE - Predecessor segment

Current plan operation predecessor.

Note: Y and N are the indicator values.

Table 189. CPPRE Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|---------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 60 | CPPRE | OPERATION PREDECESSOR |
| 0 | (0) | CHARACTER | 16 | CPPREADI | APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | CPPREIA | INPUT ARRIVAL |
| 16 | (10) | CHARACTER | 6 | CPPREIAD | MODIFIED IF IA IS MODIFIED |
| 22 | (16) | CHARACTER | 4 | CPPREIAT | ELSE ORIGINAL FROM PLAN |
| 26 | (1A) | SIGNED | 4 | CPPRENO | OPERATION NUMBER |
| 30 | (1E) | CHARACTER | 1 | CPPRECO | PREDECESSOR COMPLETED (Y N) |
| 31 | (1F) | CHARACTER | 1 | CPPREN R | PRED. WS WAS NONREPORTING |
| 32 | (20) | SIGNED | 4 | CPPRETT | TRANSPORT TIME |
| 36 | (24) | CHARACTER | 1 | CPPREND | PENDING PRED OCCURRENCE |
| 37 | (25) | UNSIGNED | 1 | CPPREVERS | VERSION NUMBER=1 |
| 38 | (26) | CHARACTER | 8 | CPPREJN | PREDECESSOR JOB NAME |
| 46 | (2E) | CHARACTER | 1 | CPPREST | PREDECESSOR STATUS |
| 47 | (2F) | CHARACTER | 1 | CPPMATC | PREDECESSOR RESOLUTION CRITERIA: BLANK (MANUALLY CHOSEN) C (CLOSEST PRECEDING) S (SAME DAY) A (ABSOLUTE INTERVAL) R (RELATIVE INTERVAL) |
| 48 | (30) | SIGNED | 4 | CPPRECRITPATH | PREDECESSOR OF AN OPERATION BELONGING TO A CRITICAL PATH |
| 52 | (34) | CHARACTER | 1 | CPPMANDP | Y: MANDATORY PENDING (CANNOT BE SET) |
| 53 | (35) | CHARACTER | 10 | CPPREFRIA | MANDATORY PENDING INTERVAL START DATE IN THE YYDDMMHHMM FORMAT |
| 63 | (3F) | CHARACTER | 10 | CPPRETOIA | MANDATORY PENDING INTERVAL END DATE IN THE YYDDMMHHMM FORMAT |
| 73 | (49) | CHARACTER | 7 | * | RESERVED |
| 80 | (50) | CHARACTER | 4 | * | RESERVED |

CPREND - Distributed remote job info segment

Distributed remote job info segment.

Note: Y and N are the indicator values.

Table 190. CPREND Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|---------------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 108 | CPREND | OPERATION DISTRIBUTED REMOTE JOB INFO |
| 0 | (0) | UNSIGNED | 1 | CPRDVERS | RECORD VERSION NUMBER |
| 1 | (1) | CHARACTER | 1 | CPRDCOMP | COMPLETE ON FAILED BIND (EDIT) Y N |
| 2 | (2) | CHARACTER | 2 | * | FREE |
| 4 | (4) | SIGNED | 4 | CPRDOWOP | OWNING OP NUMBER |
| 8 | (8) | CHARACTER | 16 | CPRDJSN | JOB STREAM NAME (EDIT) |
| 24 | (18) | CHARACTER | 16 | CPRDJSWS | JOB STREAM WORKSTATION (EDIT) |
| 40 | (28) | CHARACTER | 40 | CPRJOBN | JOB NAME (EDIT) |
| 80 | (50) | CHARACTER | 10 | CPRDIA | INPUT ARRIVAL (BROWSE) |
| 80 | (50) | CHARACTER | 6 | CPRDIAD | DATE BLANK |
| 86 | (56) | CHARACTER | 4 | CPRDIAT | TIME BLANK |
| 90 | (5A) | CHARACTER | 18 | * | |

CPRENTZ - z/OS remote job info segment

z/OS remote job info segment.

Note: Y and N are the indicator values.

Table 191. CPRENTZ Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|------------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 68 | CPRENTZ | OPERATION z/OS REMOTE JOB INFO |
| 0 | (0) | UNSIGNED | 1 | CPRZVERS | RECORD VERSION NUMBER |
| 1 | (1) | CHARACTER | 1 | CPRZCOMP | COMPLETE ON FAILED BIND (EDIT) Y N |
| 2 | (2) | CHARACTER | 2 | * | FREE |
| 4 | (4) | SIGNED | 4 | CPRZOWOP | OWNING OP NUMBER |
| 8 | (8) | SIGNED | 4 | CPRZOPNO | OP NUMBER (EDIT) |
| 12 | (C) | CHARACTER | 16 | CPRZOCCN | APPLICATION ID (EDIT) |
| 18 | (1C) | CHARACTER | 4 | CPRZWS | JOB WORKSTATION (BROWSE) |
| 32 | (20) | CHARACTER | 8 | CPRZJOBN | JOB NAME (BROWSE) |
| 40 | (28) | CHARACTER | 10 | CPRZIA | INPUT ARRIVAL (BROWSE) |
| 40 | (28) | CHARACTER | 6 | CPRZIAD | DATE BLANK |
| 46 | (2E) | CHARACTER | 4 | CPRZIAT | TIME BLANK |
| 50 | (32) | CHARACTER | 28 | * | |

CPSAI - Operation system automation information segment

System automation information.

Note: This segment exists for system automation operations only.

Table 192. CPSAI Control Block

| Offsets | | | | | |
|---------|-------|-----------|-----|---------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 352 | CPSAI | SYSTEM AUTOMATION INFO FOR CURRENT PLAN OPERATION |
| 0 | (0) | CHARACTER | 256 | CPSAICOMMTEXT | SYSTEM AUTOMATION OPERATION COMMAND TEXT |
| 0 | (0) | CHARACTER | 64 | CPSAICOMMTEX1 | SYSTEM AUTOMATION OPERATION COMMAND TEXT ROW1 |
| 64 | (40) | CHARACTER | 64 | CPSAICOMMTEX2 | SYSTEM AUTOMATION OPERATION COMMAND TEXT ROW2 |
| 128 | (80) | CHARACTER | 64 | CPSAICOMMTEX3 | SYSTEM AUTOMATION OPERATION COMMAND TEXT ROW3 |
| 192 | (C0) | CHARACTER | 63 | CPSAICOMMTEX4 | SYSTEM AUTOMATION OPERATION COMMAND TEXT ROW4 |
| 255 | (FF) | CHARACTER | 1 | CPSAIFILLER | RESERVED |
| 256 | (100) | CHARACTER | 8 | CPSAIAUTOOPER | SYSTEM AUTOMATION AUTOMATED FUNCTION (FOR OPERATION) |
| 264 | (108) | CHARACTER | 8 | CPSAISECELEM | SYSTEM AUTOMATION SECURITY ELEMENT |
| 272 | (110) | CHARACTER | 64 | CPSAICOMPINFO | SYSTEM AUTOMATION COMPLETION INFORMATION |
| 336 | (150) | CHARACTER | 4 | * | RESERVED |
| 340 | (154) | SIGNED | 4 | CPSAIOWNOP | OWNING OPERATION NUMBER |
| 344 | (158) | CHARACTER | 8 | * | RESERVED |
| | | | | | |

CPSUC - Successor segment

Current plan operation successor.

Note: Y and N are the indicator values.

Table 193. CPSUC Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|---------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 48 | CPSUC | OPERATION SUCCESSOR |
| 0 | (0) | CHARACTER | 16 | CPSUCADI | APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | CPSUCIA | INPUT ARRIVAL |
| | | | | | |

Table 193. CPSUC Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 16 | (10) | CHARACTER | 6 | CPSUCIAD | MODIFIED IF IA IS MODIFIED |
| 22 | (16) | CHARACTER | 4 | CPSUCIAT | ELSE ORIGINAL FROM PLAN |
| 26 | (1A) | SIGNED | 4 | CPSUCNO | OPERATION NUMBER |
| 30 | (1E) | CHARACTER | 1 | CPSUCCR | ON CRITICAL PATH (Y N) |
| 31 | (1F) | UNSIGNED | 1 | CPSUCVERS | VERSION NUMBER=1 |
| 32 | (20) | CHARACTER | 8 | CPSUCJN | SUCCESSOR JOB NAME |
| 40 | (28) | CHARACTER | 1 | CPSUCST | SUCCESSOR STATUS |
| 41 | (29) | CHARACTER | 7 | * | RESERVED |

CPSR - Special resource segment

Current plan operation special resource use.

Table 194. CPSR Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|-------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 66 | CPSR | |
| 0 | (0) | CHARACTER | 44 | CPSRN | NAME |
| 44 | (2C) | CHARACTER | 1 | CPSRU | USAGE (S=SHARED, X=EXCLUSIVE) |
| 45 | (2D) | UNSIGNED | 1 | CPSRVERS | VERSION |
| 46 | (2E) | CHARACTER | 1 | CPSRONER | ON ERROR FLAG |
| 47 | (2F) | CHARACTER | 1 | * | FREE |
| 48 | (30) | SIGNED | 4 | CPSRAMNT | QUANTITY |
| 52 | (34) | CHARACTER | 1 | CPSRAVACO | ON COMPLETE (Y N R blank) |
| 53 | (35) | CHARACTER | 13 | * | RESERVED |

Note: For CPSRAMNT, the value 0 means the total quantity of the special resource.

CPREC - Operation recovery segment

Table 195. CPREC Control Block

| Offsets | | | | | |
|---------|-----|-----------|-----|----------|--------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 158 | CPREC | OPERATION RECOVERY |
| 0 | (0) | CHARACTER | 16 | CPRECAID | APPLICATION ID |

Table 195. CPREC Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|--------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 16 | (10) | CHARACTER | 10 | CPRECIA | INPUT ARRIVAL |
| 16 | (10) | CHARACTER | 6 | CPRECIAD | MODIFIED IF IA IS MODIFIED |
| 22 | (16) | CHARACTER | 4 | CPRECIAT | ELSE ORIGINAL FROM PLAN |
| 26 | (1A) | SIGNED | 4 | CPRECNO | OPERATION NUMBER |
| 30 | (1E) | CHARACTER | 8 | CPRECJREID | ID OF RECOVERY JOB |
| 38 | (26) | CHARACTER | 4 | CPRECWSN | WORK STATION NAME OF REC JOB |
| 42 | (2A) | SIGNED | 10 | CPRECS | RECOVERY JOB START |
| 42 | (2A) | CHARACTER | 6 | CPRECSD | DATE BLANK |
| 48 | (30) | SIGNED | 4 | CPRECST | TIME SEC*100 0 |
| 52 | (34) | CHARACTER | 10 | CPRECE | RECOVERY JOB END |
| 52 | (34) | CHARACTER | 6 | CPRECE | DATE BLANK |
| 58 | (3A) | SIGNED | 4 | CPRECET | TIME SEC*100 0 |
| 62 | (3E) | CHARACTER | 1 | CPRECRJST | RECOVERY JOB STATUS |
| 63 | (3F) | CHARACTER | 1 | CPRECTYPE | RECOVERY TYPE: S - STOP C - CONTINUE R - RERUN |
| 64 | (40) | SIGNED | 4 | CPRECDUR | RECOVERY JOB DURATION |
| 68 | (44) | SIGNED | 4 | CPRECPRMPTID | RECOVERY PROMPT ID |
| 72 | (48) | CHARACTER | 64 | CPRECPRMSG | RECOVERY MESSAGE |
| 136 | (88) | CHARACTER | 1 | CPRECPRSTAT | RECOVERY PROMPT STATUS ' ' - NO REPLY 'N' - REPLY WITH N 'Y' - REPLY WITH Y |
| 137 | (89) | CHARACTER | 8 | CPRECJID | ID OF JOB TO RECOVER |
| 145 | (91) | CHARACTER | 4 | CPRECERC | RECOVERY JOB ERROR CODE |
| 149 | (95) | UNSIGNED | 1 | CPRECVERS | VERSION NUMBER |
| 150 | (96) | CHARACTER | 8 | * | RESERVED |

Current plan status (resource code CPST)

The current plan status record consists of one segment:

CPST Common segment. One must always exist.

CPST - Common segment

Current plan status.

The CPSTTURN can have one of these values:

- W** A daily plan batch job that creates a new plan sets this value when it runs.
- H** A daily plan batch job that has successfully created a new plan sets this value when the plan (the NCP data set) is finished.
- N** IBM Workload Scheduler for z/OS sets this value when it is started and no turnover is in progress.
- ' '** IBM Workload Scheduler for z/OS sets this value (a blank) after a successful turnover if a daily plan batch that set the value W ends without setting the value H.

Table 196. CPST Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|-------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 96 | CPST | CURRENT PLAN STATUS |
| 0 | (0) | UNSIGNED | 1 | CPSTVERS | VERSION NUMBER=1 |
| 1 | (1) | CHARACTER | 6 | CPSTCRD | CURRENT PLAN CREATE DATE |
| 7 | (7) | CHARACTER | 4 | CPSTCRT | CURRENT PLAN CREATE TIME |
| 11 | (B) | CHARACTER | 6 | CPSTENDD | CURRENT PLAN END DATE |
| 17 | (11) | CHARACTER | 4 | CPSTENDT | CURRENT PLAN END TIME |
| 21 | (15) | CHARACTER | 6 | CPSTBUD | LAST BACKUP DATE |
| 27 | (1B) | CHARACTER | 4 | CPSTBUT | LAST BACKUP TIME |
| 31 | (1F) | CHARACTER | 6 | CPST1ED | 1ST EVENT AFTER BACKUP DATE |
| 37 | (25) | CHARACTER | 4 | CPST1ET | 1ST EVENT AFTER BACKUP TIME |
| 41 | (29) | CHARACTER | 8 | CPST1EDTS | TIMESTAMP DATE FROM 1ST EVENT |
| 49 | (31) | CHARACTER | 8 | CPST1ETTS | TIMESTAMP TIME FROM 1ST EVENT |
| 57 | (39) | CHARACTER | 1 | CPSTTURN | TURNOVER PRODUCES NCP |
| 58 | (3A) | CHARACTER | 1 | CPSTCP | CURRENT PLAN EXIST (Y N) |
| 59 | (3B) | CHARACTER | 8 | CPSTCPDDN | CURRENT PLAN DDNAME |
| 67 | (43) | CHARACTER | 8 | CPSTJTDDN | JOB TRACKING DDNAME |
| 75 | (4B) | CHARACTER | 8 | CPSTJSDDN | JCL REPOSITORY DDNAME |
| 83 | (53) | CHARACTER | 13 | * | RESERVED |
| | | | | | |

Current plan operation user field (resource codes CPUSRF, CPUSRFELEM)

The current plan operation user field consists of one segment (CPUSRF).

CPUSRF - Operation user field segment

Current plan operation user field.

Table 197. CPUSRF Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 132 | CPUSRF | OPERATION USER FIELD |
| 0 | (0) | CHARACTER | 16 | CPUFADI | APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | CPUFIA | APPLICATION INPUT ARRIVAL |
| 16 | (10) | CHARACTER | 6 | CPUFIAD | MODIFIED IF IA IS MODIFIED |
| 22 | (16) | CHARACTER | 4 | CPUFIAT | ELSE ORIGINAL FROM PLAN |
| 26 | (1A) | SIGNED | 4 | CPUFOPNO | OPERATION NUMBER |
| 30 | (1E) | CHARACTER | 16 | CPUFNAME | USER FIELD NAME |
| 46 | (2E) | CHARACTER | 54 | CPUFVALUE | USER FIELD VALUE |
| 100 | (64) | SIGNED | 4 | CPUF#UF | NUMBER OF USER FIELDS |
| 104 | (68) | SIGNED | 1 | CPUFVERS | VERSION |
| 105 | (69) | CHARACTER | 27 | * | FREE |

Table 198. CPUSRFELEM Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-------------|-----------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 70 | CPUSRFELEM | ELEMENT (NAME, VALUE) |
| 0 | (0) | CHARACTER | 16 | CPUSRFNAME | USER FIELD NAME |
| 16 | (10) | CHARACTER | 54 | CPUSRFVALUE | USER FIELD VALUE |

Current plan workstation (resource codes CPWS, CPWSCOM)

The current plan workstation record consists of the common (CPWS) segment. One must appear as the first segment in each record. The CPWS segment can be followed by a variable number of CPIVL segments that represent the open intervals for the workstation.

CPWS - Common segment

Current plan workstation.

Workstation types:

- C Computer workstation
- P Printer workstation
- G General workstation
- R Remote engine workstation

Reporting attribute:

- A Automatic reporting

S Manual reporting start and stop
 C Manual reporting, completion only
 N Nonreporting

- The number of started operations is the number of parallel servers in use.
- Minutes are the unit of duration.
- Y and N are the indicator values.

Table 199. CPWS Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|--------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 132 | CPWS | CURRENT PLAN WORK STATION |
| 0 | (0) | CHARACTER | 4 | CPWSN | WORKSTATION NAME |
| 4 | (4) | CHARACTER | 32 | CPWSDESC | WORKSTATION DESCRIPTION |
| 36 | (24) | CHARACTER | 12 | CPWSSC | COMPLETED OPERATIONS SUMMARY |
| 36 | (24) | SIGNED | 4 | CPWSSC# | NUMBER OF COMPLETED OPS |
| 40 | (28) | SIGNED | 4 | CPWSSCE | ESTIMATED DURATION |
| 44 | (2C) | SIGNED | 4 | CPWSSCR | ACTUAL DURATION |
| 48 | (30) | CHARACTER | 12 | CPWSSI | INTERRUPTED OPERATIONS SUMMARY |
| 48 | (30) | SIGNED | 4 | CPWSSI# | NUMBER OF INTERRUPTED OPS |
| 52 | (34) | SIGNED | 4 | CPWSSIE | ESTIMATED DURATION |
| 56 | (38) | SIGNED | 4 | CPWSSIR | ACTUAL DURATION |
| 60 | (3C) | CHARACTER | 8 | CPWSSS | STARTED OPERATIONS SUMMARY |
| 60 | (3C) | SIGNED | 4 | CPWSSS# | NUMBER OF STARTED OPERATIONS |
| 64 | (40) | SIGNED | 4 | CPWSSSE | ESTIMATED DURATION |
| 68 | (44) | CHARACTER | 8 | CPWSSR | READY OPERATIONS SUMMARY |
| 68 | (44) | SIGNED | 4 | CPWSSR# | NUMBER OF READY OPERATIONS |
| 72 | (48) | SIGNED | 4 | CPWSSRE | ESTIMATED DURATION |
| 76 | (4C) | CHARACTER | 8 | CPWSSW | WAITING OPERATIONS SUMMARY |
| 76 | (4C) | SIGNED | 4 | CPWSSW# | NUMBER OF WAITING OPERATIONS |
| 80 | (50) | SIGNED | 4 | CPWSSWE | ESTIMATED DURATION |
| 84 | (54) | SIGNED | 4 | CPWSR1IU# | NUMBER OF R1 RESOURCES IN USE |
| 88 | (58) | SIGNED | 4 | CPWSR2IU# | NUMBER OF R2 RESOURCES IN USE |
| 92 | (5C) | SIGNED | 4 | CPWSIVL# | NUMBER OF OPEN INTERVALS |
| 96 | (60) | CHARACTER | 1 | CPWSTYPE | WORKSTATION TYPE (G C P R) |
| 97 | (61) | CHARACTER | 1 | CPWSREP | REPORTING ATTRIBUTE (A S C N) |
| 98 | (62) | CHARACTER | 1 | CPWSPSC | CONTROL ON PARALLEL SERVERS |
| 99 | (63) | CHARACTER | 2 | CPWSR1N | R1 RESOURCE NAME |
| 101 | (65) | CHARACTER | 1 | CPWSR1C | R1 RESOURCE USED FOR CONTROL |
| 102 | (66) | CHARACTER | 2 | CPWSR2N | R2 RESOURCE NAME |

Table 199. CPWS Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|--------------|--------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 104 | (68) | CHARACTER | 1 | CPWSR2C | R2 RESOURCE USED FOR CONTROL |
| 105 | (69) | CHARACTER | 1 | CPWSPREP | JOB SETUP ABILITY |
| 106 | (6A) | UNSIGNED | 1 | CPWSVERS | VERSION NUMBER=1 |
| 107 | (6B) | CHARACTER | 1 | CPWSSTC | STARTED TASK (Y N) |
| 108 | (6C) | CHARACTER | 1 | CPWSWTO | DEADLINE WTO (Y N) |
| 109 | (6D) | CHARACTER | 1 | CPWSSTAT | WORKSTATION STATUS (A O F) |
| 110 | (6E) | CHARACTER | 1 | CPWSRERUT | REROUTE MODE (Y N) |
| 111 | (6F) | CHARACTER | 4 | CPWSALTWS | ALTERNATE WS NAME |
| 115 | (73) | CHARACTER | 1 | CPWSTWS | FTA WS STATUS (Y N) |
| 116 | (74) | CHARACTER | 1 | CPWSLNK | LINK WS STATUS (L U) |
| 117 | (75) | CHARACTER | 1 | CPWSFLK | FULL LINKED (Y N) |
| 118 | (76) | CHARACTER | 1 | CPWSAUTO | SYSTEM AUTOMATION WORKSTATION |
| 119 | (77) | CHARACTER | 1 | CPWSVIRT | VIRTUAL WORKSTATION |
| 120 | (78) | CHARACTER | 8 | CPWSDEST | DESTINATION |
| 128 | (80) | CHARACTER | 1 | CPWSWAIT | WAIT WORKSTATION (Y N) |
| 129 | (81) | CHARACTER | 1 | CPWSFULLYACT | VIRTUAL FULLY ACTIVE (Y N) |
| 130 | (82) | CHARACTER | 1 | CPWSZCEN | Z-CENTRIC WORKSTATION (Y N) |
| 131 | (83) | CHARACTER | 1 | CPWSRETY | REMOTE ENGINE TYPE (D Z blank) |
| 132 | (84) | SIGNED | 4 | CPWSSX | SUM OF SUPPRESSED COND OP |
| 136 | (88) | CHARACTER | 8 | * | RESERVED |

CPIVL - Current plan workstation open interval segment

Workstation open interval.

Table 200. CPIVL Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|---------|-----------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 64 | CPIVL | WORKSTATION IVL |
| 0 | (0) | CHARACTER | 10 | CPIVLFR | INTERVAL START |
| 0 | (0) | CHARACTER | 6 | CPIVLFD | DATE YYMMDD |
| 6 | (6) | CHARACTER | 4 | CPIVLFT | TIME HHMM |
| 10 | (A) | CHARACTER | 10 | CPIVLTO | INTERVAL END |
| 10 | (A) | CHARACTER | 6 | CPIVLTD | DATE YYMMDD |
| 16 | (10) | CHARACTER | 4 | CPIVLTT | TIME HHMM |

Table 200. CPIVL Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|------------|---------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 20 | (14) | SIGNED | 4 | CPIVL#PS | MAX PARALLEL SERVERS |
| 24 | (18) | SIGNED | 4 | CPIVL#DPPS | PS SET BY DAILY PLANNING |
| 28 | (1C) | SIGNED | 4 | CPIVL#R1 | CURRENT R1 CAPACITY |
| 32 | (20) | SIGNED | 4 | CPIVL#DPR1 | R1 SET BY DAILY PLANNING |
| 36 | (24) | SIGNED | 4 | CPIVL#R2 | CURRENT R2 CAPACITY |
| 40 | (28) | SIGNED | 4 | CPIVL#DPR2 | R2 SET BY DAILY PLANNING |
| 44 | (2C) | UNSIGNED | 1 | CPIVLVERS | VERSION NUMBER |
| 45 | (2D) | CHARACTER | 4 | CPIVLDPAWS | DP ALTERNATE WORK STATION |
| 49 | (31) | CHARACTER | 4 | CPIVLAWS | CURRENT ALTERNATE WS |
| 53 | (35) | CHARACTER | 1 | CPIVLMOD | Y – MCP MODIFIED OR ADDED |
| 54 | (36) | CHARACTER | 1 | CPIVLDP | Y – ORIGINATES FROM WSD |
| 55 | (37) | CHARACTER | 9 | * | RESERVED |

CPOPT - workstation description record segment

Plan workstation record.

Table 201. CPOPT Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|--------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | | CPOPT | Workstation options |
| 0 | (0) | CHARACTER | 47 | CPOPTJOBUSR | Default JOBUSER |
| 47 | (2F) | CHARACTER | 1 | CPOPTJOBPWD | Default JOBPWD |
| 48 | (2E) | CHARACTER | 40 | CPOPTJOBTYPE | Default JOBTYP |
| 88 | (58) | CHARACTER | 1 | CPOPTBROKER | The workstation is a BROKER workstation |
| 89 | (59) | CHARACTER | 40 | CPOPTPOOL | Pool |
| 129 | (81) | CHARACTER | 40 | CPOPTDYNPOOL | Dynamic pool |
| 169 | (A4) | CHARACTER | 8 | | Reserved |

Note: The creation of dynamic agents, pools and dynamic pools is not supported using PIF. To perform these operations, use the Dynamic Workload Console. To install dynamic agents, run the related installation program.

Current plan virtual workstation destination (resource codes CPWSV, CPWSVCOM)

The current plan virtual workstation destination record consists of the common (CPWSV) segment. One must appear as the first segment in each record. The CPWSV segment can be followed by a variable number of CPIVVV segments that represent the open intervals for the workstation destination.

CPWSV - Common segment

Current plan virtual workstation destination.

Workstation types:

C Computer workstation

Reporting attribute:

A Automatic reporting

- The number of started operations is the number of parallel servers in use.
- Y and N are the indicator values.

Table 202. CPWSV Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|--|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 137 | CPWSV | CURRENT PLAN VIRTUAL WORK STATION |
| 0 | (0) | CHARACTER | 12 | CPWSVKEY | KEY |
| 0 | (0) | CHARACTER | 4 | CPWSVNAM | WORKSTATION NAME |
| 4 | (4) | CHARACTER | 8 | CPWSVDST | WORKSTATION DESTINATION |
| 12 | (C) | CHARACTER | 32 | CPWSVDESC | DESCRIPTION (NOT USED) |
| 44 | (2C) | CHARACTER | 12 | CPWSVSC | SUM OF COMPLETED OPERATIONS (NOT USED) |
| 44 | (2C) | SIGNED | 4 | CPWSVSC# | NUMBER (NOT USED) |
| 48 | (30) | SIGNED | 4 | CPWSVSCE | ESTIMATED DURATION (NOT USED) |
| 52 | (34) | SIGNED | 4 | CPWSVSCR | REAL DURATION (NOT USED) |
| 56 | (38) | CHARACTER | 12 | CPWSVSI | SUM OF INTERRUPTED OPERATIONS (NOT USED) |
| 56 | (38) | SIGNED | 4 | CPWSVSI# | NUMBER (NOT USED) |
| 60 | (3C) | SIGNED | 4 | CPWSVSIE | ESTIMATED DURATION (NOT USED) |
| 64 | (40) | SIGNED | 4 | CPWSVSIR | REAL DURATION (NOT USED) |
| 68 | (44) | CHARACTER | 8 | CPWSVSS | SUM OF STARTED OPERATIONS |
| 68 | (44) | SIGNED | 4 | CPWSVSS# | NUMBER |
| 72 | (48) | SIGNED | 4 | CPWSVSSE | ESTIMATED DURATION |
| 76 | (4C) | CHARACTER | 8 | CPWSVSR | SUM OF READY OPERATIONS (NOT USED) |

Table 202. CPWSV Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|------------|---------------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 76 | (4C) | SIGNED | 4 | CPWSVSR# | NUMBER (NOT USED) |
| 80 | (50) | SIGNED | 4 | CPWSVSRE | ESTIMATED DURATION (NOT USED) |
| 84 | (54) | CHARACTER | 8 | CPWSVSW | SUM OF WAITING OPERATIONS (NOT USED) |
| 88 | (58) | SIGNED | 4 | CPWSVSWE | ESTIMATED DURATION (NOT USED) |
| 92 | (5C) | SIGNED | 4 | CPWSVR1IU# | NUMBER OF RESOURCE 1 IN USE |
| 96 | (60) | SIGNED | 4 | CPWSVR2IU# | NUMBER OF RESOURCE 2 IN USE |
| 100 | (64) | SIGNED | 4 | CPWSVIVL# | NUMBER OF OPEN INTERVALS |
| 104 | (68) | CHARACTER | 1 | CPWSVTYPE | WORK STATION TYPE (C ONLY) |
| 105 | (69) | CHARACTER | 1 | CPWSVREP | REPORTING ATTRIBUTE (A ONLY) |
| 106 | (6A) | CHARACTER | 1 | CPWSVPSC | CONTROL ON PARALLELL SERVERS |
| 107 | (6B) | CHARACTER | 2 | CPWSVR1N | RESOURCE 1 NAME |
| 109 | (6D) | CHARACTER | 1 | CPWSVR1C | RESOURCE 1 USED AT CONTROL (NOT USED) |
| 110 | (6E) | CHARACTER | 2 | CPWSVR2N | RESOURCE 2 NAME |
| 112 | (70) | CHARACTER | 1 | CPWSVR2C | RESOURCE 2 USED AT CONTROL (NOT USED) |
| 113 | (71) | CHARACTER | 1 | * | JOB SETUP ABILITY (NOT USED) |
| 114 | (72) | UNSIGNED | 1 | CPWSVVERS | VERSION NUMBER=1 |
| 115 | (73) | CHARACTER | 1 | CPWSVSTC | STARTED TASK YN |
| 116 | (74) | CHARACTER | 1 | * | DEADLINE WTO Y N (NOT USED) |
| 117 | (75) | CHARACTER | 1 | CPWSVSTAT | WORK STATION STATUS A O F |
| 118 | (76) | CHARACTER | 1 | * | REROUTE MODE (NOT USED) |
| 119 | (77) | CHARACTER | 4 | * | ALTERNATE WS (NOT USED) |
| 123 | (7B) | CHARACTER | 1 | * | FTA WS STATUS Y N (NOT USED) |
| 124 | (7C) | CHARACTER | 1 | * | LINK WS STATUS L U (NOT USED) |
| 125 | (7D) | CHARACTER | 1 | * | FULL LINKED Y N (NOT USED) |
| 126 | (7E) | CHARACTER | 1 | * | SYSTEM AUTOMATION WS (NOT USED) |
| 127 | (7F) | CHARACTER | 1 | * | VIRTUAL WS (NOT USED) |
| 128 | (80) | CHARACTER | 8 | * | DESTINATION (NOT USED) |
| 136 | (88) | CHARACTER | 1 | * | WAIT WORKSTATION Y N (NOT USED) |

CPVIVL - Current plan virtual workstation destination open interval segment

Workstation open interval.

Table 203. CPVIVL Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-------------|--|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 64 | CPVIVL | VIRTUAL WORK STATION DESTINATION OPEN INTERVAL |
| 0 | (0) | CHARACTER | 10 | CPVIVLFR | INTERVAL START |
| 0 | (0) | CHARACTER | 6 | CPVIVLFD | DATE YYMMDD |
| 6 | (6) | CHARACTER | 4 | CPVIVLFT | TIME HHMM |
| 10 | (A) | CHARACTER | 10 | CPVIVLTO | INTERVAL END |
| 10 | (A) | CHARACTER | 6 | CPVIVLTD | DATE YYMMDD |
| 16 | (10) | CHARACTER | 4 | CPVIVLTT | TIME HHMM |
| 20 | (14) | SIGNED | 4 | CPVIVL#PS | MAX PARALLEL SERVERS |
| 24 | (18) | SIGNED | 4 | CPVIVL#DPPS | PARALLEL SERVERS SET AT DAILY PLANNING |
| 28 | (1C) | SIGNED | 4 | CPVIVL#R1 | CURRENT RESOURCE CAPACITY |
| 32 | (20) | SIGNED | 4 | CPVIVL#DPR1 | CAPACITY SET AT DAILY PLANNING |
| 36 | (24) | SIGNED | 4 | CPVIVL#R2 | CURRENT RESOURCE CAPACITY |
| 40 | (28) | SIGNED | 4 | CPVIVL#DPR2 | CAPACITY SET AT DAILY PLANNING |
| 44 | (2C) | UNSIGNED | 1 | CPVIVLVERS | VERSION NUMBER |
| 45 | (2D) | CHARACTER | 4 | * | FREE |
| 49 | (31) | CHARACTER | 4 | * | FREE |
| 53 | (35) | CHARACTER | 1 | CPVIVLMOD | Y - MCP MODIFIED OR ADDED |
| 54 | (36) | CHARACTER | 1 | CPVIVLDP | Y - ORIGINATES FROM WSD |
| 55 | (37) | CHARACTER | 9 | * | FREE |

Current plan special resource (resource codes CSR, CSRCOM)

A current plan special resource consists of four segments:

CSRCOM

Common segment.

CSRIVL

Special resource interval segment.

CSRIWS

Special resource interval workstation segment.

CSRDWS

Special resource default workstation segment.

CSRIVL and CSRSDWS are subsegments to CSRCOM. CSRIWS is a subsegment to CSRIVL.

Note: For a correct interpretation of the fields described as "Tod clock at last update", see "TOD fields" on page 271.

CSRCOM - Current plan resource common segment

Note:

1. Fields in CSRLSTEXT are set only at LIST requests
2. CSROVAV, blank means no overriding availability
3. CSROVQ , zero means no overriding quantity
4. For REPLACE request: Fields marked by (R) below are updated. Other fields are either the identifier, set implicitly or cannot be changed, except for the identifier their values are ignored.

Table 204. CSRCOM Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|------------|-------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 240 | CSRCOM | RESOURCE INSTANCE STRUCTURE |
| 0 | (0) | CHARACTER | 44 | CSRNAME | SPECIAL RESOURCE NAME |
| 44 | (2C) | CHARACTER | 8 | CSRRODM | RODM SETTING (N I P A) |
| 44 | (2C) | CHARACTER | 1 | CSRRODMA | AVAILABILITY |
| 45 | (2D) | CHARACTER | 1 | CSRRODMQ | QUANTITY |
| 46 | (2E) | CHARACTER | 1 | CSRRODMD | DEVIATION |
| 47 | (2F) | CHARACTER | 5 | * | RESERVED |
| 52 | (34) | CHARACTER | 8 | CSRGROUP | GROUP ID |
| 60 | (3C) | CHARACTER | 1 | CSRHIPER | DLF RESOURCE (Y N) |
| 61 | (3D) | CHARACTER | 1 | CSRUSEDFOR | (R) USED FOR (N P C B) |
| 62 | (3E) | CHARACTER | 2 | CSRONERROR | (R) ON ERROR (F FX FS K) |
| 64 | (40) | CHARACTER | 3 | * | RESERVED |
| 67 | (43) | CHARACTER | 1 | CSROVAV | (R) OVERRID AVAILABILITY(Y N) |
| 68 | (44) | SIGNED | 4 | SROVQ | (R) OVERRID QUANT, 0 IF NONE |
| 72 | (48) | SIGNED | 4 | CSRDEVI | (R) DEVIATION |

Table 204. CSRCOM Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|--------------|-----------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 76 | (4C) | SIGNED | 4 | CSRIVLNUM | NUMBER OF INTERVALS |
| 80 | (50) | SIGNED | 4 | CSRCIVLN | CURRENT INTERVAL NUMBER |
| 84 | (54) | CHARACTER | 46 | CSRDESC | DESCRIPTION |
| 130 | (82) | CHARACTER | 10 | CSRLIFTIEDAT | LIFESPAN EXPIRATION DATE AND TIME |
| 138 | (8A) | CHARACTER | 10 | * | RESERVED |
| 140 | (8C) | SIGNED | 4 | CSRDEFNWSC | NUMBER CONNECTED WORKSTATIONS |
| 144 | (90) | SIGNED | 4 | CSRDEFQUANT | (R) DEFAULT QUANTITY |
| 148 | (94) | CHARACTER | 1 | CSRDEFAVAIL | (R) DEFAULT AVAILABILITY |
| 149 | (95) | CHARACTER | 1 | CSRLIFTIEACT | LIFESPAN ACTION (Y N R) |
| 150 | (96) | CHARACTER | 1 | CSRONCOMPL | (R) ON COMPLETE (Y N R OR BLANK) |
| 151 | (97) | CHARACTER | 1 | CSRMAXTYPE | (R) MAX USAGE TYPE (Y N R) |
| 152 | (98) | CHARACTER | 8 | CSRLUSER | LAST UPDATING USER |
| 160 | (A0) | CHARACTER | 6 | CSRLDATE | DATE OF LAST UPDATE |
| 166 | (A6) | CHARACTER | 4 | CSRLTIME | TIME OF LAST UPDATE |
| 170 | (AA) | CHARACTER | 8 | CSRLUTS | TOD CLOCK LAST UPDATE |
| 178 | (B2) | UNSIGNED | 1 | CSRVER | RECORD VERSION |
| 179 | (B3) | CHARACTER | 1 | CSRACTAVAIL | ACTUAL AVAILABILITY |
| 180 | (B4) | SIGNED | 4 | CSRACTQUANT | ACTUAL QUANTITY |
| 184 | (B8) | SIGNED | 4 | CSRUSAGECNT | (R) USAGE COUNTER |
| 188 | (BC) | SIGNED | 4 | CSRMAXLIMIT | (R) MAX USAGE LIMIT |
| 192 | (C0) | CHARACTER | 48 | CSRLISTX | SET AT LIST REQUEST ONLY |
| 192 | (C0) | SIGNED | 4 | CSRXUSE | AMOUNT CURRENTLY USED EXCL |
| 196 | (C4) | SIGNED | 4 | CSRSUSE | AMOUNT CURRENTLY USED SHARED |
| 200 | (C8) | CHARACTER | 1 | CSRXALL | EXCLUSIVE USER NOW (Y N) |
| 201 | (C9) | CHARACTER | 1 | CSRSALL | SHARED USER NOW (Y N) |
| 202 | (CA) | CHARACTER | 1 | CSRWAITQ | ANY ON WAIT QUEUE (Y N) |

Table 204. CSRCOM Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|------------|-----------------------|
| Dec | Hex | Type | Len | Name | Description |
| 203 | (CB) | CHARACTER | 1 | CSRLASTM | LAST MODIFY TYPE |
| 208 | (D0) | CHARACTER | 32 | CSRCURIVL | CURRENT INTERVAL DATA |
| 208 | (D0) | CHARACTER | 6 | CSRCIDATE | DATE |
| 214 | (D6) | CHARACTER | 2 | * | RESERVED |
| 216 | (D8) | CHARACTER | 4 | CSRCIFTIME | FROM TIME |
| 220 | (DC) | CHARACTER | 4 | CSRCITTIME | TO TIME |
| 224 | (E0) | SIGNED | 4 | CSRCIQUANT | ALLOCATION CAPACITY |
| 228 | (E4) | SIGNED | 4 | CSRCIADJQ | ADJUST QUANTITY |
| 232 | (E8) | CHARACTER | 1 | CSRCIAVAIL | AVAILABLE (Y N) |
| 233 | (E9) | CHARACTER | 7 | * | RESERVED |

CSRIVL - Current plan special resource interval segment

Table 205. CSRIVL Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|------------|----------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 32 | CSRIVL | INTERVAL |
| 0 | (0) | CHARACTER | 6 | CSRIDATE | SPECIFIC DATE |
| 6 | (6) | CHARACTER | 2 | * | RESERVED |
| 8 | (8) | CHARACTER | 4 | CSRIFTIME | FROM TIME |
| 12 | (C) | CHARACTER | 4 | CSRITTIME | TO TIME |
| 16 | (10) | SIGNED | 4 | CSRIQUANT | ALLOCATABLE AMOUNT |
| 20 | (14) | SIGNED | 4 | CSRIWSCNUM | NUMBER OF CONNECTED WORKSTATIONS |
| 24 | (18) | CHARACTER | 1 | CSRIAVAIL | AVAILABLE (Y N) |
| 25 | (19) | CHARACTER | 7 | * | RESERVED |

CSRIWS - Current plan resource interval "connected" workstation

Table 206. CSRIWS Control Block

| Offsets | | | | | |
|---------|-----|-----------|-----|--------|----------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 8 | CSRIWS | CONNECTED WS SEGMENT |

Table 206. CSRIWS Control Block (continued)

| Offsets | | | | | |
|---------|-----|-----------|-----|------------|------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | CHARACTER | 4 | CSRIWSNAME | WORKSTATION NAME |
| 4 | (4) | CHARACTER | 4 | * | RESERVED |

CSR DWS - Current plan resource default "connected" workstation

Table 207. CSR DWS Control Block

| Offsets | | | | | |
|---------|-----|-----------|-----|-------------|----------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 8 | CSR DWS | CONNECTED WS SEGMENT |
| 0 | (0) | CHARACTER | 4 | CSR DWSNAME | WORKSTATION NAME |
| 4 | (4) | CHARACTER | 4 | * | RESERVED |

ETT - Event triggered tracking criteria segment

Table 208. ETT Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|--|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | BASED | 128 | ETT | ETT TRACKING CRITERIA REC |
| 0 | (0) | CHARACTER | 64 | ETTKEY | KEY |
| 0 | (0) | CHARACTER | 1 | ETTTYPE | RECORD TYPE=EVENT TYPE: 2 = JOB, 3 = RESOURCE |
| 1 | (1) | CHARACTER | 44 | ETTNAME | NAME OF TRIGGERING EVENT |
| 45 | (2D) | CHARACTER | 19 | * | RESERVED |
| 64 | (40) | CHARACTER | 2 | ETTVERS | RECORD VERSION |
| 66 | (42) | CHARACTER | 1 | * | RESERVED |
| 67 | (43) | CHARACTER | 16 | ETTAPPL | CORRESPONDING APPLICATION |
| 83 | (53) | CHARACTER | 1 | ETTJREP | JOB REPLACE: Y=YES, N=NO |
| 84 | (54) | CHARACTER | 8 | ETTLUSER | USED OF LAST UPDATED |
| 92 | (5C) | CHARACTER | 6 | ETTLDATE | DATE OF LAST UPDATE |
| 98 | (62) | CHARACTER | 4 | ETTLTIME | TIME OF LAST UPDATE |
| 102 | (66) | CHARACTER | 8 | * | RESERVED |
| 110 | (6E) | CHARACTER | 1 | ETTDEPR | DEP RESOLUTION: Y=YES, N=NO |
| 111 | (6F) | CHARACTER | 1 | ETTASSW | AVAIL STATUS: Y=YES, N=NO |

Table 208. ETT Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|---------|--------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 112 | (70) | CHARACTER | 8 | ETTLUTS | TOD CLOCK AT LAST UPDATE |
| 120 | (78) | CHARACTER | 8 | * | RESERVED |

Dates generated by run cycle rules (resource code GENDAYS)

The output of a LIST GENDAYS request includes both the original dates and the dates that come from a change in the free day rule. A set of flags provide information about the free day rule actions on the date. The output is made up by the **GNDAY** segment:

Table 209. GNDAY Control Block

| Offsets | | | | | |
|---------|-----|-----------|-----|-----------|--|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | * | GNDAY | RUN DAY GENERATED BY GENDAYS |
| 0 | (0) | CHARACTER | 6 | GNDAYDATE | GENERATED RUN DAY DATE (YYMMDD) |
| 6 | (6) | CHARACTER | 1 | GNDAYFMOB | ORIGINAL DATE: MOVED BEFORE BECAUSE OF FREE DAY RULE (Y/N) |
| 7 | (7) | CHARACTER | 1 | GNDAYFMOA | ORIGINAL DATE: MOVED AFTER BECAUSE OF FREE DAY RULE (Y/N) |
| 8 | (8) | CHARACTER | 1 | GNDAYFKEP | ORIGINAL DATE: KEPT BECAUSE OF FREE DAY RULE (Y/N) |
| 9 | (9) | CHARACTER | 1 | GNDAYFCAN | ORIGINAL DATE: CANCELLED BECAUSE OF FREE DAY RULE (Y/N) |
| 10 | (A) | CHARACTER | 1 | GNDAYFEIA | RUN ON FREE DAY - EARLY INPUT ARRIVAL TIME (Y/N) |
| 11 | (B) | CHARACTER | 1 | GNDAYFOUT | ORIGINAL DATE: MOVED OUTSIDE INTERVAL BECAUSE OF FREE DAY RULE (Y/N) |
| 12 | (C) | CHARACTER | 1 | GNDAYFREM | NEW WORK DATE: OUTSIDE INTERVAL (Y/N) |
| 13 | (D) | CHARACTER | 7 | * | RESERVED |

JCL setup variables (resource codes JCLPREP, JCLPREPA)

A JCL setup variable record (JSV) can contain these segments:

JSVC Fixed part of the promptable variables.

JSVV Variable update part of the promptable variables.

JSVC - Common segment

Common part of JCL setup for promptable variables.

Table 210. JSVC Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 35 | JSVC | PROMPTABLE VARIABLES |
| 0 | (0) | CHARACTER | 35 | JSVCCOM | IDENTIFIER |
| 0 | (0) | CHARACTER | 32 | JSVCKEY | KEY OF OPERATION |
| 0 | (0) | CHARACTER | 16 | JSVCADID | APPLICATION ID |
| 16 | (10) | CHARACTER | 6 | JSVCIAD | INPUT ARRIVAL DATE YYMMDD |
| 22 | (16) | CHARACTER | 2 | * | RESERVED |
| 24 | (18) | CHARACTER | 4 | JSVCIAT | INPUT ARRIVAL TIME HHMM |
| 28 | (1C) | SIGNED | 4 | JSVCOPNO | OPERATION NUMBER |
| 32 | (20) | SIGNED | 2 | JSVC#VARS | NUMBER OF VARIABLES |
| 34 | (22) | CHARACTER | 1 | JSVCFROM | JCL FROM JS REPOSITORY Y N |

JSVV - Variable definition segment

Update part of JCL setup for promptable variables.

Table 211. JSVV Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 53 | JSVV | PROMPTABLE VARIABLES |
| 0 | (0) | CHARACTER | 8 | JSVVNAME | VARIABLE NAME |
| 8 | (8) | CHARACTER | 44 | JSVVVALUE | VALUE SET OR DEFAULT VALUE |
| 52 | (34) | CHARACTER | 1 | JSVVTYPE | USAGE TYPE (% & ?) |

JCL variable table (resource codes JCLV, JCLVCOM)

A JCL variable table record (JCLV) can contain these segments:

JCLVC

Common part of the JCL variable table record

JCLVV

Variable definition part of the JCL variable table record

JCLVD

Dependency part of a JCL variable table record.

Note: For a correct interpretation of the fields described as "Tod clock at last update", see "TOD fields" on page 271.

JCLVC - Common segment

Identifies a JCL variable table.

Table 212. JCLVC Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 96 | JCLVC | COMMON PART |
| 0 | (0) | CHARACTER | 96 | JCLVCCOM | IDENTIFIER |
| 0 | (0) | CHARACTER | 1 | * | RESERVED |
| 1 | (1) | CHARACTER | 16 | JCLVCKEY | KEY OF RECORD TABLE |
| 1 | (1) | CHARACTER | 16 | JCLVCTAB | JCL VARIABLE TABLE ID |
| 17 | (11) | CHARACTER | 1 | * | RESERVED |
| 18 | (12) | CHARACTER | 8 | JCLVCLU | LAST UPDATING USER |
| 26 | (1A) | CHARACTER | 4 | JCLVCLT | LAST UPDATE TIME HHMM |
| 30 | (1E) | CHARACTER | 6 | JCLVCLD | LAST UPDATE DATE YYMMDD |
| 36 | (24) | SIGNED | 2 | JCLVC#V | NUMBER OF VARIABLES IN TABLE |
| 38 | (26) | CHARACTER | 24 | JCLVCDSC | DESCRIPTION |
| 62 | (3E) | CHARACTER | 16 | JCLVCOWN | OWNER ID |
| 78 | (4E) | CHARACTER | 2 | * | RESERVED |
| 80 | (50) | CHARACTER | 8 | JCLVCLUTS | TOD CLOCK AT LAST UPDATE |
| 88 | (58) | CHARACTER | 8 | * | RESERVED |

JCLVV - Variable definition segment

Defines a JCL variable.

Table 213. JCLVV Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 464 | JCLVV | JCL VARIABLE DEFINITIONS |
| 0 | (0) | CHARACTER | 8 | JCLVVVAR | JCL VARIABLE NAME |
| 8 | (8) | CHARACTER | 44 | JCLVVDFL | JCL VARIABLE DEF VALUE |
| 52 | (34) | CHARACTER | 1 | JCLVVSTP | PROMPT SETUP SUBMIT |
| 53 | (35) | CHARACTER | 1 | JCLVVUC | UPPER CASE (Y N) |
| 54 | (36) | SIGNED | 2 | JCLVVLG | VALUE LENGTH |
| 56 | (38) | CHARACTER | 7 | JCLVV TYP | VERIFICATION TYPE |
| 63 | (3F) | CHARACTER | 8 | JCLVVEX | SUBSTITUTION EXIT NAME |
| 71 | (47) | CHARACTER | 1 | JCLVVINP | INPUT REQUIRED |
| 72 | (48) | SIGNED | 2 | JCLVVPOS | REPLACE POSITION IN JCL DATA |

Table 213. JCLVV Control Block (continued)

| Offsets | | | | | |
|---------|-------|-----------|-----|-----------|----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 74 | (4A) | CHARACTER | 1 | JCLVVNUM | NUMERIC |
| 75 | (4B) | CHARACTER | 2 | JCLVVCMP | COMPARISON OPERATOR |
| 77 | (4D) | CHARACTER | 44 | JCLVVPAT | VALIDATION PATTERN |
| 121 | (79) | CHARACTER | 102 | JCLVVVLD | VALID VALUES |
| 223 | (DF) | CHARACTER | 204 | JCLVVTEXT | DIALOG TEXT |
| 427 | (1AB) | CHARACTER | 20 | JCLVVDES | DESCRIPTION |
| 447 | (1BF) | CHARACTER | 1 | * | RESERVED |
| 448 | (1C0) | SIGNED | 2 | JCLVVNRP | NUMBER OF DEPENDENT VALUES |
| 450 | (1C2) | CHARACTER | 8 | JCLVVIND | INDEPENDENT VARIABLE NAME |
| 458 | (1CA) | CHARACTER | 2 | JCLVVVER | RECORD VERSION NUMBER=1 |
| 460 | (1CC) | CHARACTER | 2 | JCLVVSUS | SUBSTRING START POSITION |
| 462 | (1CE) | CHARACTER | 2 | JCLVVSUL | SUBSTRING LENGTH |

Note: JCLVVVLD is 2 lines each of 51 characters. If values continue to the second line, the first line must end with a comma.

JCLVD - Dependency segment

Defines dependencies for a JCL variable.

Table 214. JCLVD Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|---------|---------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 88 | JCLVD | DEPENDENCY VALUES |
| 0 | (0) | CHARACTER | 44 | JCLVDIV | VALUE OF SETTING VARIABLE |
| 44 | (2C) | CHARACTER | 44 | JCLVDDV | OVERRIDE VALUE FOR DEPEND |

Job control language (resource codes JS, JSCOM)

A job control language record consists of only one segment, but there are two forms to choose from:

JSCOM

Job control language segment excluding JCL lines.

JS

Job control language segment including JCL lines. The text that starts at field JST is included.

JS - Job control language segment

Description of the JCL of an operation.

Status can be:

- S** Submitted.
- T** Temporarily saved.
- V** Saved.
- C** Complete.
- b** Blank. The JCL was not retrieved from the JS data set.

Last updating function can be:

- L** LTP
- W** WSD
- R** RL
- S** Submit
- M** MCP
- P** PIF

Table 215. JS Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|---------|----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 96 | JS | JCL OF AN OPERATION |
| 0 | (0) | CHARACTER | 30 | JSKEY | KEY |
| 0 | (0) | CHARACTER | 16 | JSADID | APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | JSIA | OCCURRENCE INPUT ARRIVAL |
| 16 | (10) | CHARACTER | 6 | JSIAD | DATE |
| 22 | (16) | CHARACTER | 4 | JSIAT | TIME |
| 26 | (1A) | SIGNED | 4 | JSOPNO | OPERATION NUMBER |
| 30 | (1E) | CHARACTER | 8 | JSJOBN | JOBNAME |
| 38 | (26) | CHARACTER | 4 | JSWSN | WORKSTATION NAME |
| 42 | (2A) | CHARACTER | 1 | JSST | STATUS |
| 43 | (2B) | CHARACTER | 1 | JSUPDT | LAST UPDATING FUNCTION |
| 44 | (2C) | CHARACTER | 10 | JSLUPD | LAST UPDATED |
| 44 | (2C) | CHARACTER | 6 | JSLDATE | DATE |
| 50 | (32) | CHARACTER | 4 | JSLTIME | TIME |
| 54 | (36) | CHARACTER | 8 | JSLUSER | USERID OF LAST UPDATER |
| 62 | (3E) | UNSIGNED | 1 | JSVERS | RECORD VERSION NUMBER=1 |
| 63 | (3F) | CHARACTER | 1 | * | RESERVED |
| 64 | (40) | SIGNED | 4 | JSLINES | NUMBER OF TEXT ROWS |
| 68 | (44) | CHARACTER | 1 | JSJFROM | JCL FROM JS REPOSITORY Y N |
| 69 | (45) | CHARACTER | 27 | * | RESERVED |

Table 215. JS Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|------|--|
| Dec | Hex | Type | Len | Name | Description |
| 96 | (60) | CHARACTER | | JST | START OF TEXT ROWS. THE LENGTH OF EACH ROW IS 80 CHARACTERS. |

Job log (resource code JLCOM)

A job log record (JLC) consists of one segment:

JLCOM

Common segment.

JLCOM - Common segment

Common part of job log.

Table 216. JLCOM Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 64 | JLCOM | JOBLOG OF AN OPERATION |
| 0 | (0) | CHARACTER | 30 | JLKEY | KEY |
| 0 | (0) | CHARACTER | 16 | JLADID | APPLICATION ID |
| 16 | (10) | CHARACTER | 10 | JLIA | OCC. INPUT ARRIVAL YMMDDHHMM |
| 16 | (10) | CHARACTER | 6 | JLIAD | INPUT ARRIVAL DATE YYMMDD |
| 22 | (16) | CHARACTER | 4 | JLIAT | INPUT ARRIVAL TIME HHMM |
| 26 | (1A) | SIGNED | 4 | JLOPNO | OPERATION NUMBER |
| 30 | (1E) | CHARACTER | 8 | JLJOBNAME | JOB NAME |
| 38 | (26) | CHARACTER | 4 | JLWSN | WORKSTATION NAME |
| 42 | (2A) | CHARACTER | 8 | JLJOBID | JES JOB NUMBER |
| 50 | (32) | CHARACTER | 14 | * | RESERVED |

Long-term plan occurrence (resource codes LTOC, LTOCCOM)

Each LTP occurrence can contain these segments:

LTOC Common segment. Only one must always exist.

LTOP Operation segment.

LTCPRE

Conditional predecessor segment.

LTCSUC

Conditional successor segment.

LTPRE

Predecessor segment.

LTSUC

Successor segment.

LTEXT

External run cycle group for variable duration and deadline.

LTOC - Common segment

LTP occurrence.

Notes:

1. Minutes are the unit of duration.
2. Y and N are the indicator values.

Table 217. LTOC Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|---------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 162 | LTOC | LONG-TERM PLAN OCCURRENCE |
| 0 | (0) | CHARACTER | 26 | LTOCKEY | OCCURRENCE IDENTIFIER |
| 0 | (0) | CHARACTER | 6 | LTOCIAD | RUN DATE |
| 6 | (6) | CHARACTER | 16 | LTOCADI | APPLICATION ID |
| 22 | (16) | CHARACTER | 4 | LTOCIAT | INPUT ARRIVAL TIME |
| 26 | (1A) | CHARACTER | 10 | LTOCIAO | ORIGINAL INPUT ARRIVAL |
| 26 | (1A) | CHARACTER | 6 | LTOCIAOD | DATE |
| 32 | (20) | CHARACTER | 4 | LTOCIAOT | TIME |
| 36 | (24) | CHARACTER | 10 | LTOCDL | DEADLINE |
| 36 | (24) | CHARACTER | 6 | LTOCDLD | DATE |
| 42 | (2A) | CHARACTER | 4 | LTOCDLT | TIME |
| 46 | (2E) | CHARACTER | 8 | LTOCGRP | AUTHORITY GROUP |
| 54 | (36) | CHARACTER | 16 | LTOCOID | OWNER ID |
| 70 | (46) | CHARACTER | 4 | LTOCERR | OCCURRENCE ERROR CODE |
| 74 | (4A) | CHARACTER | 1 | LTOCRDST | RUN DAY STATUS W I F |
| 75 | (4B) | UNSIGNED | 1 | LTOCVERS | VERSION NUMBER=1 |
| 76 | (4C) | SIGNED | 4 | LTOCPRI | PRIORITY |
| 80 | (50) | SIGNED | 4 | LTOC#PRE | NUMBER OF EXTERNAL PREDECESSORS |
| 84 | (54) | SIGNED | 4 | LTOC#SUC | NUMBER OF EXTERNAL SUCCESSORS |
| 88 | (58) | SIGNED | 4 | LTOC#OP | NUMBER OF CHANGED OPERATIONS |
| 92 | (5C) | CHARACTER | 1 | LTOCDEL | DELETED ONLINE |
| 93 | (5D) | CHARACTER | 1 | LTOCADD | ADDED TO LTP |
| 94 | (5E) | CHARACTER | 1 | LTOCMOD | MODIFIED IN LTP |
| 95 | (5F) | CHARACTER | 1 | LTOCMOV | RUN DATE OR TIME MODIFIED |
| 96 | (60) | CHARACTER | 1 | LTOCDEPM | EXTERNAL DEPENDENCY MODIFIED |
| 97 | (61) | CHARACTER | 1 | LTOCCOMP | COMPLETED BY JOB TRACKING |
| 98 | (62) | CHARACTER | 1 | LTOCMOVO | MOVED BECAUSE OF OPTIONAL RULE |

Table 217. LTOC Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|---|
| Dec | Hex | Type | Len | Name | Description |
| 99 | (63) | CHARACTER | 16 | LTOJVT | JCL VARIABLE TABLE |
| 115 | (73) | CHARACTER | 16 | LTGROUPID | GROUP DEFINITION ID |
| 131 | (83) | CHARACTER | 16 | LTOCCAL | CALENDAR NAME |
| 147 | (93) | CHARACTER | 1 | * | RESERVED |
| 148 | (94) | CHARACTER | 4 | LTOC#CPRE | NUMBER OF CONDITIONAL PREDECESSORS |
| 152 | (98) | CHARACTER | 4 | LTOC#CSUC | NUMBER OF CONDITIONAL SUCCESSORS |
| 156 | (9C) | CHARACTER | 4 | * | UNUSED |
| 158 | (9E) | SIGNED | 4 | LTOC#MAND | NUMBER OF MANDATORY PENDING PREDECESSORS |
| 162 | (A2) | CHARACTER | 8 | LTOCRUNC | RUN CYCLE THAT GENERATED THE OCCURRENCE |
| 170 | (AA) | SIGNED | 8 | LTOCRUNN | NUMBER OF EXTERNAL RUN CYCLE GROUPS FOR VDD |

LTOP - Operation segment

LTP changed operation.

Note: Y and N are the indicator values.

Table 218. LTOP Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|-------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 64 | LTOP | CHANGED OPERATION |
| 0 | (0) | CHARACTER | 4 | LTOPWSN | WORKSTATION NAME |
| 4 | (4) | SIGNED | 4 | LTOPNO | OPERATION NUMBER |
| 8 | (8) | CHARACTER | 10 | LTOPOI | OPERATION INPUT ARRIVAL |
| 8 | (8) | CHARACTER | 6 | LTOPOID | DATE BLANK |
| 14 | (E) | CHARACTER | 4 | LTOPOIT | TIME BLANK |
| 18 | (12) | CHARACTER | 10 | LTOPOD | OPERATION DEADLINE |
| 18 | (12) | CHARACTER | 6 | LTOPODD | DATE BLANK |
| 24 | (18) | CHARACTER | 4 | LTOPODT | TIME BLANK |
| 28 | (1C) | CHARACTER | 24 | LTOPDESC | OPERATION TEXT |
| 52 | (34) | UNSIGNED | 1 | LTOPVERS | VERSION NUMBER=1 |
| 53 | (35) | CHARACTER | 11 | * | RESERVED |

LTCPRE- Conditional predecessor segment.

LTP conditional predecessor.

| Offsets | Type | Length | Name | Description |
|---------|----------------|--------|-------------|------------------------------------|
| 0 | (0) STRUCTURE | 33 | LTCPRE | OCCURRENCE CONDITIONAL PREDECESSOR |
| 0 | (0) CHARACTER | 26 | LTCPREKEY | CONDITIONAL PREDECESSOR IDENTIFIER |
| 0 | (0) CHARACTER | 6 | LTCPREIAD | RUN DATE |
| 6 | (6) CHARACTER | 16 | LTCPREADI | APPLICATION ID |
| 22 | (16) CHARACTER | 4 | LTCPREIAT | INPUT ARRIVAL TIME |
| 26 | (1A) CHARACTER | 1 | LTCPREDEL | DEPENDENCY DELETED |
| 27 | (1B) CHARACTER | 1 | LTCPREPDONE | PREDECESSOR COMPLETED |
| 28 | (1C) CHARACTER | 1 | LTCPREPEND | IF Y IT IS A MANDATORY PENDING |
| 29 | (1D) CHARACTER | 1 | LTCPREMAND | REQUIRED VALUE: C,P,OR N |
| 30 | (1E) UNSIGNED | 1 | LTCPREVERS | VERSION NUMBER |
| 31 | (1F) CHARACTER | 2 | * | UNUSED |

LTCSUC- Conditional successor segment.

LTP conditional successor.

| Offsets | Type | Length | Name | Description |
|---------|----------------|--------|------------|----------------------------------|
| 0 | (0) STRUCTURE | 32 | LTCSUC | OCCURRENCE CONDITIONAL SUCCESSOR |
| 0 | (0) CHARACTER | 26 | LTCSUCKEY | CONDITIONAL SUCCESSOR IDENTIFIER |
| 0 | (0) CHARACTER | 6 | LTCSUCIAD | RUN DATE |
| 6 | (6) CHARACTER | 16 | LTCSUCADI | APPLICATION ID |
| 22 | (16) CHARACTER | 4 | LTCSUCIAT | INPUT ARRIVAL TIME |
| 26 | (1A) CHARACTER | 1 | LTCSUCDEL | DEPENDENCY DELETED |
| 27 | (1B) CHARACTER | 2 | * | UNUSED |
| 29 | (1D) UNSIGNED | 1 | LTCSUCVERS | VERSION NUMBER |
| 30 | (1E) CHARACTER | 2 | * | UNUSED |

LTPRE - Predecessor segment

LTP occurrence predecessor.

Note: Y and N are the indicator values.

Table 219. LTPRE Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|---------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 32 | LTPRE | OCCURRENCE PREDECESSOR |
| 0 | (0) | CHARACTER | 26 | LTPREKEY | PREDECESSOR IDENTIFIER |
| 0 | (0) | CHARACTER | 6 | LTPREIAD | RUN DATE |
| 6 | (6) | CHARACTER | 16 | LTPREADI | APPLICATION ID |
| 22 | (16) | CHARACTER | 4 | LTPREIAT | INPUT ARRIVAL TIME |
| 26 | (1A) | CHARACTER | 1 | LTPREDEL | DEPENDENCY DELETED |
| 27 | (1B) | CHARACTER | 1 | LTPREADD | MANUALLY ADDED |
| 28 | (1C) | CHARACTER | 1 | LTPREDONE | PREDECESSOR COMPLETED |
| 29 | (1D) | UNSIGNED | 1 | LTPREVERS | VERSION NUMBER=1 |
| 30 | (1E) | CHARACTER | 1 | LTPREPEND | Y: IS MANDATORY PENDING |
| 31 | (1F) | CHARACTER | 1 | LTPREMAND | C/P/N IS A REQUIRED VALUE |

LTSUC - Successor segment

Table 220. LTSUC Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|----------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 32 | LTSUC | OCCURRENCE SUCCESSOR |
| 0 | (0) | CHARACTER | 26 | LTSUCKEY | SUCCESSOR IDENTIFIER |
| 0 | (0) | CHARACTER | 6 | LTSUCIAD | RUN DATE |
| 6 | (6) | CHARACTER | 16 | LTSUCADI | APPLICATION ID |
| 22 | (16) | CHARACTER | 4 | LTSUCIAT | INPUT ARRIVAL TIME |
| 26 | (1A) | CHARACTER | 1 | LTSUCDEL | DEPENDENCY DELETED |
| 27 | (1B) | CHARACTER | 1 | LTSUCADD | MANUALLY ADDED |
| 28 | (1C) | UNSIGNED | 1 | LTSUCVERS | VERSION NUMBER=1 |
| 29 | (1D) | CHARACTER | 3 | * | RESERVED |
| | | | | | |

LTEXT - External run cycle group for variable duration and deadline

Table 221. LTEXT Control Block

| Offsets | | | | | |
|---------|-----|-----------|-----|------------|----------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 14 | LTEXT | EXTERNAL RUN CYCLE GROUP FOR VDD |
| 0 | (0) | CHARACTER | 6 | LTEXTOPID | OPERATION IDENTIFIER |
| 0 | (0) | CHARACTER | 4 | LTEXTOPWS | WORKSTATION |
| 4 | (4) | SIGNED | 2 | LTEXTOPNUM | OPERATION NUMBER |
| 6 | (6) | CHARACTER | 8 | LTEXTOPRG | EXTERNAL RUN CYCLE GROUP |
| 14 | (E) | UNSIGNED | 1 | LTEXTVERS | VERSION NUMBER = 1 |
| | | | | | |

Operator instruction (resource codes OI, OICOM)

The operator instruction record consists of only one segment, but there are two forms to choose from:

OICOM

Operator instruction segment excluding text.

OI Operator instruction segment including text. The text that starts at field OIT is included.

OI - Operator instruction segment

An operator instruction.

Table 222. OI Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|---------|--|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 96 | OICOM | OPERATOR INSTRUCTION |
| 0 | (0) | CHARACTER | 30 | OIKEY | KEY |
| 0 | (0) | CHARACTER | 16 | OIADID | APPLICATION ID |
| 16 | (10) | SIGNED | 4 | OIOPNO | OPERATION NUMBER |
| 20 | (14) | CHARACTER | 10 | OITO | VALID TO |
| 20 | (14) | CHARACTER | 6 | OITOD | DATE |
| 26 | (1A) | CHARACTER | 4 | OITOT | TIME |
| 30 | (1E) | CHARACTER | 10 | OIFROM | VALID FROM |
| 30 | (1E) | CHARACTER | 6 | OIFROMD | DATE |
| 36 | (24) | CHARACTER | 4 | OIFROMT | TIME |
| 40 | (28) | CHARACTER | 4 | OIWSN | WORKSTATION NAME |
| 44 | (2C) | CHARACTER | 8 | OIJOBN | JOBNAME |
| 52 | (34) | CHARACTER | 10 | OILUPD | LAST UPDATED |
| 52 | (34) | CHARACTER | 6 | OILDATE | DATE |
| 58 | (3A) | CHARACTER | 4 | OILTIME | TIME |
| 62 | (3E) | CHARACTER | 8 | OILUSER | USERID OF LAST UPDATER |
| 70 | (46) | UNSIGNED | 1 | OIVERS | RECORD VERSION NUMBER=1 |
| 71 | (47) | CHARACTER | 1 | * | RESERVED |
| 72 | (48) | SIGNED | 4 | OILINES | NUMBER OF TEXT ROWS |
| 76 | (4C) | CHARACTER | 8 | OILUTS | TOD CLOCK AT LAST UPDATE |
| 84 | (54) | CHARACTER | 12 | * | RESERVED |
| 96 | (60) | CHARACTER | | OIT | START OF TEXT ROWS. THE LENGTH OF EACH ROW IS 72 CHARACTERS. |

Period (resource codes PR, PRCOM)

A period record consists of only one segment, but there are two forms to choose from:

PRCOM

Period segment excluding origin dates and interval end dates.

PR Period segment including origin dates and interval end dates. The text that starts at field PRTAB is included.

PR - Period segment

Description of a period. Defines a program interface data area. PRTYPE can be:

- A** A cyclic period that includes both work days and free days
- W** A cyclic period that includes only work days
- N** A noncyclic period

Interval end dates are optional and follow the origin dates array. They are paired with origin dates; the first origin date with the first interval end date, and so on. If the segment contains interval end dates, they must match the number of origin dates, but they can be blank.

Table 223. PR Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|---------|----------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 96 | PRCOM | PERIOD DEFINITION |
| 0 | (0) | CHARACTER | 8 | PRKEY | UNIQUE IDENTIFIER |
| 0 | (0) | CHARACTER | 8 | PRNAME | PERIOD NAME |
| 8 | (8) | UNSIGNED | 1 | PRVERS | RECORD VERSION=1 |
| 9 | (9) | CHARACTER | 1 | PRTYPE | CYCLIC/NONCYCLIC TYPE A W N |
| 10 | (A) | CHARACTER | 30 | PRDESC | DESCRIPTION OF PERIOD |
| 40 | (28) | SIGNED | 4 | PRINTVL | INTERVAL OF CYCLIC ORIGINS |
| 44 | (2C) | SIGNED | 4 | PRORIG# | NUMBER OF ORIGIN DATES IN PERIOD |
| 48 | (30) | CHARACTER | 6 | PRLDATE | DATE LAST UPDATED |
| 54 | (36) | CHARACTER | 4 | PRLTIME | TIME LAST UPDATED |
| 58 | (3A) | CHARACTER | 8 | PRLUSER | USERID OF LAST UPDATER |
| 66 | (42) | CHARACTER | 16 | PRJVT | JCL VARIABLE TABLE |
| 82 | (52) | CHARACTER | 6 | * | RESERVED |
| 88 | (58) | CHARACTER | 8 | PRLUTS | TOD CLOCK AT LAST UPDATE |
| 96 | (60) | CHARACTER | * | PRTAB | START OF ORIGIN DATES |

Note: For a correct interpretation of the fields described as "Tod clock at last update", see "TOD fields" on page 271.

Table 224. Period Origin Dates

| Offsets | | | | | |
|---------|------|-----------|-----|--------|-----------------------|
| Dec | Hex | Type | Len | Name | Description |
| 96 | (60) | CHARACTER | * | PRTAB | START OF ORIGIN DATES |
| 96 | (60) | CHARACTER | 6 | PRORIG | ORIGIN DATE (YYMMDD) |

Table 225. Period Interval End Dates

| Offsets | | | | | |
|---------|-----|-----------|-----|----------|---|
| Dec | Hex | Type | Len | Name | Description |
| | | CHARACTER | 6 | PRIVLEND | INTERVAL END DATE (YYMMDD) (PRTAB+(PRORIG# * 6)) |

Run cycle group (resource codes RG, RGC0M)

A run cycle group record can contain these segments:

RGC0M

Common segment. Only one common segment must appear as the first segment in each record.

RGRUN

Run cycle group segment. One segment for every run cycle in the group.

RGC0M - Common segment

The common part of a run cycle group.

The reserved fields marked by an * in the name column should be treated as record data. Their value should be preserved when a record is updated and set to zero when a new segment is created.

Table 226. RGC0M Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|-----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 160 | RGC0M | COMMON SECTION OF RG |
| 0 | (0) | CHARACTER | 8 | RGKEY | KEY |
| 0 | (0) | CHARACTER | 8 | RGID | RUN CYCLE GROUP ID |
| 8 | (8) | CHARACTER | 4 | RGIAT | DEFAULT INPUT ARRIVAL TIME |
| 12 | (C) | CHARACTER | 16 | RGJVTAB | DEFAULT JCL VARIABLE TABLE |
| 28 | (1C) | CHARACTER | 16 | RGCAL | DEFAULT CALENDAR |
| 44 | (2C) | CHARACTER | 50 | RGDESC | RUN CYCLE GROUP DESCRIPTION |
| 94 | (5E) | CHARACTER | 8 | RGLUSER | USERID OF LAST UPDATER |
| 102 | (66) | CHARACTER | 6 | RGLDATE | DATE OF LAST UPDATE |
| 108 | (6C) | CHARACTER | 4 | RGLTIME | TIME OF LAST UPDATE |
| 112 | (70) | CHARACTER | 8 | RGLUTS | TOD CLOCK AT LAST UPDATE |
| 120 | (78) | UNSIGNED | 1 | RGC0MVERS | RECORD VERSION NUMBER |
| 121 | (79) | CHARACTER | 3 | * | RESERVED |
| 124 | (7C) | CHARACTER | 16 | RGOWNER | OWNER |
| 140 | (8C) | SIGNED | 4 | RGDD | DEFAULT DEADLINE DATE |
| 144 | (90) | CHARACTER | 4 | RGDT | DEFAULT DEADLINE TIME |

Table 226. RGC0M Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|------|-------------|
| Dec | Hex | Type | Len | Name | Description |
| 148 | (94) | CHARACTER | 8 | * | RESERVED |

RGRUN - Run cycle segment

Each run cycle in a run cycle group. The run cycles of a run cycle group are based on rules only. The segment contains the fixed part plus the rule definition.

Type Required input.

The type can be one of the following:

- R** Regular run cycle that identifies times and days when the application runs.
- E** Exclusion run cycle that identifies times and days when the application does NOT run. If you specify a particular day and time as an exclusion run cycle, no occurrences of the application are generated for that day and time, regardless of what is generated by a regular or normal run cycle. Run cycles are used in conjunction; exclusion run cycles are used to suppress run days generated by regular or normal run cycles.
- A** Rule-based run cycle group or subset. Applies to all the run cycles within a run cycle group or a run cycle group subset.
- D** Exclusion rule-based run cycle group or subset. Applies to all the run cycles within a run cycle group or a run cycle group subset.

Free day rule

Required input for all run cycles, which indicates how run days are treated:

- E** Free days excluded; only work days are taken into account
- 1** Free days included; run on the nearest day *before* the free day
- 2** Free days included; run on the nearest day *after* the free day
- 3** Free days included; run *on* the free day
- 4** Free days included; do *not* run at all.

Table 227. RGRUN Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|---------|-------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 160 | RGRUN | RUN CYCLE SECTION |
| 0 | (0) | CHARACTER | 8 | RGRNAME | RULE NAME |
| 8 | (8) | CHARACTER | 6 | RGRVALF | RUN CYCLE VALID-FROM |
| 14 | (E) | CHARACTER | 6 | RGRVALT | RUN CYCLE VALID-TO |
| 20 | (14) | CHARACTER | 50 | RGRDESC | RUN CYCLE DESCRIPTION |
| 70 | (46) | CHARACTER | 1 | RGRRULE | RULE FOR WORK/FREE DAYS |
| 71 | (47) | CHARACTER | 1 | RGRTYPE | TYPE (R E A D) |
| 72 | (48) | CHARACTER | 4 | RGRIAT | INPUT ARRIVAL TIME |

Table 227. RGRUN Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|----------------|--------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 76 | (4C) | UNSIGNED | 1 | RGRUNVERS | RECORD VERSION NUMBER |
| 77 | (4D) | CHARACTER | 3 | * | RESERVED |
| 80 | (50) | CHARACTER | 16 | RGRJV TAB | JCL VARIABLE TABLE |
| 96 | (60) | CHARACTER | 4 | * | RESERVED |
| 100 | (64) | SIGNED | 2 | RGRIRDLEN | RULE DEFINITION LENGTH |
| 102 | (66) | CHARACTER | 4 | RGRREPEATEVERY | REPEAT EVERY |
| 106 | (6A) | CHARACTER | 4 | RGRREPEATENDT | REPEAT END TIME |
| 110 | (6E) | CHARACTER | 8 | RGRSETID | RUN CYCLE CORRELATOR |
| 118 | (76) | CHARACTER | 16 | RGRCALENDAR | RUN CYCLE CALENDAR |
| 134 | (86) | CHARACTER | 2 | * | RESERVED |
| 136 | (88) | SIGNED | 4 | RGDD | DEADLINE DAY RELATIVE TO START |
| 140 | (8C) | CHARACTER | 4 | RGDT | DEADLINE TIME |
| 144 | (90) | CHARACTER | 16 | * | RESERVED |
| 160 | (A0) | CHARACTER | * | RGRIADALL | RULE DEFINITION |

Table 228. Rule Definition

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|---------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 160 | (A0) | STRUCTURE | * | RGRIADALL | RULE DEFINITION |
| 160 | (A0) | SIGNED | 4 | RGRULEL | RULE LENGTH (RGRULEL + RGRULET) |
| 164 | (A4) | CHARACTER | * | RGRULET | RULE TEXT |

RGRIRDLEN identifies the length of the rule definition. The RGRIADALL structure contains a fullword copy of RGRIRDLEN (RGRULEL), which is followed by the rule text. RGRULEL must specify the same length as RGRIRDLEN. You can insert comments or extra blanks when creating a rule, but these characters are not saved in the RG database.

Special resource (resource codes SR, SRCOM)

A special resource consists of four segments:

SRCOM

Common segment which is followed by the first SRIVL segment, the second SRIVL segment, and so forth.

SRIVL

Special resource interval segment.

SRIWS

Special resource interval workstation segment.

SRDWS

Special resource default workstation segment.

SRIVL and SRDWS are subsegments to SRCOM. SRIWS is a subsegment to SRIVL.

Table 229. SRCOM Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-------------|-----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | BASED | 192 | SRCOM | RESOURCE INSTANCE STRUCTURE |
| 0 | (0) | CHARACTER | 44 | SRCKEY | KEY |
| 0 | (0) | CHARACTER | 44 | SRCNAME | SPECIAL RESOURCE NAME |
| 44 | (2C) | CHARACTER | 8 | * | RESERVED |
| 52 | (34) | CHARACTER | 8 | SRCGROUP | GROUP ID |
| 60 | (3C) | CHARACTER | 1 | SRCHIPER | DLF RESOURCE (Y N) |
| 61 | (3D) | CHARACTER | 1 | SRCUSEDFOR | USED FOR (N/P/C/B) |
| 62 | (3E) | CHARACTER | 2 | SRCONERROR | ON ERROR OPTION |
| 64 | (40) | SIGNED | 4 | SRCIVLNUM | NUMBER OF INTERVALS |
| 68 | (44) | CHARACTER | 46 | SRCDESC | DESCRIPTION |
| 114 | (72) | CHARACTER | 1 | SRCONCOMPL | ON COMPLETE (Y/N/R/blank) |
| 115 | (73) | CHARACTER | 1 | SRCMAXTYPE | MAX LIMIT TYPE (Y/N/R) |
| 116 | (74) | SIGNED | 4 | SRCMAXLIMIT | MAX LIMIT VALUE |
| 120 | (78) | CHARACTER | 12 | * | RESERVED |
| 132 | (84) | SIGNED | 4 | SRCDEFQUANT | DEFAULT QUANTITY |
| 136 | (88) | CHARACTER | 1 | SRCDEFAVAIL | DEFAULT AVAILABILITY |
| 137 | (89) | CHARACTER | 11 | * | RESERVED |

Table 229. SRCOM Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|----------|--------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 148 | (94) | CHARACTER | 8 | SRCLUSER | LAST UPDATING USER |
| 156 | (9C) | CHARACTER | 6 | SRCLDATE | DATE OF LAST UPDATE |
| 162 | (A2) | CHARACTER | 4 | SRCLTIME | TIME OF LAST UPDATE |
| 166 | (A6) | CHARACTER | 2 | * | RESERVED |
| 168 | (A8) | CHARACTER | 8 | SRCLUTS | TOD CLOCK AT LAST UPDATE |
| 176 | (B0) | SIGNED | 1 | SRCVER | RECORD VERSION |
| 177 | (B1) | CHARACTER | 15 | * | RESERVED |

Notes:

1. Day number must be from 1 to 7 for Monday to Sunday or 8 for standard.
2. For a correct interpretation of the fields described as "Tod clock at last update", see "TOD fields" on page 271.

Table 230. SRIVL Segment

| Offsets | | | | | |
|---------|------|-----------|-----|------------|-------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | CHARACTER | 32 | SRIVL | INTERVAL |
| 0 | (0) | CHARACTER | 32 | SRIVLCOM | COMMON DATA |
| 0 | (0) | SIGNED | 4 | SRIVLDAY | DAY NUMBER |
| 4 | (4) | CHARACTER | 6 | SRIVLDATE | SPECIFIC DATE |
| 10 | (A) | CHARACTER | 2 | * | RESERVED |
| 12 | (C) | CHARACTER | 4 | SRIVLFTIME | FROM TIME |
| 16 | (10) | CHARACTER | 4 | SRIVLTIME | TO TIME |
| 20 | (14) | SIGNED | 4 | SRIVLQUANT | MAX NUMBER OF SRs TO ALLOCATE |

Table 230. SRIVL Segment (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|-------------|-------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 24 | (18) | SIGNED | 4 | SRIVLWSCNUM | NUMBER OF CONNECTED WSs |
| 28 | (1C) | CHARACTER | 1 | SRIVLAVAIL | AVAILABLE (Y N) |
| 29 | (1D) | CHARACTER | 3 | RESERVED | RESERVED |

Table 231. SRIWS Segment

| Offsets | | | | | |
|---------|-----|-----------|-----|-----------|----------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | CHARACTER | 8 | SRIWS | CONNECTED WS SEGMENT |
| 0 | (0) | CHARACTER | 4 | SRIWSNAME | WORKSTATION NAME |
| 4 | (4) | CHARACTER | 4 | * | RESERVED |

Table 232. SRDWS Segment

| Offsets | | | | | |
|---------|-----|-----------|-----|-----------|----------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | CHARACTER | 8 | SRDWS | CONNECTED WS SEGMENT |
| 0 | (0) | CHARACTER | 4 | SRDWSNAME | WORKSTATION NAME |
| 4 | (4) | CHARACTER | 4 | * | RESERVED |

Workstation description (resource codes WS, WSCOM)

The workstation description record can contain these segments:

WSCOM

Common segment. One, and only one, common segment must appear as the first segment in each record.

WSDEST

Workstation destination segment.

WSIVL

Workstation open interval segment.

WSSD

Workstation specific date segment.

WSWD

Workstation weekday segment.

WSAM

Workstation access method segment.

Note: For a correct interpretation of the fields described as "Tod clock at last update", see "TOD fields" on page 271.

WSCOM - Common segment

Common description of a workstation.

Workstation types:

G General
C Computer
P Printer
R Remote engine

Reporting attribute:

A Automatic reporting
S Manual reporting start and stop
C Manual reporting, completion only
N Nonreporting

Table 233. WSCOM Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|------------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 128 | WSCOM | |
| 0 | (0) | CHARACTER | 4 | WSKEY | UNIQUE IDENTIFIER |
| 0 | (0) | CHARACTER | 4 | WSNAME | WORKSTATION NAME |
| 4 | (4) | UNSIGNED | 1 | WSVERS | VERSION OF RECORD=1 |
| 5 | (5) | CHARACTER | 1 | WSTRYPE | WORKSTATION TYPE (G C P R) |
| 6 | (6) | CHARACTER | 1 | WSREP | REPORTING ATTRIBUTE A S C N |
| 7 | (7) | CHARACTER | 1 | WSPREP | JOBSETUP ABILITY |
| 8 | (8) | SIGNED | 4 | WSTRSPT | TRANSPORT TIME FROM PREDECESSOR WS |
| 12 | (C) | SIGNED | 4 | WSOPDUR | DEFAULT OPERATION DURATION |
| 16 | (10) | SIGNED | 4 | WSDAY# | TOTAL NUMBER OF DAYS |
| 20 | (14) | SIGNED | 4 | WSTOTIVL# | NUMBER OF OPEN INTERVALS |
| 24 | (18) | CHARACTER | 8 | WSROUT | PRINTOUT ROUTING FOR DP |
| 32 | (20) | CHARACTER | 32 | WSDISC | WORKSTATION DESCRIPTION |
| 64 | (40) | CHARACTER | 1 | WSPSJT | CONTROL ON SERVERS |
| 65 | (41) | CHARACTER | 1 | WSSPLIT | SPLITTABLE ATTRIBUTE |
| 66 | (42) | CHARACTER | 2 | WSR1NAM | WS RESOURCE NAME |
| 68 | (44) | CHARACTER | 1 | WSR1PLAN | RESOURCE USED AT PLANNING |

Table 233. WSCOM Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|----------|--|
| Dec | Hex | Type | Len | Name | Description |
| 69 | (45) | CHARACTER | 1 | WSR1CONT | RESOURCE USED FOR CONTROL |
| 70 | (46) | CHARACTER | 2 | WSR2NAM | WS RESOURCE NAME |
| 72 | (48) | CHARACTER | 1 | WSR2PLAN | RESOURCE USED AT PLANNING |
| 73 | (49) | CHARACTER | 1 | WSR2CONT | RESOURCE USED FOR CONTROL |
| 74 | (4A) | CHARACTER | 8 | WSSUDS | DESTINATION |
| 82 | (52) | CHARACTER | 6 | WSLDATE | DATE LAST UPDATED |
| 88 | (58) | CHARACTER | 4 | WSLTIME | TIME LAST UPDATED |
| 92 | (5C) | CHARACTER | 8 | WSLUSER | USERID OF LAST UPDATER |
| 100 | (64) | CHARACTER | 1 | WSSTC | STARTED TASK Y N |
| 101 | (65) | CHARACTER | 1 | WSWTO | WTO ABILITY Y N |
| 102 | (66) | CHARACTER | 1 | WSPSPL | PLANNING ON SERVERS Y N |
| 103 | (67) | CHARACTER | 1 | WSAUTO | SYSTEM AUTOMATION WORKSTATION |
| 104 | (68) | CHARACTER | 8 | WSLUTS | TOD CLOCK AT LAST UPDATE |
| 112 | (70) | SIGNED | 4 | WSOPDURI | DEFAULT OP. DURATION, IN 100th OF SECOND |
| 116 | (74) | CHARACTER | 1 | WSTWS | FTA WS STATUS Y N |
| 117 | (75) | CHARACTER | 1 | WSWAIT | WAIT WORKSTATION (Y N) |
| 118 | (76) | CHARACTER | 1 | WSVIRT | VIRTUAL WORKSTATION (Y N) |
| 119 | (77) | CHARACTER | 1 | WSZCENTR | Z-CENTRIC WORKSTATION (Y N) |
| 120 | (78) | SIGNED | 4 | WSDDES# | NUMBER OF DESTINATIONS |
| 124 | (7C) | CHARACTER | 1 | WSRENG | REMOTE ENGINE TYPE: Z, D OR BLANK |
| 125 | (7D) | CHARACTER | 1 | WSDYN | DYNAMIC SCHEDULING (Y N) |
| 126 | (7E) | CHARACTER | 2 | * | RESERVED |

WSDDEST – Destination segment

Workstation description: a virtual workstation destination name.

For each destination segment, the database contains a virtual workstation destination description record. WSVCOM is the corresponding segment.

Table 234. WSDDEST Control Block

| Offsets | | | | | |
|---------|-----|-----------|-----|----------|-------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 16 | WSDDEST | WORK STATION DESTINATION |
| 0 | (0) | CHARACTER | 8 | WSDVDEST | WORK STATION DESTINATION NAME |

Table 234. WSDEST Control Block (continued)

| Offsets | | | | | |
|---------|-----|-----------|-----|------|-------------|
| Dec | Hex | Type | Len | Name | Description |
| 8 | (8) | CHARACTER | 8 | * | FREE |

WSIVL - Open interval segment

Workstation description: an open interval.

Table 235. WSIVL Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 32 | WSIVL | |
| 0 | (0) | CHARACTER | 4 | WSIVLS | START TIME OF INTERVAL |
| 4 | (4) | CHARACTER | 4 | WSIVLE | END TIME OF INTERVAL |
| 8 | (8) | SIGNED | 4 | WSIVLPS# | NUMBER OF PARALLEL SERVERS |
| 12 | (C) | SIGNED | 4 | WSIVLR1# | R1 CAPACITY |
| 16 | (10) | SIGNED | 4 | WSIVLR2# | R2 CAPACITY |
| 20 | (14) | CHARACTER | 4 | WSIVLAWS | ALTERNATE WORKSTATION NAME |
| 24 | (18) | CHARACTER | 8 | * | RESERVED |

WSSD - Specific date segment

Workstation description: a specific date.

Table 236. WSSD Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|--------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 48 | WSSD | |
| 0 | (0) | CHARACTER | 6 | WSSDDATE | SPECIFIC DATE |
| 6 | (6) | CHARACTER | 2 | * | RESERVED |
| 8 | (8) | CHARACTER | 24 | WSSDDESC | DESCRIPTION OF THE DATE |
| 32 | (20) | SIGNED | 4 | WSSDIVL# | NUMBER OF OPEN INTERVALS |
| 36 | (24) | CHARACTER | 12 | * | RESERVED |

WSWD - Weekday segment

Workstation description: a weekday.

Weekday can be:
 MONDAY
 TUESDAY

WEDNESDAY
 THURSDAY
 FRIDAY
 SATURDAY
 SUNDAY
 STANDARD

Note: WEDNESDAY is actually stored as WEDNESDA.

Table 237. WSWD Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|--------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 48 | WSWD | |
| 0 | (0) | CHARACTER | 8 | WSWDDAY | WEEK DAY |
| 8 | (8) | CHARACTER | 24 | WSWDDESC | DESCRIPTION OF THE DAY |
| 32 | (20) | SIGNED | 4 | WSWDDIVL# | NUMBER OF OPEN INTERVALS |
| 36 | (24) | CHARACTER | 12 | * | RESERVED |

WSAM - Workstation access method segment

Workstation access method.

Note: The Workstation access method segment is no longer supported. If you specify this segment in a workstation description record, the segment is ignored.

Table 238. WSAM Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|----------|--------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 80 | WSAM | |
| 0 | (0) | CHARACTER | 12 | WSAMACC | ACCESS METHOD NAME |
| 12 | (C) | CHARACTER | 52 | WSAMADDR | NODE ADDRESS |
| 64 | (40) | SIGNED | 4 | WSAMPORT | PORT NUMBER |
| 68 | (44) | CHARACTER | 12 | * | RESERVED |

WSOPT - workstation description record segment

Workstation description record.

Table 239. WSOPT Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|--------------|---|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | | WSOPT | Workstation options |
| 0 | (0) | CHARACTER | 47 | WSOPTJOBUSR | Default JOBUSER |
| 47 | (2F) | CHARACTER | 1 | WSOPTJOBPWD | Default JOBPWD |
| 48 | (2E) | CHARACTER | 40 | WSOPTJOBTYPE | Default JOBTYP |
| 88 | (58) | CHARACTER | 1 | WSOPTBROKER | The workstation is a BROKER workstation |
| 89 | (59) | CHARACTER | 40 | WSOPTPOOL | Pool |
| 129 | (81) | CHARACTER | 40 | WSOPTDYNPOOL | Dynamic pool |
| 169 | (44) | CHARACTER | 8 | | Reserved |

Note: The creation of dynamic agents, pools and dynamic pools is not supported using PIF. To perform these operations, use the Dynamic Workload Console. To install dynamic agents, run the related installation program.

Virtual workstation destination description (resource codes WSV, WSVCOM)

The virtual workstation destination description record can contain these segments:

WSVCOM

Common segment. One, and only one, common segment must appear as the first segment in each record.

WSVIVL

Virtual workstation destination open interval segment.

WSVSD

Virtual workstation destination specific date segment.

WSVWD

Virtual workstation destination weekday segment.

Note:

1. For REPLACE request: you can only update fields marked by (R). Other fields are either the identifier, set implicitly, or cannot be changed.
2. For a correct interpretation of the fields described as "Tod clock at last update", see "TOD fields" on page 271.

WSVCOM - Common segment

Common description of a virtual workstation destination.

Workstation types:

C Computer

Reporting attribute:

A Automatic reporting

Table 240. WSVCOM Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|------------|--------------------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 96 | WSVCOM | |
| 0 | (0) | CHARACTER | 12 | WSVKEY | UNIQUE IDENTIFIER |
| 0 | (0) | CHARACTER | 4 | WSVNAME | WORKSTATION NAME |
| 4 | (4) | CHARACTER | 8 | WSVDESTN | WORKSTATION DESTINATION |
| 12 | (C) | UNSIGNED | 1 | WSVVERS | VERSION OF RECORD=1 |
| 13 | (D) | CHARACTER | 1 | * | WORKSTATION TYPE (NOT USED) |
| 14 | (E) | CHARACTER | 1 | * | REPORTING ATTRIBUTE (NOT USED) |
| 15 | (F) | CHARACTER | 1 | * | JOBSETUP ABILITY (NOT USED) |
| 16 | (10) | SIGNED | 4 | WSVDAY# | TOTAL NUMBER OF DAYS |
| 20 | (14) | SIGNED | 4 | WSVTOTIVL# | NUMBER OF OPEN INTERVALS |
| 24 | (18) | CHARACTER | 8 | * | PRINTOUT ROUTING FOR DP (NOT USED) |
| 32 | (20) | CHARACTER | 1 | WSVPSJT | CONTROL ON SERVERS (R) |
| 33 | (21) | CHARACTER | 1 | * | SPLITTABLE ATTRIBUTE (NOT USED) |
| 34 | (22) | CHARACTER | 2 | WSVR1NAM | WS RESOURCE NAME (R) |
| 36 | (24) | CHARACTER | 1 | WSVR1PLAN | RESOURCE USED AT PLANNING (NOT USED) |
| 37 | (25) | CHARACTER | 1 | WSVR1CONT | RESOURCE USED FOR CONTROL (R) |
| 38 | (26) | CHARACTER | 2 | WSVR2NAM | WS RESOURCE NAME (R) |
| 40 | (28) | CHARACTER | 1 | WSVR2PLAN | RESOURCE USED AT PLANNING (NOT USED) |
| 41 | (29) | CHARACTER | 1 | WSVR2CONT | RESOURCE USED FOR CONTROL (R) |
| 42 | (2A) | CHARACTER | 8 | * | DESTINATION (NOT USED) |
| 50 | (32) | CHARACTER | 6 | WSVLDATE | DATE LAST UPDATED |
| 56 | (38) | CHARACTER | 4 | WSVLTIME | TIME LAST UPDATED |
| 60 | (3C) | CHARACTER | 8 | WSVLUSER | USERID OF LAST UPDATER |
| 68 | (44) | CHARACTER | 1 | * | STARTED TASK Y N (NOT USED) |
| 69 | (45) | CHARACTER | 1 | * | WTO ABILITY Y N (NOT USED) |

Table 240. WSVCOM Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|---------|---|
| Dec | Hex | Type | Len | Name | Description |
| 70 | (46) | CHARACTER | 1 | * | PLANNING ON SERVERS Y N (NOT USED) |
| 71 | (47) | CHARACTER | 1 | * | SYSTEM AUTOMATION WORKSTATION (NOT USED) |
| 72 | (48) | CHARACTER | 8 | WSVLUTS | TOD CLOCK AT LAST UPDATE |
| 80 | (50) | SIGNED | 4 | * | DEFAULT OP. DURATION, IN 100th OF SECOND (NOT USED) |
| 84 | (54) | CHARACTER | 1 | * | FTA WS STATUS Y N (NOT USED) |
| 85 | (55) | CHARACTER | 1 | * | WAIT WORKSTATION (Y N) (NOT USED) |
| 86 | (56) | CHARACTER | 10 | * | RESERVED |

WSVIVL - Open interval segment

Workstation description: an open interval.

Table 241. WSVIVL Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-----------|----------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 32 | WSVIVL | |
| 0 | (0) | CHARACTER | 4 | WSVIVLS | START TIME OF INTERVAL |
| 4 | (4) | CHARACTER | 4 | WSVIVLE | END TIME OF INTERVAL |
| 8 | (8) | SIGNED | 4 | WSVIVLPS# | NUMBER OF PARALLEL SERVERS |
| 12 | (C) | SIGNED | 4 | WSVIVLR1# | R1 CAPACITY |
| 16 | (10) | SIGNED | 4 | WSVIVLR2# | R2 CAPACITY |
| 20 | (14) | CHARACTER | 4 | * | RESERVED |
| 24 | (18) | CHARACTER | 8 | * | RESERVED |

WSVSD - Specific date segment

Workstation description: a specific date.

Table 242. WSVSD Control Block

| Offsets | | | | | |
|---------|-----|-----------|-----|------------|-------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 48 | WSVSDD | |
| 0 | (0) | CHARACTER | 6 | WSVSDDDATE | SPECIFIC DATE |
| 6 | (6) | CHARACTER | 2 | * | RESERVED |
| 8 | (8) | CHARACTER | 24 | WSVSDDDESC | DESCRIPTION OF THE DATE |

Table 242. WSVSD Control Block (continued)

| Offsets | | | | | |
|---------|------|-----------|-----|------------|--------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 32 | (20) | SIGNED | 4 | WSVSDDIVL# | NUMBER OF OPEN INTERVALS |
| 36 | (24) | CHARACTER | 12 | * | RESERVED |

WSVWD - Weekday segment

Workstation description: a weekday.

Weekday can be:

MONDAY
 TUESDAY
 WEDNESDAY
 THURSDAY
 FRIDAY
 SATURDAY
 SUNDAY
 STANDARD

Note: WEDNESDAY is actually stored as WEDNESDA.

Table 243. WSWD Control Block

| Offsets | | | | | |
|---------|------|-----------|-----|-------------|--------------------------|
| Dec | Hex | Type | Len | Name | Description |
| 0 | (0) | STRUCTURE | 48 | WSVWDD | |
| 0 | (0) | CHARACTER | 8 | WSVWDDDAY | WEEK DAY |
| 8 | (8) | CHARACTER | 24 | WSVWDDDESC | DESCRIPTION OF THE DAY |
| 32 | (20) | SIGNED | 4 | WSVWDDDIVL# | NUMBER OF OPEN INTERVALS |
| 36 | (24) | CHARACTER | 12 | * | RESERVED |

Appendix B. API object fields

This appendix describes the field names of each API object. It also identifies the fields that you can specify in APPSEL and APPFLD sections of a buffer.

Fields in the IBM Workload Scheduler for z/OS API data dictionary are defined in one of these formats:

BIN A binary value.

CHAR
A character value.

DATE A character value, in the format YYMMDD.

TIME A character value, in the format HHMM.

DUR A character value, in the format HHMM or HHHHMM depending on the field length.

FLAG A character value, in the format Y or N.

Each APPSEL and APPFLD column has one of these values:

R Required. You must specify this field and the operator value must be EQ or =. For a GET request with a key type of OWNER, PRED, or SUCC, you must specify these fields and the operator must be EQ to ensure that there is only one possible match. When the key type is SAME, these fields are optional.

For PUT and DEL requests you must specify these fields. Also, the key type must be SAME and the operator EQ.

O Optional.

N Not supported.

Current plan status object

This option is valid for the current plan status object:

- GET request with key type SAME.

The default key type is SAME.

Table 244. CP_STATUS Object Fields

| Field | Type | Len | Description | APPSEL | APPFLD |
|----------------|------|-----|-------------------------------|--------|--------|
| CP_CREATE_DATE | DATE | 6 | Current plan creation date | O | O |
| CP_CREATE_TIME | TIME | 4 | Current plan creation time | O | O |
| CP_END_DATE | DATE | 6 | Current plan end date | O | O |
| CP_END_TIME | TIME | 4 | Current plan end time | O | O |
| BACKUP_DATE | DATE | 6 | Last backup date | O | O |
| BACKUP_TIME | TIME | 4 | Last backup time | O | O |
| FIRST_EV_DATE | DATE | 6 | First event after backup date | O | O |
| FIRST_EV_TIME | TIME | 4 | First event after backup time | O | O |

Table 244. CP_STATUS Object Fields (continued)

| Field | Type | Len | Description | APPSEL | APPFLD |
|----------------|------|-----|--|--------|--------|
| FIRST_EV_D_TS | CHAR | 8 | First event after backup date. The field format is 00YYDDDF for dates in the 20th century, and 01YYDDDF for dates in the 21st century. | O | O |
| FIRST_EV_T_TS | CHAR | 8 | First event after backup time in format HHMMSSSTH | O | O |
| TURNOVER_NCP | CHAR | 1 | Turnover in progress, Y or N | O | O |
| CP_EXIST | CHAR | 1 | Current plan exists, Y or N | O | O |
| CP_DDNAME | CHAR | 8 | Current plan ddname | O | O |
| JT_DDNAME | CHAR | 8 | Job-tracking ddname | O | O |
| JCL_REP_DDNAME | CHAR | 8 | JCL repository ddname | O | O |
| NUM_PIF_ADDS | BIN | 4 | Number of occs added by PIF | O | O |
| NUM_MCP_ADDS | BIN | 4 | Number of occs added by MCP | O | O |
| NUM_ETT_ADDS | BIN | 4 | Number of occs added by ETT | O | O |
| NUM_AR_ADDS | BIN | 4 | Number of occs added by autorec | O | O |
| NUM_OCCS | BIN | 4 | Number of occurrences | O | O |
| NUM_OPERS | BIN | 4 | Number of operations | O | O |

Current plan operation object

These options are valid for the current plan operation object:

- GET request with key type SAME, PRED, or SUCC
- PUT request with key type SAME
- DEL request with key type SAME.

The default key type is SAME.

Table 245. CP_OPERATION Object Fields

| Field | Type | Len | Description | APPSEL | APPFLD | |
|-----------------|------|-----|--|--------|--------|-----|
| | | | | | Get | Put |
| OPER_NUM | BIN | 2 | Operation number | R | O | N |
| AUTHORITY_GROUP | CHAR | 8 | Authority group | O | O | N |
| CATMGMT_STATUS | CHAR | 1 | CleanUp status: <blank> None C Completed E Ended in error I Initiated O OPInfo is available R OpInfo requested S Started W Waiting for OPInfo | O | O | N |
| APPL_ID | CHAR | 16 | Application ID | R | O | N |
| APPL_IA_DATE | DATE | 6 | Application input arrival date | R | O | N |

Table 245. CP_OPERATION Object Fields (continued)

| Field | Type | Len | Description | APPSEL | APPFLD | |
|------------------|------|-----|---|--------|--------|-----|
| | | | | | Get | Put |
| APPL_IA_TIME | TIME | 4 | Application input arrival time | R | O | N |
| OPER_TEXT | CHAR | 24 | Descriptive text for the operation | O | O | N |
| JOBNAME | CHAR | 8 | Job name | O | O | N |
| WS_NAME | CHAR | 4 | Workstation name | O | O | N |
| CLASS | CHAR | 1 | Job class or SYSOUT class value | O | O | O |
| IA_DEFAULTED | FLAG | 1 | Operation input arrival defaulted | N | O | N |
| IMM_CATMGMT_DEF | FLAG | 1 | Immediate Clean Up is defined | N | O | N |
| DEFR_CATMGMT_DEF | FLAG | 1 | Manual or automatic Clean up is defined | N | O | N |
| MANUALLY_HELD | FLAG | 1 | Manually held operation | N | O | O |
| NOP_OPER | FLAG | 1 | NOP operation | N | O | O |
| EXECUTE_OPER | FLAG | 1 | Execute requested for operation | N | O | O |
| WAIT_MAN_CATMGMT | FLAG | 1 | Always N | N | O | N |
| FORM_NUMBER | CHAR | 8 | Form number | O | O | O |
| PLAN_START_DATE | DATE | 6 | Planned start date | O | O | N |
| PLAN_START_TIME | TIME | 6 | Planned start time | O | O | N |
| PLAN_END_DATE | DATE | 6 | Planned end date | O | O | N |
| PLAN_END_TIME | TIME | 4 | Planned end time | O | O | N |
| OPER_IA_DATE | DATE | 6 | Operation input arrival date | O | O | N |
| OPER_IA_TIME | TIME | 4 | Operation input arrival time | O | O | N |
| DL_DATE | DATE | 6 | Operation deadline date | O | O | N |
| DL_TIME | TIME | 4 | Operation deadline time | O | O | N |
| LATEST_OUT_DATE | DATE | 6 | Operation latest out date | O | O | N |
| LATEST_OUT_TIME | TIME | 4 | Operation latest out time | O | O | N |
| ACT_START_DATE | DATE | 6 | Actual start date | O | O | N |
| ACT_START_TIME | TIME | 4 | Actual start time | O | O | N |
| ACT_ARRIVAL_DATE | DATE | 6 | Actual arrival date | O | O | N |
| ACT_ARRIVAL_TIME | TIME | 4 | Actual arrival time | O | O | N |
| INTER_START_DATE | DATE | 6 | Intermediate start date | O | O | N |
| INTER_START_TIME | TIME | 4 | Intermediate start time | O | O | N |
| ACT_END_DATE | DATE | 6 | Actual end date | O | O | N |
| ACT_END_TIME | TIME | 4 | Actual end time | O | O | N |
| EST_DUR | DUR | 4 | Estimated duration | O | O | O |
| ACT_DUR | DUR | 6 | Actual duration | O | O | N |

Table 245. CP_OPERATION Object Fields (continued)

| Field | Type | Len | Description | APPSEL | APPFLD | |
|------------------|------|-----|--|--------|--------|-----|
| | | | | | Get | Put |
| NUM_PAR_SERV_REQ | BIN | 2 | Number of parallel servers required | O | O | N |
| NUM_WS_R1_REQ | BIN | 2 | Number of R1 resources required | O | O | N |
| NUM_WS_R2_REQ | BIN | 2 | Number of R2 resources required | O | O | N |
| CURRENT_STATUS | CHAR | 1 | Current status of the operation: A Arriving C Completed D Deleted E Ended in error I Interrupted R Ready, all preds complete S Started U Undecided W Waiting, uncompleted preds * Ready, nonreporting pred | O | O | O |
| ERROR_CODE | CHAR | 4 | Error code | O | O | O |
| AUTO_ERROR_COMPL | CHAR | 1 | Auto error completion Y or N | O | O | N |
| PRIORITY | CHAR | 1 | Priority 1 to 9 | O | O | N |
| EXTENDED_STATUS | CHAR | 1 | Extended status: A Waiting for deferred CM C Waiting for CM to complete E Error during job submission G Started on WAIT workstation H Manually held L Time operation is late M Status set manually N NOP operation Q Job added to JES queue R Automatic error reset S Job or started task executing T Waiting for time U Submit in progress X Waiting for special resource | O | O | N |

Table 245. CP_OPERATION Object Fields (continued)

| Field | Type | Len | Description | APPSEL | APPFLD | |
|-------------------|------|-----|--|--------|--------|-----|
| | | | | | Get | Put |
| NUM_SUCC | BIN | 2 | Number of successors | O | O | N |
| NUM_PRED | BIN | 2 | Number of predecessors | O | O | N |
| NUM_DEPENDENCIES | BIN | 2 | Number of successors and predecessors | O | O | N |
| NUM_COMPL_PRED | BIN | 2 | Number of predecessors completed | O | O | N |
| NUM_SR | BIN | 2 | Number of special resources | O | O | N |
| RERUN_RECORD | FLAG | 1 | Rerun record for this operation | N | O | N |
| VALID_EXIT_PASS | FLAG | 1 | Validation exit passed | N | O | N |
| ASSUMED_COMPLETE | FLAG | 1 | Assumed completed | N | O | N |
| SPECIFY_IA | FLAG | 1 | Specified input arrival for op | N | O | N |
| SPECIFY_DL | FLAG | 1 | Specified deadline for op | N | O | N |
| AUTO_SUBMISSION | FLAG | 1 | Auto submission of job | N | O | N |
| AUTO_HOLD_REL | FLAG | 1 | Automatic hold/release | N | O | N |
| LATE_MSG_ISSUED | FLAG | 1 | Late operator message issued | N | O | N |
| JOB_SUBMITTED | FLAG | 1 | Job submitted | N | O | N |
| TIME_JOB | FLAG | 1 | Time job | N | O | N |
| PREP_WS_NOTCOMPL | FLAG | 1 | Prep op exists but is not complete | N | O | N |
| SUPPRESS_IF_LATE | FLAG | 1 | Suppress if late | N | O | N |
| HIGH_RC_USED | FLAG | 1 | High return code used | N | O | N |
| PENDING_PRED | FLAG | 1 | Pending predecessor | N | O | N |
| LONG_DUR_ISSUED | FLAG | 1 | Long duration message issued | N | O | N |
| LAST_MCP_UP_DATE | BIN | 4 | Date of last MCP update | N | O | N |
| LAST_MCP_UP_TIME | BIN | 4 | Time of last MCP update | N | O | N |
| DEPENDENCY_TYPE | CHAR | 1 | Dependency type: P Predecessor S Successor | N | O | N |
| RESTARTABLE | FLAG | 1 | Restartable operation | N | O | N |
| INSTPARAM_RESTART | FLAG | 1 | Installation default for workload restart | N | O | N |
| REROUTABLE | FLAG | 1 | Reroutable operation | N | O | N |
| INSTPARAM_REROUTE | FLAG | 1 | Installation default for workload reroute | N | O | N |
| REROUTED | FLAG | 1 | Operation rerouted | N | O | N |
| DL_WTO_WANTED | FLAG | 1 | Deadline WTO required | N | O | N |
| DL_WTO_REQ_SENT | FLAG | 1 | Deadline WTO request sent | N | O | N |

Table 245. CP_OPERATION Object Fields (continued)

| Field | Type | Len | Description | APPSEL | APPFLD | |
|------------------|------|-----|-------------------------------------|--------|--------|-----|
| | | | | | Get | Put |
| DL_WTO_REQ_PROC | FLAG | 1 | Deadline WTO request processed | N | O | N |
| HIGHRC_NOT_ERROR | BIN | 2 | Highest return code not in error | O | O | N |
| ALT_WS_NAME | CHAR | 4 | Alternate workstation name | O | O | N |
| USER_FIELD | CHAR | 16 | User field | O | O | N |
| ON_CRITICAL_PATH | CHAR | 1 | Critical path indicator, Y, N, or F | O | O | N |
| LATEST_OUT_PASS | CHAR | 1 | Latest out passed, Y or N | O | O | N |
| URGENT | CHAR | 1 | Urgent, Y or N | O | O | N |
| TRANSPORT_TIME | BIN | 4 | Transport time HHMM | O | O | N |
| APPL_TEXT | CHAR | 24 | Application text | O | O | N |
| APPL_OWNER_ID | CHAR | 16 | Application owner ID | O | O | N |
| JOB_ID | CHAR | 8 | JES job number | O | O | N |
| SMF_READER_DATE | BIN | 4 | SMF reader date | O | O | N |
| SMF_READER_TIME | BIN | 4 | SMF reader time | O | O | N |
| JOB_STATUS | CHAR | 1 | Job status H, Q, N, or blank | O | O | N |
| JCL_PREPARATION | CHAR | 1 | JCL preparation operation, Y or N | O | O | N |
| OL_EXIST | CHAR | 1 | Op instruction exists Y, N, or + | O | O | N |
| RESOURCE_USE | CHAR | 1 | Blank in OPC/ESA Release 3 | O | O | N |

Table 245. CP_OPERATION Object Fields (continued)

| Field | Type | Len | Description | APPSEL | APPFLD | |
|------------------|------|-----|---|--------|--------|-----|
| | | | | | Get | Put |
| EXTENDED_STATUS2 | CHAR | 1 | Additional status explanation: A Automatic error reset C Workstation closed D Job submission deactivated F Job submission failed H Workstation close in progress J No auto job submission L Time job is late P All parallel servers in use S Resource unavailable T Start time not reached U Work station is unlinked 1 Insufficient WS resource 1 2 Insufficient WS resource 2 | O | O | N |
| WS_TYPE | CHAR | 1 | Workstation type: 1 General 2 Computer 3 Print | O | O | N |
| WTO_WS | CHAR | 1 | WTO workstation type, Y or N | O | O | N |
| OCC_GROUP_DEF | CHAR | 16 | Occurrence group name | O | O | N |

Current plan special resource object

This option is valid for the current plan special resource object:

- GET request with key type OWNER.

The default key type is OWNER.

Table 246. CP_RESOURCE Object Fields

| Field | Type | Len | Description | APPSEL | APPFLD |
|------------------|------|-----|---|--------|--------|
| SR_NAME | CHAR | 44 | Special resource name | N | O |
| ALLOCATION_TYPE | CHAR | 1 | Allocation type (S or X) | N | O |
| AVAILABLE | FLAG | 1 | Availability indicator | N | O |
| SHR_IN_USE | FLAG | 1 | Special resource allocated - SHR | N | O |
| IN_USE_EXCLUSIVE | FLAG | 1 | Special resource allocated - EXCL | N | O |
| KEPT_AT_ERROR | FLAG | 1 | Special resource has been kept on error | N | O |

Table 246. CP_RESOURCE Object Fields (continued)

| Field | Type | Len | Description | APPSEL | APPFLD |
|----------------|------|-----|-------------------------------------|--------|--------|
| KEPT_EXCLUSIVE | FLAG | 1 | EXCL special resource kept on error | N | O |
| QUANTITY | BIN | 31 | Quantity requested by the operation | N | O |
| KEEP_ON_ERROR | CHAR | 1 | On-error indicator (Y, N, blank) | N | O |

Note:

1. Because you must identify the owning operation to retrieve special resource information, you must specify the selection fields that are mandatory for the CP_OPERATION object. No CP_OPERATION fields are returned in the receive buffer.
2. The values returned for fields ALLOCATION_TYPE, QUANTITY, and KEEP_ON_ERROR depend on the operation that is used to access the CP_RESOURCE object.

Current plan workstation object

This option is valid for the current plan workstation object:

- GET request with key type SAME.

The default key type is SAME.

Table 247. CP_WORK_STATION Object Fields

| Field | Type | Len | Description | APPSEL | APPFLD |
|------------------|------|-----|--|--------|--------|
| WS_NAME | CHAR | 4 | Workstation name | O | O |
| WS_TEXT | CHAR | 32 | Workstation description | O | O |
| NUM_COMPL | BIN | 4 | Number of completed operations | O | O |
| EST_DUR_COMPL | BIN | 4 | Estimated duration of completed operations | O | O |
| ACT_DUR_COMPL | BIN | 4 | Actual duration of completed operations | O | O |
| NUM_INTER | BIN | 4 | Number of interrupted operations | O | O |
| EST_DUR_INTER | BIN | 4 | Estimated duration of interrupted operations | O | O |
| ACT_DUR_INTER | BIN | 4 | Actual duration of interrupted operations | O | O |
| NUM_START | BIN | 4 | Number of started operations | O | O |
| EST_DUR_START | BIN | 4 | Estimated duration of started operations | O | O |
| NUM_READY | BIN | 4 | Number of ready operations | O | O |
| EST_DUR_READY | BIN | 4 | Estimated duration of ready operations | O | O |
| NUM_WAITING | BIN | 4 | Number of waiting operations | O | O |
| EST_DUR_WAITING | BIN | 4 | Estimated duration of waiting operations | O | O |
| NUM_ARRIVING | BIN | 4 | Number of arriving operations | O | O |
| NUM_NONREP_READY | BIN | 4 | Number of nonreporting ready operations | O | O |

Table 247. CP_WORK_STATION Object Fields (continued)

| Field | Type | Len | Description | APPSEL | APPFLD |
|------------------|------|-----|--|--------|--------|
| NUM_UNDECIDED | BIN | 4 | Number of undecided operations | O | O |
| NUM_ERROR | BIN | 4 | Number of error operations | O | O |
| NUM_LATE | BIN | 4 | Number of late operations | O | O |
| WS_TYPE | CHAR | 1 | Workstation type: 1 General 2 Computer 3 Print | O | O |
| REPORTING_ATTR | CHAR | 1 | Reporting attribute: 1 Automatic 2 Manual, start and complete 3 Manual, completion only 4 Nonreporting | O | O |
| R1_NAME | CHAR | 2 | R1 resource name | O | O |
| NUM_R1_IN_USE | BIN | 2 | Number of R1 resources in use | O | O |
| R1_USED_AT_CNTL | FLAG | 1 | R1 resource used at control | N | O |
| R2_NAME | CHAR | 2 | R2 resource name | O | O |
| NUM_R2_IN_USE | BIN | 2 | Number of R2 resources in use | O | O |
| R2_USED_AT_CNTL | FLAG | 1 | R2 resource used at control | N | O |
| READY_LIST_TYPE | CHAR | 1 | Ready list type | O | O |
| JOB_SETUP_ABIL | FLAG | 1 | Job setup ability | N | O |
| IVL_NOT_USED | FLAG | 1 | Interval not used at all | N | O |
| NO_PAR_SERV | FLAG | 1 | Parallel servers used for control | N | O |
| STARTED_TASK_SUP | FLAG | 1 | Started task support | N | O |
| WTO_DL_SUP | FLAG | 1 | WTO workstation | N | O |
| PENDING_OFFLINE | FLAG | 1 | Workstation is pending offline | N | O |
| T_EVENT_PENDING | FLAG | 1 | T-event pending | N | O |
| ALT_WS_VARIED | FLAG | 1 | Varied alternate workstation set | N | O |
| PREV_EVENT_DATE | BIN | 4 | Previous event date | O | O |
| PREV_EVENT_TIME | BIN | 4 | Previous event time | O | O |
| DDNAME_SUBREL_DS | CHAR | 8 | Submit/release data set ddname | O | O |
| NUM_IVL | BIN | 2 | Number of open intervals | O | O |
| MAX_NUM_EVENTS | BIN | 2 | Max number of events in 15 minutes | O | O |
| WS_STATUS | CHAR | 1 | Workstation status: A Active O Offline F Failed U Unknown | O | O |
| DEF_TRANS_TIME | BIN | 2 | Transport time default | O | O |
| OFFLINE_DATE | BIN | 4 | Offline date | O | O |
| OFFLINE_TIME | BIN | 4 | Offline time | O | O |
| CURRENT_ALT_WS | CHAR | 4 | Alternate workstation name | O | O |

Current plan open interval object

This option is valid for the current plan open interval object:

- GET request with key type OWNER.

The default key type is OWNER.

Note: Because you must identify the owning workstation to retrieve open interval information, you must specify selection fields from the CP_WORK_STATION object. No CP_WORK_STATION fields are returned in the receive buffer.

Table 248. CP_OPEN_INTERVAL Object Fields

| Field | Type | Len | Description | APPSEL | APPFLD |
|------------------|------|-----|--|--------|--------|
| START_DATE | DATE | 6 | Start date | N | O |
| START_TIME | TIME | 4 | Start time | N | O |
| END_DATE | DATE | 6 | End date | N | O |
| END_TIME | TIME | 4 | End time | N | O |
| MAX_PAR_SERV | BIN | 2 | Maximum parallel servers | N | O |
| MAX_PAR_SERV_DP | BIN | 2 | Maximum parallel servers set at daily planning | N | O |
| SET_BY_MCP | FLAG | 1 | Interval created by MCP | N | O |
| SET_BY_DP | FLAG | 1 | Interval created by DP | N | O |
| CURR_R1_CAP | BIN | 2 | Current R1 resource capacity | N | O |
| R1_CAP_SET_BY_DP | BIN | 2 | R1 resource capacity set at DP | N | O |
| CURR_R2_CAP | BIN | 2 | Current R2 resource capacity | N | O |
| R2_CAP_SET_BY_DP | BIN | 2 | R2 resource capacity set at DP | N | O |
| ALT_WS_NAME | CHAR | 4 | Alternate workstation name | N | O |
| ALT_WS_NAME_DP | CHAR | 4 | Alternate wsname set by DP | N | O |

Current plan operation event object

This option is valid for the current plan operation event object:

- CREATE request with key type SAME.

The default key type is SAME.

Table 249. CP_OPER_EVENT Object Fields

| Field | Type | Len | Description | APPSEL | APPFLD |
|----------------|------|-----|--|--------|--------|
| SUBSYSTEM_NAME | CHAR | 4 | Subsystem name | O | N |
| WS_NAME | CHAR | 4 | Workstation name | O | N |
| JOBNAME | CHAR | 8 | Job name | O | N |
| APPL_ID | CHAR | 16 | Application ID | O | N |
| OPER_NUM | BIN | 15 | Operation number (decimal 1-99) | O | N |
| APPL_IA_DATE | CHAR | 6 | Input arrival date | O | N |
| APPL_IA_TIME | CHAR | 4 | Input arrival time | O | N |
| FORM_NUMBER | CHAR | 8 | Form number for operations at print workstations | O | N |

Table 249. CP_OPER_EVENT Object Fields (continued)

| Field | Type | Len | Description | APPSEL | APPFLD |
|------------------|------|-----|--|--------|--------|
| CLASS | CHAR | 1 | SYSOUT class for operations at print workstations | O | N |
| OPER_TOKEN | CHAR | 8 | Operation token | O | N |
| STATUS | CHAR | 1 | New status: C Complete E Ended in error I Interrupted Q Extended status of a started operation (S) is Q (queued awaiting execution) S Started T Extended status of a started operation (S) is S (operation is executing) X Reset the current status for this operation | N | O |
| ERROR_CODE | CHAR | 4 | Error code (for new status E) | N | O |
| ACT_DUR | CHAR | 4 | Actual duration HHMM (for new status C or E) | N | O |
| EV_CREATION_DATE | CHAR | 4 | Event creation date. The field format is 00YYDDDF for dates in the 20th century, and 01YYDDDF for dates in the 21st century. Default is current date | N | O |
| EV_CREATION_TIME | BIN | 31 | Event creation time (100 * secs). Default is current time | N | O |
| JOB_NUMBER | CHAR | 5 | Job number | N | O |

Note:

- To select an operation, specify at least OPER_TOKEN, or WS_NAME with either JOBNAME or APPL_ID. The remaining values can be initialized to zeros or blanks.
 OPER_TOKEN is a hexadecimal value that uniquely identifies an operation. If you stored the token set in the OPCTOKEN parameter of the operation-initiation exit (EQQUX009), you can provide this token to identify the operation. OPER_TOKEN is valid only for operations at workstations that have a user-defined destination.
- SUBSYSTEM_NAME is the name of the IBM Workload Scheduler for z/OS subsystem that the event should be reported to. It is used only to select the target for the event and is not stored in the representation of the object.
 If you specify SUBSYSTEM_NAME in APPSEL but do not provide a value in the APPVAL section, or you specify MSTR, the event is broadcast to all IBM Workload Scheduler for z/OS subsystems on the same z/OS image. If you do not specify SUBSYSTEM_NAME, the event is reported to the IBM Workload Scheduler for z/OS subsystem that owns the target LU.
 If your ATP invokes the EQQUSIN subroutine directly, and you do not specify SUBSYSTEM_NAME, the event is broadcast to all IBM Workload Scheduler for z/OS subsystems on the same z/OS image.
- If you do not provide enough information to uniquely identify the operation, and IBM Workload Scheduler for z/OS finds more than one operation that

matches the criteria you specified, IBM Workload Scheduler for z/OS must determine the most applicable operation to update. IBM Workload Scheduler for z/OS selects the operation from operations in status R, A, *, S, I, or E, by investigating these characteristics in the stated order:

- a. The operation has priority 9.
- b. Earliest latest start time.
- c. Priority 8-1.
- d. Input arrival time specified for the operation or the occurrence input arrival if the operation does not have input arrival specifically defined.

So from the operations that match the selection criteria, the operation with priority 9 is updated. If more than one operation has priority 9, the operation with the earliest latest start time is updated. If latest start is equal, the operation with the highest priority is updated. If priority is equal, the operation with the earliest input arrival time is updated. If input arrival is also equal, the update is performed on a first-in first-out basis.

4. In the APPFLD section, you must specify at least STATUS.
5. JOB_NUMBER is a number that you can provide for the job. It is valid only for operations at general automatic workstations and workstations that have a user-defined destination. Do not specify JOB_NUMBER for operations that are submitted through a tracker.

Current plan OPINFO event object

This option is valid for the current plan OPINFO event object:

- CREATE request with key type SAME.

The default key type is SAME.

Table 250. CP_OPINFO_EVENT Object Fields

| Field | Type | Len | Description | APPSEL | APPFLD |
|----------------|------|-----|---|--------|--------|
| SUBSYSTEM_NAME | CHAR | 4 | Subsystem name | O | N |
| WS_NAME | CHAR | 4 | Workstation name | O | N |
| JOBNAME | CHAR | 8 | Job name | O | N |
| APPL_ID | CHAR | 16 | Application ID | O | N |
| OPER_NUM | BIN | 15 | Operation number (decimal 1-99) | O | N |
| APPL_IA_DATE | CHAR | 6 | Input arrival date | O | N |
| APPL_IA_TIME | CHAR | 4 | Input arrival time | O | N |
| FORM_NUMBER | CHAR | 8 | Form number for operations at print workstations | O | N |
| CLASS | CHAR | 1 | SYSOUT class for operations at print workstations | O | N |
| USERDATA | CHAR | 16 | User data (free form text) | N | O |

Note:

1. If the OPINFOSCOPE keyword of the JTOPTS statement is IP, which is the default, you must specify WS_NAME for IBM Workload Scheduler for z/OS to identify the operation. If OPINFOSCOPE keyword is set to ALL, you must specify JOBNAME or APPL_ID. The remaining values can be initialized to zeros or blanks.

2. SUBSYSTEM_NAME. See the explanation of this field 2 on page 355.
3. If you do not provide enough information to uniquely identify the operation, and IBM Workload Scheduler for z/OS finds more than one operation that matches the criteria you specified, IBM Workload Scheduler for z/OS must determine the most applicable operation to update. IBM Workload Scheduler for z/OS considers operations in status R, A, *, S, I, or E when selecting the operation. IBM Workload Scheduler for z/OS selects the operation to update by investigating these characteristics in the stated order:
 - a. The operation has priority 9.
 - b. Earliest latest start time.
 - c. Priority 8-1.
 - d. Input arrival time specified for the operation or the occurrence input arrival if the operation does not have input arrival specifically defined.
 - e. Longest in Ready status.

So from the operations that match the selection criteria, the operation with priority 9 is updated. If more than one operation has priority 9, the operation with the earliest latest start time is updated. If latest start is equal, the operation with the highest priority is updated. If priority is equal, the operation with the earliest input arrival time is updated.

If no match has been found, IBM Workload Scheduler for z/OS uses the value of the OPINFOSCOPE keyword of JTOPTS to determine if operations in status C and W are also considered. OPINFOSCOPE can have the value IP (in progress) or ALL. Operations in status C and W are considered only if the value is ALL. The operation with the earliest latest-start-time is selected.

Current plan special resource event object

This option is valid for the current plan special resource event object:

- CREATE request with key type SAME.

The default key type is SAME.

Table 251. CP_SR_EVENT Object Fields

| Field | Type | Len | Description | APPSEL | APPFLD |
|------------------|------|-----|---------------------------------------|--------|--------|
| SUBSYSTEM_NAME | CHAR | 4 | Subsystem name | O | N |
| SR_NAME | CHAR | 44 | Name of special resource | R | N |
| AVAILABLE | CHAR | 1 | Resource availability (Y N K R) | N | O |
| QUANTITY | BIN | 31 | Number available (1-999999) | N | O |
| QUANTITY_OPTION | CHAR | 8 | Quantity option (KEEP RESET) | N | O |
| DEVIATION | BIN | 31 | Number to deviate (-999999 to 999999) | N | O |
| DEVIATION_OPTION | CHAR | 8 | Deviation option (KEEP RESET) | N | O |
| CREATE | CHAR | 1 | Create resource if undefined (Y N) | N | O |

Note:

1. SUBSYSTEM_NAME. See the explanation of this field2 on page 355.
2. AVAILABLE updates the Available field of the special resource, which overrides interval and default values. Specify Y (YES) to make the resource available or N

(NO) to make it unavailable. Specify R (RESET) to set the availability status to the planned status in the current plan, or K (KEEP) to leave availability unchanged.

3. QUANTITY and QUANTITY_OPTION fields are mutually exclusive. They update the Quantity field in the special resource, which overrides interval and default values. Use QUANTITY to set a numeric value or QUANTITY_OPTION to specify KEEP or RESET. If you specify both fields, message EQQE056W is written to the controller message log and the event is ignored.
4. DEVIATION and DEVIATION_OPTION fields are mutually exclusive. Use DEVIATION to set a numeric value or QUANTITY_OPTION to specify KEEP or RESET. If you specify both fields, message EQQE056W is written to the controller message log and the event is ignored. The deviation field in the special resource can contain a positive or negative number, which varies the total amount of the resource. IBM Workload Scheduler for z/OS determines the total amount by adding together the quantity and the deviation. For example, if you specify -2 and the current quantity is 10, the total amount that operations can allocate reduces to 8.
5. CREATE specifies if IBM Workload Scheduler for z/OS should create a resource in the current plan if the resource does not exist. NO indicates that the resource should not be added to the resource definitions of the receiving IBM Workload Scheduler for z/OS subsystem. If the resource is already defined in the receiving subsystem, NO has no effect. You can specify NO if the resource is being used only as a means to generate an event for ETT: the event is generated even if the resource does not exist.
If YES is specified and the DYNAMICADD keyword of the RESOPTS initialization statement is set to YES or EVENT, a resource definition is created in the receiving IBM Workload Scheduler for z/OS subsystem if the resource is not already defined.
6. When you set the quantity or availability of a resource through the API (or other interfaces such as the SRSTAT TSO command or the MCP dialog), the specified value lasts over interval boundaries, even though the next interval can specify a different value. Specify RESET to restore the planned value.

Current plan backup event object

This option is valid for the current plan backup event object:

- CREATE request with key type SAME.

The default key type is SAME.

Table 252. BACKUP_EVENT Object Fields

| Field | Type | Len | Description | APPSEL | APPFLD |
|----------------|------|-----|-----------------------------|--------|--------|
| SUBSYSTEM_NAME | CHAR | 4 | Subsystem name | O | N |
| FILENAME | CHAR | 2 | Name of data set (CP or JS) | R | N |

Note: SUBSYSTEM_NAME. See the explanation of this field 2 on page 355.

Current plan workstation event object

This option is valid for the current plan workstation event object:

- CREATE request with key type SAME.

The default key type is SAME.

Table 253. CP_WS_EVENT Object Fields

| Field | Type | Len | Description | APPSEL | APPFLD |
|------------------|------|-----|---|--------|--------|
| SUBSYSTEM_NAME | CHAR | 4 | Subsystem name | O | N |
| WS_NAME | CHAR | 4 | Workstation name | R | N |
| WS_STATUS | CHAR | 1 | New workstation status: A Active O Offline F Failed | N | R |
| STARTED_FAIL_OPT | CHAR | 1 | For new status O or F: R Restart operations automatically on the alternate workstation L Leave the operations in started status E Set all started operations to ended in error | N | O |
| REROUTE_OPT | CHAR | 1 | For new status O or F: Y Reroute operations to alternate workstation N Leave operations at the inactive workstation | N | O |
| ALT_WS | CHAR | 4 | For new status O or F, workstation for rerouted operations | N | O |

Note:

1. SUBSYSTEM_NAME. See the explanation of this field 2 on page 355.
2. If the value provided for WS_STATUS is equal to the current status, the event is ignored.

Appendix C. Sample library (SEQQSAMP)

The SEQQSAMP library contains samples to help you use IBM Workload Scheduler for z/OS programming interfaces. In most cases, you need only add installation-specific JCL to adapt a member in SEQQSAMP to your requirements. Table 254 lists the members in the SEQQSAMP library that apply to programming interfaces, and provides a brief description of each member. The pages that follow describe the members in more detail. A list of all samples provided with IBM Workload Scheduler for z/OS is found in *IBM Workload Scheduler for z/OS: Planning and Installation*.

If you need to change a sample member, copy the source to a separate library; the original sample member is then available for reference. Also, create an SMP/E usermod for each sample member you execute in the production environment. Changes to the sample source code are then flagged for your attention, and subsequent updates can be reflected in the production code as soon as possible.

Table 254. SEQQSAMP Library Members for Programming Interfaces and the API

| Member | Brief description |
|----------|--|
| EQQAPISM | ASCII file containing a sample API application |
| EQQOCWTO | Assembler routine for programmers to communicate with operators |
| EQQPIFAD | Program-interface PL/I sample that creates a two-operation application in the AD database |
| EQQPIFAP | Program-interface PL/I sample that resolves nonpromptable JCL variables. |
| EQQPIFCB | Program-interface assembler samples for various current plan or LTP actions |
| EQQPIFCL | Program-interface assembler sample that uses the DAYSTAT command to return work or free status for a particular date |
| EQQPIFDJ | Program-interface assembler sample that deletes JCL for completed occurrences from the JCL repository (JS) data set |
| EQQPIFJC | Program-interface COBOL sample to manipulate JCL variable tables |
| EQQPIFJD | Program-interface PL/I sample that can either list or delete records in the JCL repository data set (JS) |
| EQQPIFJV | Program-interface PL/I sample to manipulate JCL variable tables |
| EQQPIFOP | Program-interface REXX sample to modify an operation in the current plan |
| EQQPIFPR | Program-interface REXX sample to list all cyclic periods |
| EQQPIFWI | Program-interface PL/I sample to modify capacity values in an open interval of a current plan workstation |
| EQQRXSTG | An assembler routine to get and free storage for the REXX PIF samples |

IBM Workload Scheduler for z/OS Application Programming Interface

This section provides details of SEQQSAMP members that can help you use the application programming interface (API).

API buffer examples

SEQQSAMP contains samples that show you how to use the EQQUSIN subroutine. Because the format of the buffers used by EQQUSIN is the same for requests made through the API, you can use these samples to develop API applications. But these

samples do not show you how to invoke APPC services; you must develop your own transaction programs (TPs) that initialize and allocate a conversation with IBM Workload Scheduler for z/OS. Use the EQQUSIN samples to create buffers that your TPs can pass to IBM Workload Scheduler for z/OS.

The IBM Workload Scheduler for z/OS sample library SYS1.SAMPLIB contains many APPC samples in a variety of languages. All APPC samples have member names starting with ATB.

IBM Workload Scheduler for z/OS program interface

This section provides details of the SEQQSAMP members, which are samples that use the IBM Workload Scheduler for z/OS program interface (PIF).

The IBM Workload Scheduler for z/OS program interface lets you automate and integrate tasks that must otherwise be performed manually by operators or schedulers.

Install all PIF programs that you use in the production environment as SMP/E usermods to ensure that they are correctly relinked if PIF maintenance is received.

These samples demonstrate practical implementations. Some might fit your requirements exactly.

JS data set maintenance

The sample library contains PIF programs to perform maintenance on the JCL repository (JS) data set. EQQPFDJ deletes the JCL for an operation from the JS file if the entire occurrence of the application is completed. JCL is deleted from the JS file if the JCL can be located and the input arrival time of the application is earlier than current-plan end.

EQQPFDJ can either list or delete records in the JS file for the given SYSIN criteria. This program deletes the JCL for occurrences from the JS file if the entire application status is complete. JCL is deleted from the JS file if the JCL can be located and the input arrival time of the occurrence is earlier than the current-plan end and the input arrival time specified for the input parameter.

The application name can be specified generically. You can use this program to delete all JCL from the JS file with input arrival equal to or earlier than a specific date. Consider scheduling this program regularly.

JCL variable substitution

SEQQSAMP contains PIF samples for JCL variable substitution actions. EQQPFAF provides a PL/1 program to retrieve JCL and resolve all nonpromptable setup variables. The program can be called as a CLIST, REXX exec, or ISPF edit macro. Both CLIST and REXX versions are included in the sample.

You might find this program useful in resolving date and day variables that are shared between business systems, particularly in cases where one system is running late.

EQQPFAF is a sample written in PL/I that can perform general maintenance on the JCL variable tables. You can delete, copy, create, and modify JCL variable tables using this program. EQQPFAF is a sample written in COBOL that provides the same functions as EQQPFAF.

Current plan and LTP actions

The majority of the current plan and LTP PIF samples can be found in a single SEQQSAMP member called EQQPFCB. This member contains these assembler language programs:

- EQQADD adds an occurrence to the CP or LTP.
- EQQDEL deletes an occurrence from the CP or LTP.
- EQQARES adds a special resource to an operation in the CP.
- EQQDRES deletes a special resource from an operation in the CP.
- EQQAPRE adds a predecessor dependency to an occurrence in the CP.
- EQQDPRE deletes a predecessor dependency from an occurrence in the CP.

These programs can be of use if your business systems have dynamic job scheduling requirements. You can use them to build large job streams with a minimum of effort.

Member EQQPFIWI contains a PL/I sample to modify the capacity ceilings of parallel servers and workstation fixed resources for a particular open interval of a workstation in the current plan. You could use this to automatically reflect fixed resource availability affected by hardware problems.

Member EQQPFIOP contains a REXX sample to modify an operation in the current plan.

Other PIF samples

EQQPFIAD provides a PIF sample to define an application description. Use the program-interface for AD and OI updates only if you cannot satisfy your requirements using the batch loader. The batch loader, although itself implemented using the program interface, provides a purpose-built interface for batch updates to application descriptions and operator instructions.

The SEQQSAMP member EQQPFICL shows the use of the DAYSTAT CLIST. DAYSTAT retrieves calendar information from IBM Workload Scheduler for z/OS and determines if an input date is a work day or a free day. The result is returned as an indicator in a TSO CLIST variable.

EQQPFIOPR is a REXX sample that lists all cyclic periods.

EQQRXSTG is an assembler routine that you can use to get and free storage for PIF samples that are written in REXX.

EQQOCWTO is an assembler routine used by the OCL to communicate with operators.

Notices

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

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

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

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

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

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

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

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

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

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

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

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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

COPYRIGHT LICENSE:

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

© (your company name) (year).
Portions of this code are derived from IBM Corp. Sample Programs.
© Copyright IBM Corp. _enter the year or years_.

Trademarks

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

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.



Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

ITIL is a Registered Trade Mark of AXELOS Limited.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Index

A

- access authorization 170
- accessibility xiv
- AD record
 - deleting 17
 - format 272
 - high date considerations 13
 - inserting 30
 - replacing 64
 - selecting 70
- AD segment, replacing 64
- ADAPD segment, record format 272
- ADCIV segment, record format 273
- ADCNC segment
 - record format 276
- ADCNS segment
 - record format 276
- ADCOM segment
 - listing 42
 - record format 274
 - selecting 70
- ADD OCL instruction 175
- ADDCOND OCL instruction 177
- ADDEP segment, record format 275
- ADDPPOP OCL instruction 180
- ADDPRED OCL instruction 184
- ADDRESS OCL instruction 187
- ADDSIMP OCL instruction 189
- ADEXT segment, record format 277
- ADKEY segment
 - listing 42
 - record format 277
- ADOP segment, record format 277
- ADRE segment, record format 279
- ADRROUT 13
- ADRUN segment
 - ADRVALT field 13
 - record format 280
- ADSAI segment
 - record format 282
- ADSR segment, record format 283
- ADUSF segment, record format 284
- advanced program-to-program communication (APPC) 83
- ADVDD segment, record format 284
- ADXIV segment, record format 284
- APARs
 - PI26372 26
 - PI63875 104
 - PI70574 79
 - PI77416 79
 - PI79679 168
 - PK00109 177
 - PK40969 25, 167
 - PK57435 81
 - PK68762 257, 263
 - PK87319 351
 - PK88734 274, 289, 336
 - PK96689 104
 - PM01090 5
 - PM05335 104
- APARs (*continued*)
 - PM07014 206
 - PM10370 26, 167
 - PM23805 59
 - PM60019 121
 - PQ77970 105, 127, 180, 295
 - PQ82419 216
 - PQ87120 308
 - PQ89261 108, 119
 - PQ92888 104
 - PQ95011 105, 106
 - PQ95637 108, 109, 110, 111, 121, 150
 - PQ96176 105, 106, 151
 - PQ97482 31
 - PQ99252 105, 132
 - PQ99633 114
- API (application programming interface)
 - APPC 83
 - application transaction program (ATP) 83
 - buffer layouts 85
 - APP section 86
 - APPDAT section 93
 - APPFLD section 92
 - APPOBJ section 88
 - APPSEL section 90
 - APPVAL section 91
 - communication, establishing 83
 - CPI-C verbs 84
 - CMACCP 84
 - CMALLC 84
 - CMCFMD 84
 - CMINIT 84
 - CMRCV 84
 - CMSSED 84
 - CMSERR 85
 - CMSLD 85
 - CMSTPN 85
 - EBCDIC to ASCII
 - code conversion 84
 - objects 93
 - BACKUP_EVENT fields 358
 - CP_OPEN_INTERVAL fields 354
 - CP_OPER_EVENT fields 354
 - CP_OPERATION fields 346
 - CP_OPINFO_EVENT fields 356
 - CP_RESOURCE fields 351
 - CP_SR_EVENT fields 357
 - CP_STATUS fields 345
 - CP_WORK_STATION fields 352
 - CP_WS_EVENT fields 359
 - selecting instances 94
 - selection field types 95
 - partner transaction program 84
 - reason codes 96
 - return codes 96
 - samples 361
 - security 98
 - selection criteria, specifying 95
 - send/receive buffer formats 85
- API (application programming interface) (*continued*)
 - transaction program (TP) names 83, 85, 98
 - uses of 83
- APP (fixed section)
 - description 86
 - record format 86
- APPC (advanced program-to-program communication) 83
- APPDAT (data section)
 - description 93
 - record format 93
- APPFLD (field section)
 - description 92
 - record format 92
- application programming interface (API)
 - APPC 83
 - application transaction program (ATP) 83
 - buffer layouts 85
 - APP section 86
 - APPDAT section 93
 - APPFLD section 92
 - APPOBJ section 88
 - APPSEL section 90
 - APPVAL section 91
 - communication, establishing 83
 - CPI-C verbs 84
 - CMACCP 84
 - CMALLC 84
 - CMCFMD 84
 - CMINIT 84
 - CMRCV 84
 - CMSSED 84
 - CMSERR 85
 - CMSLD 85
 - CMSTPN 85
 - EBCDIC to ASCII code
 - conversion 84
 - objects 93
 - BACKUP_EVENT fields 358
 - CP_OPEN_INTERVAL fields 354
 - CP_OPER_EVENT fields 354
 - CP_OPERATION fields 346
 - CP_OPINFO_EVENT fields 356
 - CP_RESOURCE fields 351
 - CP_SR_EVENT fields 357
 - CP_STATUS fields 345
 - CP_WORK_STATION fields 352
 - CP_WS_EVENT fields 359
 - selecting instances 94
 - selection field types 95
 - partner transaction program 84
 - reason codes 96
 - return codes 96
 - samples 361
 - security 98
 - selection criteria, specifying 95
 - send/receive buffer formats 85

- application programming interface (API)
 - (*continued*)
 - transaction program (TP) names 83, 85, 98
 - uses of 83
- application transaction program (ATP) 83
- APPOBJ (object section)
 - description 88
 - record format 88
- APPSEL (selection section)
 - description 90
 - record format 90
- APPVAL (selection value section)
 - description 91
 - record format 91
- ASCII 84
- ATP (application transaction program) 83
- AWSCL record, deleting 18
- AWSCL segment
 - listing 42
 - record format 285
 - selecting 70

B

- BACKUP_EVENT object, field names 358
- Batch Command Interface Tool
 - BCIT output 106
 - description 103
 - GROUPDEF support 120
 - instructions 107
 - COPY 108
 - COPY AD 108
 - COPY JCLV 109
 - COPY OI 110
 - COPY RG 110
 - DELETE 110
 - DELETE AD 111
 - DELETE CPCOND 112
 - DELETE CPOC 112
 - DELETE CPOCPRE 112
 - DELETE CPOP 113
 - DELETE CPPRE 114
 - DELETE CPSIMP 115
 - DELETE CPSR 116
 - DELETE JS 117
 - DELETE JSCOM 117
 - DELETE LTCPRE 118
 - DELETE LTOC 118
 - DELETE LTPRE 118
 - DELETE OI 119
 - DELETE RG 119
 - description 107
 - EXPORT 120
 - EXPORT RG 120
 - IMPORT 121
 - IMPORT AD 121
 - IMPORT OI 122
 - IMPORT RG 123
 - INSERT 123
 - INSERT CPCOND 124
 - INSERT CPOC 125
 - INSERT CPOP 127
 - INSERT CPPRE 126, 130

- Batch Command Interface Tool
 - (*continued*)
 - instructions (*continued*)
 - INSERT CPSIMP 131
 - INSERT CPSR 132
 - INSERT LTOC 132
 - INSERT LTPRE 133
 - LIST 133
 - LIST ADCOM 134
 - LIST ADKEY 135
 - LIST CLCOM 135
 - LIST CPCONDCO 135
 - LIST CPOC 136
 - LIST CPOCCOM 136
 - LIST CPOPCOM 136
 - LIST CPWSCOM 138
 - LIST CPWSVCOM 138
 - LIST JCLVCOM 139
 - LIST JSCOM 139
 - LIST LTOCCOM 139
 - LIST OICOM 139
 - LIST PRCOM 140
 - LIST RGCOM 140
 - LIST RGKEY 140
 - LIST WSCOM 141
 - LIST WSVCOM 141
 - LISTSTAT 142
 - LISTSTAT CPOC 142
 - LISTSTAT CPOPCOM 142
 - MODIFY 144
 - MODIFY CPCOND 144
 - MODIFY CPEXT 144
 - MODIFY CPOC 145
 - MODIFY CPOP 146
 - MODIFY CPREND 149
 - MODIFY CPRENZ 149
 - MODIFY CPSAI 149
 - MODIFY LTOC 150
 - REPLACE 151
 - SELECT 151
 - SELECT AD 152
 - SELECT CL 153
 - SELECT CPCOND 153
 - SELECT CPOC 153
 - SELECT CPOP 154
 - SELECT CPST 156
 - SELECT CPWS 156
 - SELECT CPWSV 156
 - SELECT JCLPREP 156
 - SELECT JCLPREPA 157
 - SELECT JCLV 157
 - SELECT JS 157
 - SELECT LTOC 157
 - SELECT OI 158
 - SELECT PR 158
 - SELECT RG/RGCOM 158
 - SELECT WS 159
 - SELECT WSV 159
 - SETSTAT 160
 - SETSTAT CPSIMP 160
 - list 133
 - options 150
 - program example 105
 - return codes 107
 - uses 103
 - branching (OCL GOTO instruction) 224

- branching target (OCL LABEL instruction) 234
- broadcasting events 95, 355
- buffer layouts (API)
 - APP section 86
 - APPDAT section 93
 - APPFLD section 92
 - APPOBJ section 88
 - APPSEL section 90
 - APPVAL section 91
 - description 85
- built-in functions (OCL) 255

C

- CALL OCL instruction 192
- CHGEXTNAME OCL instruction 192
- CHGJOB OCL instruction 194
- CHGOPSAI OCL instruction 195
- CHKAPPL OCL instruction 197
- CHKDATE OCL instruction 201
- CL record
 - deleting 18
 - format 286
 - selecting 70
- CLCOM segment
 - listing 43
 - record format 286
 - selecting 70
- Cloud & Smarter Infrastructure technical training xiv
- CLSD segment
 - record format 287
- CLWD segment
 - record format 287
- Common Programming Interface for Communications (CPI-C)
 - introduction 83
 - verbs 84
- comparison operations (OCL) 227
- COMPL OCL instruction 208
- CP_OPEN_INTERVAL object, field names 354
- CP_OPER_EVENT object, field names 354
- CP_OPERATION object, field names 346
- CP_OPINFO_EVENT object, field names 356
- CP_RESOURCE object, field names 351
- CP_SR_EVENT object, field names 357
- CP_STATUS object, field names 345
- CP_WORK_STATION object, field names 352
- CP_WS_EVENT object, field names 359
- CPCOND record
 - deleting 18
 - inserting 32
 - modifying 51
- CPCOND segment
 - record format 288
- CPCONDCO record
 - format 288
- CPCONDCO segment
 - listing 43
 - selecting 71
- CPCPR segment
 - record format 293

- CPCSU segment
 - record format 294
 - CPEXT record
 - format 294
 - CPEXT segment
 - modifying 51
 - CPI-C (Common Programming Interface for Communications)
 - introduction 83
 - verbs 84
 - CPIVL, record format 309
 - CPOC record
 - deleting 18
 - format 289
 - inserting 31
 - listing 43
 - modifying 51
 - selecting 71
 - CPOCCOM record
 - listing 43
 - selecting 71
 - CPOCCOM segment
 - record format 289
 - CPOCPRE segment
 - deleting 18
 - inserting 31
 - record format 291
 - CPOCSUC segment
 - deleting 18
 - inserting 32
 - record format 292
 - CPOP record
 - deleting 19
 - format 294
 - inserting 32
 - modifying 52
 - selecting 71
 - CPOPCOM segment
 - listing 44
 - record format 293
 - selecting 71
 - CPOPSRU segment
 - listing 45
 - record format 299
 - CPOPT segment
 - record format 310
 - CPPRE segment
 - deleting 19
 - inserting 33
 - record format 300
 - CPREND segment
 - modifying 54
 - record format 301
 - CPRENTZ segment
 - modifying 54
 - record format 302
 - CPSAI segment
 - inserting 34
 - modifying 55
 - record format 303
 - CPSIMP record
 - deleting 19
 - inserting 34
 - CPSIMP segment
 - record format 288
 - CPSR segment
 - deleting 20
 - CPSR segment (*continued*)
 - inserting 35
 - record format 304
 - CPST record
 - format 305
 - selecting 69
 - CPSUC segment
 - deleting 20
 - inserting 35
 - record format 303
 - CPUSRF record
 - format 306
 - inserting 36
 - CPUSRF segment
 - deleting 21
 - modifying 55
 - record format 306
 - selecting 73
 - CPVIVL, record format 313
 - CPWS record
 - format 307
 - modifying 55
 - selecting 73
 - CPWSCOM segment
 - listing 45
 - record format 307
 - selecting 73
 - CPWSV record
 - format 311
 - modifying 56
 - selecting 73
 - CPWSVCOM segment
 - listing 46
 - record format 311
 - selecting 73
 - CSR record
 - modifying 57
 - selecting 74
 - CSR segment
 - selecting 74
 - CSRCOM record
 - selecting 74
 - CSRCOM segment
 - listing 46
 - selecting 74
- D**
- data area (PIF), format of 10
 - default date 12
 - DEL OCL instruction 211
 - DELCOND OCL instruction 213
 - DELETE request (PIF) 16
 - DELPRED OCL instruction 214
 - DELRES OCL instruction 216
 - DELSIMP OCL instruction 218
 - diagnostic data set, EQQDUMP 5
 - Dynamic Workload Console
 - accessibility xiv
- E**
- EBCDIC 84
 - education xiv
 - ELSE OCL clause 226
 - EQQDUMP data set 5
 - EQQMLIB data set 5
 - EQQMLOG (message log data set) 5
 - EQQYCOM subroutine
 - program interface 3
 - data records 271
 - EQQYCOM subroutine (PIF)
 - communicating with 4
 - EQQYPARM 12
 - EQQYPARM (parameter data set) 5
 - EQQYRJCL sample job 265
 - EQQYRPRC
 - sample procedure 266
 - error messages
 - application programming interface 96
 - program interface 6
 - ETT record
 - deleting 21
 - selecting 74
 - ETT segment
 - listing 46
 - selecting 74
 - events, broadcasting 95, 355
 - EXECUTE request (PIF), updating the current plan 10, 24
 - EXIT OCL instruction 221
- F**
- FORCE OCL instruction 221
- G**
- GENDAYS record
 - listing 46
 - GOTO OCL instruction 224
- H**
- high date 12
 - HOLD OCL instruction 224
- I**
- IF-THEN-ELSE OCL instruction 226
 - INIT initialization statement 12
 - INIT OCL instruction 228
 - INIT request (PIF)
 - beginning a communication session 10
 - description 25
 - initialization parameters, specifying 169
 - input arrival date and time, specifying 172
 - INSERT request (PIF)
 - description 28
 - instructions
 - logging 171
 - summary of OCL 164
 - INTFOPTS initialization statement 12
 - IVL record
 - deleting 21
 - inserting 36
 - modifying 58

J

JCL preparation
 performing 80
 simulating 81
 trial 81
 variable substitution 80
JCLPREP record
 inserting 36
 selecting 74
JCLPREPA record
 selecting 74
JCLV record
 deleting 21
 format 319
 inserting 36
 replacing 64
 selecting 75
JCLVC segment
 format 320
JCLVCOM segment
 listing 47
 record format 319
 selecting 75
JCLVD segment, format 321
JCLVV segment
 format 320
JL record, deleting 21
JLCOM segment
 listing 47
 record format 323
 selecting 75
JS record
 deleting 22
 format 321
 inserting 30
 replacing 64
 selecting 75
JSCOM segment
 listing 48
 record format 321
 selecting 75
JSUACT OCL instruction 229
JSVC segment
 record format 319
JSVV segment
 record format 319

K

KILLJOB OCL instruction 230
KILLREC OCL instruction 232

L

LABEL OCL instruction 234
layout of records and segments
 API buffer 86, 93
 APP (fixed section) 86
 APPDAT (data section) 93
 APPFLD (field section) 92
 APPOBJ (object section) 88
 APPSSEL (selection section) 90
 APPVAL (selection value section) 91
 API objects 345, 361
 BACKUP_EVENT 358

layout of records and segments
(continued)

API objects (continued)
 CP_OPEN_INTERVAL 354
 CP_OPER_EVENT 354
 CP_OPERATION 346
 CP_OPINFO_EVENT 356
 CP_RESOURCE 351
 CP_SR_EVENT 357
 CP_STATUS 345
 CP_WORK_STATION 352
 CP_WS_EVENT 359
PIF 272, 330, 340
 AD 272
 ADAPD 272
 ADCIV 273
 ADCNC 276
 ADCNS 276
 ADCOM 274
 ADDEP 275
 ADEXT 277
 ADKEY 277
 ADOP 277
 ADRE 279
 ADRUN 280
 ADSAI 282
 ADSR 283
 ADUSF 284
 ADVDD 284
 ADXIV 284
 AWSCL 285
 CL 286
 CLCOM 286
 CLSD 287
 CLWD 287
 CPCOND 288
 CPCONDSCO 288
 CPCPR 293
 CPCSU 294
 CPEXT 294
 CPIVL 309
 CPOC 289
 CPOCCOM 289
 CPOCPRE 291
 CPOCSUC 292
 CPOP 294
 CPOPCOM 293
 CPOPSRU 299
 CPOPT 310
 CPPRE 300
 CPREND 301
 CPRENZ 302
 CPSAI 303
 CPSIMP 288
 CPSR 304
 CPST 305
 CPSUC 303
 CPUSRF 306
 CPVIVL 313
 CPWS 307
 CPWSCOM 307
 CPWSV 311
 CPWSVCOM 311
 CSRCOM 314
 CSRIVL 316
 ETT 317
 GENDAYS 318

layout of records and segments
(continued)

PIF (continued)
 JCLV 319
 JCLVC 320
 JCLVCOM 319
 JCLVD 321
 JCLVV 320
 JLCOM 323
 JS 321
 JSVC 319
 JSVV 319
 LTCPRE 326
 LTCSUC 326
 LTEXT 327
 LTOC 323, 324
 LTOCCOM 323
 LTOP 325
 LTPRE 326
 LTSUC 327
 OI 328
 OICOM 328
 PR 329
 PRCOM 329
 RG 330
 RGCOM 330
 RGRUN 331
 SR 332
 SRCOM 332
 WS 335
 WSAM 339
 WSCOM 335
 WSEST 337
 WSIVL 338
 WSOPT 340
 WSSD 338
 WSV 340
 WSVCOM 340
 WSVIVL 342
 WSVSD 342
 WSVWD 343
 WSWD 338
 SR 335
 SRIVL
 SRIVL segment 335
 SRIWS
 SRIWS segment 335
LEFT built-in function (OCL) 256
LIST request (PIF) 38
LTCPRE segment
 deleting 22
 record format 326
LTCSUC segment
 record format 326
LTEXT segment
 record format 327
LTOC record
 deleting 22
 format 323
 inserting 37
 modifying 58
 selecting 75
LTOC segment
 record format 324
LTOCCOM segment
 listing 48
 record format 323

LTOCCOM segment (*continued*)
 selecting 75
 LTOP segment
 record format 325
 LTPRE segment
 deleting 22
 inserting 37
 record format 326
 LTSUC segment
 record format 327

M

MATCHTYP record
 list 42
 message library data set (EQQLIB) 5
 message log data set
 EQQMLOG 5
 MLOGDDN 5, 26
 MLOGDDN data set 5, 26
 MODCOND OCL instruction 234
 MODIFY request (PIF) 50
 MODOP OCL instruction 236

N

NOP OCL instruction 241

O

OCL built-in functions 256
 LEFT 256
 RIGHT 256
 SUBSTR 256
 OCL instructions
 comparison operations 227
 description 175
 ELSE clause 226
 specifying 171
 THEN clause 226
 OI record
 deleting 23
 format 328
 inserting 30
 replacing 64
 selecting 76
 temporary, deleting 23
 OICOM segment
 listing 48
 record format 328
 selecting 76
 online tools 103
 OPC Control Language (OCL)
 access authorization 170
 advantages 164
 built-in functions 255
 customizing 167
 description of instructions 175
 executed instruction logging 171
 input arrival date and time 172
 instruction summary 164
 message format 267
 overview 163
 requirements 265
 uses 163
 OPSTAT OCL instruction 242

OPTIONS request (PIF) 59

P

parameter data set
 EQQYPARM 5
 parameter list (PIF)
 action code 7
 argument names 8
 argument values 8
 communication block 9
 data area 8
 overview 6
 resource code 7
 return code 10
 partner transaction program 84
 PIF (program interface)
 communication session
 beginning 10, 25
 ending 10, 79
 data area, format of 10
 data sets
 EQQDUMP 5
 EQQLIB 5
 EQQYPARM 5
 MLOGDDN 26
 required 5
 description 3
 error messages 6
 JCL preparation 80
 simulating 81
 trial 81
 parameter list 6
 action code 7
 argument names 8
 argument values 8
 communication block 9
 data area 8
 resource code 7
 return code 10
 record formats 272, 330, 340
 requests
 DELETE 16
 description 14
 EXECUTE 24
 INIT 25
 INSERT 28
 LIST 38
 MODIFY 50
 OPTIONS 59
 REPLACE 63
 RESET 65
 SELECT 66
 sequence of 10
 SETSTAT 78
 TERM 79
 samples 362
 security 13
 variable substitution 80
 PIF high date 12
 PR record
 deleting 23
 format 329
 selecting 76
 PRCOM segment
 listing 48
 record format 329

PRCOM segment (*continued*)
 selecting 76

program interface (PIF) 3
 communication session
 beginning 10, 25
 ending 10, 79
 data area, format of 10
 data sets
 EQQDUMP 5
 EQQLIB 5
 EQQMLOG 5
 EQQYPARM 5
 MLOGDDN 26
 required 5
 description 3
 error messages 6
 JCL preparation 80
 simulating 81
 trial 81
 parameter list 6
 action code 7
 argument names 8
 argument values 8
 communication block 9
 data area 8
 resource code 7
 return code 10
 record formats 272, 330, 340
 requests
 DELETE 16
 description 14
 EXECUTE 24
 INIT 25
 INSERT 28
 LIST 38
 MODIFY 50
 OPTIONS 59
 REPLACE 63
 RESET 65
 SELECT 66
 sequence of 10
 SETSTAT 78
 TERM 79
 samples 362
 security 13
 variable substitution 80
 PROMPTN OCL instruction 245
 PROMPTY OCL instruction 247

R

RACF
 APPC/MVS 98
 application programming
 interface 98
 program interface 13
 record formats
 API buffer 86, 93
 APP (fixed section) 86
 APPDAT (data section) 93
 APPFLD (field section) 92
 APPOBJ (object section) 88
 APPSEL (selection section) 90
 APPVAL (selection value
 section) 91
 API objects 345, 361
 BACKUP_EVENT 358

record formats (continued)

API objects (continued)

CP_OPEN_INTERVAL 354
 CP_OPER_EVENT 354
 CP_OPERATION 346
 CP_OPINFO_EVENT 356
 CP_RESOURCE 351
 CP_SR_EVENT 357
 CP_STATUS 345
 CP_WORK_STATION 352
 CP_WS_EVENT 359

PIF 272, 314, 316, 317, 330, 340

AD 272
 ADAPD 272
 ADCIV 273
 ADCNC 276
 ADCNS 276
 ADCOM 274
 ADDEP 275
 ADEXT 277
 ADKEY 277
 ADOP 277
 ADRE 279
 ADRUN 280
 ADSAI 282
 ADSR 283
 ADUSF 284
 ADVDD 284
 ADXIV 284
 AWSCL 285
 CL 286
 CLCOM 286
 CLSD 287
 CLWD 287
 CPCOND 288
 CPCONDICO 288
 CPCPR 293
 CPCSU 294
 CPEXT 294
 CPIVL 309
 CPOC 289
 CPOCCOM 289
 CPOCPRE 291
 CPOCSUC 292
 CPOP 294
 CPOPCOM 293
 CPOPSRU 299
 CPOPT 310
 CPPRE 300
 CPREND 301
 CPRENTZ 302
 CPSAI 303
 CPSIMP 288
 CPSR 304
 CPST 305
 CPSUC 303
 CPUSRF 306
 CPVIVL 313
 CPWS 307
 CPWSCOM 307
 CPWSV 311
 CPWSVCOM 311
 GENDAYS 318
 JCLV 319
 JCLVC 320
 JCLVCOM 319
 JCLVD 321

record formats (continued)

PIF (continued)

JCLVV 320
 JLCOM 323
 JS 321
 JSVC 319
 JSVV 319
 LTCPRE 326
 LTCSUC 326
 LTEXT 327
 LTOC 323, 324
 LTOCCOM 323
 LTOP 325
 LTPRE 326
 LTSUC 327
 OI 328
 OICOM 328
 PR 329
 PRCOM 329
 RG 330
 RGCOM 330
 RGRUN 331
 SR 332
 SRCOM 332
 WS 335
 WSAM 339
 WSCOM 335
 WSDEST 337
 WSIVL 338
 WSOPT 340
 WSSD 338
 WSV 340
 WSVCOM 340
 WSVIVL 342
 WSVSD 342
 WSVWD 343
 WSWD 338

RELEASE OCL instruction 249
 RELOP OCL instruction 251
 RELSUCC OCL instruction 253
 REPLACE request (PIF) 63

requests

API

CREATE 85
 DEL 85
 GET 83, 85
 PUT 85

PIF

DELETE 16
 description 14
 EXECUTE 10, 24
 INIT 10, 25
 INSERT 28
 LIST 38
 MODIFY 50
 OPTIONS 59
 REPLACE 63
 RESET 65
 SELECT 66
 sequence of 10
 SETSTAT 78
 TERM 10, 79

requirements, OCL 265

RESET request (PIF) 65

return codes

DELETE 120
 INSERT 133, 161

return codes (continued)

LIST 142
 LISTSTAT 143
 MODIFY 150
 SELECT 159

RG record

deleting 23
 format 330
 selecting 76

RGCOM record

listing 48
 selecting 76

RGCOM segment

record format 330

RGKEY record

listing 48

RGRUN segment

record format 331

RIGHT built-in function (OCL) 256

S

sample job EQQYRJCL 265

sample library (SEQQSAMP)

API samples 361

description 361

PIF samples 362

sample procedure EQQYRPRC 266

security

APPC/MVS 98

application programming

interface 98

program interface 13

SELECT request (PIF) 66

send/receive buffer formats 85

SEQQSAMP (sample library)

API samples 361

description 361

PIF samples 362

SET OCL instruction 255

SETSTAT request (PIF) 78

SETUPD OCL instruction 257

SR record

deleting 23

format 332, 335

selecting 76

SR segment, listing 49

SRCOM record, selecting 76

SRCOM segment

record format 332, 335

selecting 76

SRSTAT OCL instruction 258

substitution of variables 174

SUBSTR built-in function (OCL) 256

syntax diagrams, how to read xv

T

technical training xiv

TERM request (PIF)

description 79

ending a communication session 10, 79

terminate program (OCL EXIT

instruction) 221

THEN OCL clause 226

TOD fields 271
training
 technical xiv
transaction program (TP) names 83, 85,
 98
WSWD segment
 record format 338
WTO OCL instruction 265

U

UNNOP OCL instruction 261
UPD OCL instruction 262

V

valid-to date 12
VALTO argument
 high date considerations 13
variable
 assign a value 255
 change value of 262
 set value of 257
variable substitution
 introduction 174
 program interface 80
 simulating 81
 trial 81
VIVL record
 deleting 23
 inserting 37
 modifying 58

W

WS record
 deleting 24
 format 335
 selecting 77
WSAM segment
 record format 339
WSCOM segment
 listing 49
 record format 335
 selecting 77
WSDEST segment
 record format 337
WSIVL segment
 record format 338
WSOPT segment
 record format 340
WSSD segment
 record format 338
WSSTAT OCL instruction 263
WSV record
 deleting 24
 format 340
 selecting 77
WSVCOM segment
 listing 49
 record format 340
 selecting 77
WSVIVL segment
 record format 342
WSVSD segment
 record format 342
WSVWD segment
 record format 343



Product Number: 5698-WSH, 5698-T08

Printed in USA