IBM

eserver

iSeries

CL 프로그래밍

버전 5

SA30-0246-06

IBM

@server

iSeries

CL 프로그래밍

버전 5

SA30-0246-06

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 523 페이지의 부록 E『주의사항』을 먼저 읽으십시오.

제 7 판(2004년 4월)

| 이 개정판은 새 개정판에 별도로 명시되지 않은 한 IBM Operating System/400(프로그램 5722-SSI)의 버전 5, 릴리스 3, 수정 3 | 및 모든 후속 릴리스와 수정에 적용됩니다. 이 버전은 모든 축약 명령어 세트 컴퓨터(RISC) 모델에서 실행되는 것은 아니며 CISC | 모델에서도 실행되지 않습니다.

본 개정판은 SC41-5721-05를 대체합니다. 이 책의 내용은 RISC 시스템에만 적용됩니다.

목차

그림 vii

CL 프로그래밍(SA30-0246) 머리말 ix
이 책의 사용자 ix
요구사항 및 관련 정보 ix
고객 의견서를 보내는 방법 x

제 1 장 소개 1
제어 언어(CL) 1
프로시듀어 1
모듈 2
프로그램 2
서비스 프로그램 2
명령 구문 3
CL 프로시듀어 3
명령 정의 5
메뉴 5
명령 프롬터 6
오브젝트와 라이브러리 6
오브젝트 6
라이브러리 7
메세지 9
메세지 설명 10
메세지 대기행렬 10
테스트 기능 11

제 2 장 CL 프로그래밍 13
CL 프로그램 작성 14
대화식 입력 15
일괄처리 입력 15
CL 프로시듀어의 각 부분 16
간단한 CL 프로그램의 예 18
CL 프로그램에 사용된 명령 19
RQSDTA와 CMD 매개변수에 입력되는 명령 . . 19
CL 명령 19
CL 프로시듀어 사용 21
변수에 대한 작업 25
변수 선언 27
리스트 또는 규정된 이름을 지정하기 위해 변수 사용 28
변수 안의 소문자 29
예약된 값이나 숫자 매개변수 값을 대체하는 변수 29
변수 값의 변경 30
명령 매개변수상의 후미 공백 32
CL 프로시듀어에 주석 작성 34
CL 프로시듀어 안에서의 처리 제어 35
GOTO 명령 및 레이블 사용 36
IF 명령 사용 37
ELSE 명령 사용 39
내장 IF 명령 사용 41
DO 명령과 DO 그룹 사용 42
DOUNTIL 명령 사용 44
DOWHILE 명령 사용 44
DOFOR 명령 사용 45
ITERATE 명령 사용 46
LEAVE 명령 사용 47
SELECT 명령 및 SELECT 그룹 사용 48
*AND, *OR 및 *NOT 연산자 사용 49
%BINARY 내장 기능 사용 53
%SUBSTRING 내장 기능 사용 55
%SWITCH 내장 기능 사용 58
MONMSG(메세지 모니터) 명령 사용 60
변수로 사용할 수 있는 값 62
시스템 값 검색 62
구성 소스 검색 65
구성 상태 검색 65
네트워크 속성 검색 66
작업 속성 검색 66
오브젝트 설명 검색 68
사용자 프로파일 속성 검색 68
멤버 설명 정보 검색 69
CL 프로시듀어에 대한 작업 70
CL 프로시듀어 명령 기록(logging) 70
CL 모듈 컴파일러 리스팅 71
컴파일 시의 오류 74
프로시듀어 덤프 확보 74
모듈 속성 표시 76
프로그램 속성 표시 76
리턴 코드 요약 76
이전 릴리스의 소스 프로그램 컴파일 77
이전 릴리스(*PRV) 라이브러리 78
이전 릴리스용 CL 컴파일러 지원 설치 79

제 3 장 프로그램과 프로시듀어 간의 흐름 제어 및 통신 81

CALL 명령 81
CALLPRC 명령. 82
RETURN 명령 84
프로그램과 프로시듀어 간의 매개변수 전달 84
CALL 명령 사용 88
프로그램 및 프로시듀어 호출 시 공통 오류. . . 91
프로그램과 프로시듀어 간의 통신을 위한 자료 대기 행렬 사용 96
리모트 자료 대기행렬 98
데이터베이스 파일과 대기행렬의 사용 비교 . . 100
메세지 대기행렬과의 유사성 101
자료 대기행렬 사용 시 요구사항 101
자료 대기행렬이 사용하는 기억장치 관리 . . . 101
자료 대기행렬 할당 102
자료 대기행렬 사용의 예 102
출력 대기행렬과 연관된 자료 대기행렬 작성 . . 107
프로그램과 프로시듀어 간의 통신을 위한 자료 영역 사용 107
로컬 자료 영역. 108
그룹 자료 영역. 109
프로그램 초기화 매개변수(PIP) 자료 영역 . . . 110
리모트 자료 영역 110
자료 영역 작성. 111
자료 영역 잠금 및 할당. 112
자료 영역 표시. 112
자료 영역 변경. 112
자료 영역 검색. 112
자료 영역 검색의 예 112
자료 영역 변경 및 검색의 예 114

제 4 장 오브젝트와 라이브러리 115
오브젝트 유형과 공통 속성. 115
오브젝트에서 수행되는 기능 115
시스템이 자동으로 수행하는 기능. 115
명령을 사용하여 수행할 수 있는 기능 116
라이브러리 117
라이브러리 리스트. 117
라이브러리 리스트 표시 127
총칭 오브젝트명 사용 127
복수 오브젝트 또는 단일 오브젝트의 탐색. . . 128
라이브러리 사용 128
라이브러리 작성 129
라이브러리에 대한 권한 지정 130
오브젝트에 대한 보안 고려사항 131
새로 작성된 오브젝트에 대한 디폴트 공용 권한 132
새로 작성된 오브젝트에 대한 디폴트 감사 속성 134
라이브러리에 오브젝트 놓기 134
라이브러리 삭제 및 지우기. 135
라이브러리명과 내용 표시 136
라이브러리 설명 표시 및 검색. 137
OS/400 국제화. 137
오브젝트 설명 139
오브젝트 설명 표시 139
오브젝트 설명 검색 144
RTVOBJD 예 146
오브젝트 정보 작성 147
시스템에서 사용하지 않는 오브젝트 검출 147
라이브러리 간의 오브젝트 이동 153
오브젝트 사본 작성 156
오브젝트의 이름 변경. 158
오브젝트 압축 또는 압축 해제 160
오브젝트 압축 160
일시적으로 압축 해제된 오브젝트. 161
오브젝트의 자동 압축 해제. 162
오브젝트 삭제 163
자원 할당 164
오브젝트 잠금 상태 표시 167

제 5 장 CL 프로시듀어와 프로그램 안의 오브젝트에 대한 작업 169
CL 프로그램 안의 오브젝트 액세스. 169
예외: 명령 정의, 파일 및 프로시듀어 액세스 170
오브젝트 존재 검사 172
CL 프로시듀어 안의 파일에 대한 작업. 173
CL 프로시듀어에서 파일 참조. 176
CL 프로시듀어에서 파일 열기 및 닫기. . . . 176
파일 선언 177
화면 파일에 대한 자료 송/수신 179
메뉴 제어를 위한 CL 프로그램 작성 181
CL 프로그램에서 화면 파일 대체 182
복수 장치 화면 파일에 대한 작업 184
데이터베이스 파일로부터 자료 수신 187
CL 프로시듀어나 프로그램 안의 데이터 베이스 파일 대체 187
표시 명령으로부터의 출력 파일 참조. 188

제 6 장 확장 프로그래밍 191
QCAPCMD 프로그램 사용 191
QCMDEXC 프로그램 사용 191
DBCS 자료가 있는 QCMDEXC 프로그램 사용 194
QCMDCHK 프로그램 사용 195
CL 프로그램이나 프로시듀어 안의 메세지 서브파일 사용 197
수행 시 CL 명령의 사용자 변경 허용 197

CL 프로시듀어 또는 프로그램 내에서 OS/400 프롬터 사용 198
CL 명령의 선택 프롬팅 199
CL 프로시듀어나 프로그램에서 프롬팅되는 QCMDEXC 203
프로그래머 메뉴 사용 203
STRPGMMNU(프로그래머 메뉴 시작) 명령의 사용 203
명령 분석기 나감점 205
DBCS 자료를 위한 어플리케이션 프로그래밍. . . 205
DBCS 어플리케이션 프로그램 설계 205
DBCS 자료를 처리하기 위한 영숫자 프로그램의 변환 206
CL 프로그램에서의 DBCS 자료 사용 206
CL 프로그램 샘플. 207
설정용 초기 프로그램(프로그래머) 207
테스트 라이브러리에서 제품 라이브러리로 오브젝트 이동(프로그래머) 208
어플리케이션에서 특정 오브젝트 저장(시스템 오퍼레이터). 208
비정상 종료의 회복(시스템 오퍼레이터) 209
작업 제출(시스템 오퍼레이터) 209
장치 화면으로부터의 입력 대기 중 시간종료 . . 209
날짜 산술 수행. 211
프로그램 속성 검색 212
테이프나 광 매체로부터 어플리케이션 로드 및 수행 212
어플리케이션 출력기의 역할 212

제 7 장 메세지 정의 215
메세지 파일 작성 217
독립 ASP의 메세지 파일 218
메세지 파일의 크기 결정 218
파일에 메세지 추가 219
메세지 ID의 할당 220
메세지 및 메세지 도움말 정의. 221
심각도 코드 할당 222
대체 변수 정의. 223
응답에 대한 유효성 검사 지정. 225
즉시 메세지 송신과 응답 처리. 226
응답의 디폴트 값 정의 228
이탈 메세지에 대한 디폴트 메세지 처리 지정 228
메세지 설명의 예 230
2바이트 메세지 정의 231
시스템 메세지 파일 탐색 231
메세지 파일에 대한 탐색 231
메세지 파일 대체 232
메세지 대기행렬의 유형 236
메세지 대기행렬의 작성 또는 변경 238
작업 메세지 대기행렬. 242

제 8 장 메세지에 대한 작업 247
시스템 사용자에게 메세지 송신 247
CL 프로그램으로부터 메세지 송신 248
메세지. 250
메세지 송신의 예 252
SNDPGMMSG상의 호출 스택 항목 식별. . . 255
CL 프로시듀어나 프로그램에서 메세지 수신 . . 271
CL 프로시듀어에서 메세지 검색 277
메세지 대기행렬로부터 메세지 제거 279
CL 프로그램이나 프로시듀어에서의 메세지 모니터 280
디폴트 처리 284
통지 메세지 286
상태 메세지 286
상태 메세지의 표시 방지 287
일시 중단 처리 프로그램 288
QSYSMSG 메세지 대기행렬 290
QSYSMSG 메세지 대기행렬로 송신되는 메세지 291
QSYSMSG로부터 메세지를 수신하는 샘플 프로그램 313
시스템 응답 리스트의 사용. 315
응답 처리 319
메세지 기록 319
작업 기록부 319
QHST 이력 기록부 331
이력 기록부 형식 334
QHST 파일의 처리 336
QHST 작업 시작 및 완료 메세지 336
QHST 파일의 삭제 338

제 9 장 명령 정의 339
명령 정의 방법의 개요 340
단계 설명 340
사용자 정의 명령에 필요한 권한 343
명령 작성의 예. 343
명령 정의 방법. 344
CMD문 사용 345
매개변수 정의 345
자료 유형 및 매개변수에 대한 제한사항 351
매개변수 리스트 정의. 359
단순 리스트 정의 360
혼합 리스트 정의 365
리스트 안의 리스트 정의 368
규정된 이름 정의 372
종속 관계 정의. 375

가능한 선택사항 및 값 376
프롬트 제어 사용 377
조건 프롬팅 378
추가 매개변수 381
키 매개변수 및 프롬트 대체 프로그램 사용 . . . 381
프롬트 대체 프로그램을 사용하기 위한 절차 382
프롬트 대체 프로그램을 사용한 CL의 샘플 . . 386
명령 작성 389
명령 정의 소스 리스트 391
명령 정의문 처리 시 발생하는 오류 393
명령 정의 표시 393
프로시듀어나 프로그램 안의 명령에 대한 명령 정의 변경의 효과 395
명령 디폴트 변경 397
명령 처리 프로그램 또는 프로시듀어 작성 401
CL 또는 HLL 명령 처리 프로그램 작성 . . . 401
REXX 명령 처리 프로시듀어 작성 403
유효성 검사 프로그램 작성 404
명령 정의 및 작성의 예 405
어플리케이션 프로그램 호출 405
디폴트 값 대체 406
출력 대기행렬 표시 407
IBM 명령으로부터의 메세지 재표시 408
약어 명령 작성 409
파일 및 소스 멤버 삭제 409
프로그램 오브젝트 삭제 410

제 10 장 명령 문서화 413
명령 및 명령 도움말 413
명령 도움말 작성 414
명령 도움말의 UIM 소스 생성 414
공통 도움말 공유 416
도움말 모듈에 도움말 텍스트 구성 416
명령 문서의 HTML 소스 생성 417

제 11 장 프로그램 디버깅 419
ILE 프로그램 디버깅 419
ILE 소스 디버거 420
디버그 명령 420
디버그 세션을 위한 프로그램 오브젝트 준비 . . 421
ILE 소스 디버거 시작 423
프로그램 오브젝트를 디버그 세션에 추가 . . . 423
디버그 세션에서 프로그램 오브젝트 제거 . . . 425
프로그램 소스 보기 427
모듈 오브젝트 변경 427
프로그램 오브젝트의 단계화 435
프로그램 오브젝트의 외부 단계화 436
프로그램 오브젝트의 내부 단계화 437
변수 표시 438
변수 값 변경 440
변수 속성 예 442
이름을 변수, 표현식 또는 명령과 같게 함 . . . 442
ILE CL에 대한 소스 디버그 자국어 지원 . . . 443
OPM 프로그램 디버깅 444
디버그 모드 445
호출 스택 447
모니터되지 않은 메세지 처리 448
중단점 450
추적 455
표시 기능 459
변수 값 표시 459
변수 값 변경 461
다른 작업의 디버그를 위해 작업 사용 462
기계 인터페이스 레벨에서의 디버그 465
보안 고려사항 465

부록 A. TFRCTL 명령 467
TFRCTL 명령 사용 467
매개변수 전달 468

부록 B. 작업 기록부 출력 파일 471
작업 기록부 지시 471
1차 작업 기록부 모델 471

부록 C. 사용권 프로그램(LP) 안의 IBM 제공 라이브러리 483
OS/400 사용권 프로그램용 IBM 제공 라이브러리 483
다른 iSeries 사용권 프로그램용 IBM 제공 라이브러리 485

부록 D. CL 명령어 및 키워드의 약어 489
CL 명령어 동사 약어 489
CL 명령어 약어 491
CL 명령어 키워드 및 약어 503

부록 E. 주의사항 523
프로그래밍 인터페이스 정보 525
상표 525

참고 문헌 527

색인 529

그림

1. 리모트 자료 대기행렬 액세스 예 100
2. 호출 프로그램의 예 193
3. LODRUN 명령을 사용한 어플리케이션의 예 213
4. 수행 시 호출 스택의 예 258
5. TOPGMQ(*PRV *)의 예 259
6. 단순명 사용의 예 261
7. 복합명(complex name) 사용의 예 262
8. *PGMBDY 사용 예 1 264
9. *PGMBDY 사용 예 2 265
10. *PGMBDY 사용 예 3 267
11. 수행 시 호출 스택의 예 269
12. *CTLBDY 사용 예 270
13. 단순 리스트의 예 362
14. REXX 단순 리스트의 예 364
15. CL과 HLL에 대한 명령 관계 402
16. REXX에 대한 명령 관계 403
17. ILE 프로그램 오브젝트를 디버그 세션에 추가 424
18. ILE 프로그램 오브젝트를 디버그 세션에 추가 424
19. 디버그 세션에서 ILE 프로그램 오브젝트 제거 426
20. 디버그 세션에서 ILE 프로그램 오브젝트 제거 426
21. 모듈 보기 표시 428
22. 모듈 오브젝트의 보기 변경 429
23. 조건부 중단점 설정 432
24. F11(변수 표시) 키를 사용하여 변수 표시 438

CL 프로그래밍(SA30-0246) 머리말

이 책은 다음을 비롯한 OS/400 프로그래밍과 관련된 내용을 광범위하게 다룹니다.

- 제어 언어(CL) 프로그래밍
- OS/400 프로그래밍 개념
- 오브젝트와 라이브러리
- 메세지 처리
- 사용자 정의 명령
- 사용자 정의 메뉴
- 테스트 기능

이 책의 사용자

이 책은 CL 프로그래머가 아닌 사람을 포함하여 OS/400 프로그래머 또는 어플리케이션 프로그래머 모두를 위한 것입니다. CL 프로그래밍에 대해 자세히 설명하는 이 책의 내용 대부분은 시스템에 전반적으로 적용되며 iSeries 서버에서 지원하는 모든 고급 언어를 활용하는 프로그래머가 사용할 수 있습니다.

요구사항 및 관련 정보

iSeries 기술 정보의 시작점으로 iSeries Information Center를 사용하십시오.

다음과 같이 두 가지 방법으로 Information Center에 액세스할 수 있습니다.

- 다음 웹 사이트에서:

  `http://www.ibm.com/eserver/iseries/infocenter`

- *iSeries Information Center*, SK3T-4091-04 CD-ROM. 이 CD-ROM은 새 iSeries 하드웨어 또는 IBM Operating System/400 소프트웨어 업그레이드 주문 시 함께 제공됩니다. 다음의 IBM Publications Center에서도 CD-ROM을 주문할 수 있습니다.

  `http://www.ibm.com/shop/publications/order`

iSeries Information Center에는 소프트웨어 및 하드웨어 설치, Linux, WebSphere, Java, 고가용성, 데이터베이스, 논리 파티션, CL 명령 및 시스템 어플리케이션 프로그래밍 인터페이스(API)와 같은 새로운 iSeries 정보와 업데이트된 정보가 들어 있습니다. 또한 iSeries 하드웨어 및 소프트웨어의 계획, 문제 해결 및 구성을 도와주는 어드바이저와 파인더도 제공합니다.

신규 하드웨어를 주문할 때마다 *iSeries 설정 및 조작 CD-ROM*, SK3T-4098-02. 이 CD-ROM에 IBM @server Windows용 IBM e(logo)server iSeries Access 및 EZ-Setup 마법사가 있습니다. iSeries Access 제품군는 iSeries 서버에 PC를 연결하기 위한 강력한 클라이언트 및 서버 기능 세트를 제공합니다. EZ-Setup 마법사는 여러 iSeries 설정 타스크를 자동화합니다.

관련 정보에 대해서는 527 페이지의 『참고 문헌』을 참조하십시오.

고객 의견서를 보내는 방법

고객 여러분의 의견은 정확하고 우수한 품질의 정보를 제공하는 데 있어서 매우 중요합니다. 이 책이나 기타 iSeries 책에 관해 의견이 있으시면 전자우편 또는 아래의 전화 번호로 보내주십시오.

- 전자우편: ibmkspoe@kr.ibm.com
- 한국 아이.비.엠 고객만족센터: 02-3781-7114

의견을 보내실 때에는 반드시 다음 항목을 기록해 주십시오.

- 이 책의 이름 또는 iSeries Information Center 주제명
- 책의 주문 번호
- 의견에 해당되는 책의 페이지 번호 또는 주제

제 1 장 소개

여기에서는 Operating System/400®(OS/400)에 대한 중요한 몇 가지 개념을 설명합니다. 또한 각 장을 통해 이 개념들에 관한 자세한 설명을 제공합니다.

시스템 조작은 다음에 의해 제어됩니다.

- CL 명령. CL 명령은 일괄처리 및 대화식 작업('명령 입력' 화면에서와 같이)에서 그리고 CL 프로그램 및 프로시듀어에서 사용됩니다.
- 메뉴 옵션. 시스템 조작은 메뉴 옵션을 선택하여 제어될 수 있습니다. 대화식 사용자는 iSeries 서버 메뉴를 사용하여 여러 시스템 타스크를 수행할 수 있습니다.
- 시스템 메세지. 시스템 메세지는 프로그램 및 프로시듀어 간의 통신, 그리고 프로그램, 프로시듀어 및 사용자 간의 통신에 사용됩니다. 메세지는 상태 정보와 오류 조건을 둘 다 알려줍니다.

제어 언어(CL)

제어 언어(CL)는 오퍼레이팅 시스템에 대한 1차 인터페이스로서, 서로 다른 워크스테이션에 있는 사용자들이 이 인터페이스를 동시에 사용할 수 있습니다. 하나의 제어 언어 명령문을 **명령**이라고 합니다. 다음과 같은 방법으로 명령을 입력할 수 있습니다.

- 워크스테이션에서 개별적으로
- 일괄처리 작업의 한 부분으로
- CL 프로그램 또는 프로시듀어를 작성하기 위한 소스문으로

명령은 명령 행이나 '명령 입력' 화면에서 개별적으로 입력할 수 있습니다.

모든 명령 구문들은 제어 언어를 간단히 사용할 수 있도록 구문 상의 일관성이 있습니다. 또한 오퍼레이팅 시스템은 모든 명령에 대해 프롬트를 제공하고, 대부분의 명령 매개변수에 대해 디폴트 값을 제공하며, 기능 수행 이전에 명령이 올바르게 입력되었는지 확인하는 유효성 검사 기능을 제공합니다. 즉, CL은 서로 다른 시스템 사용자들이 사용할 수 있는 여러 가지 시스템 기능에 대해 융통성 있는 단일 인터페이스를 제공합니다.

특정 명령에 대한 자세한 정보는 iSeries™ Information Center의 CL 주제를 참조하십시오.

프로시듀어

프로시듀어는 하나의 특정 타스크를 수행한 다음 호출자로 리턴하는 하나의 완전한 고급 언어 명령문 세트입니다.

CL에 있어서 프로시듀어는 보통 PGM문으로 시작해서 ENDPGM문으로 끝납니다.

모듈

모듈은 통합 언어 환경®(ILE) 컴파일러를 사용하여 고급 언어 소스 명령문을 컴파일할 때 생기는 오브젝트입니다. CL 모듈은 CRTCLMOD(CL 모듈 작성) 명령을 사용하여 CL 소스를 컴파일하는 방식으로 작성됩니다. 모듈은 실행할 프로그램 안으로 반드시 바인드시켜야 합니다.

CL 모듈은 사용자 작성 프로시듀어와 CL 컴파일러가 생성한 프로그램 입력 프로시듀어의 두 부분으로 구성됩니다. 다른 HLL(예: C)의 경우 한 모듈에 여러 사용자들이 작성한 프로시듀어를 포함시킬 수 있습니다.

프로그램

OS/400®은 두 가지 유형의 프로그램을 지원합니다. **ILE 프로그램**은 통합 언어 환경(ILE)을 준수하는 고급 언어를 사용하여 작성됩니다. ILE 프로그램은 하나 이상의 모듈을 포함하는 OS/400 오브젝트입니다. 모듈을 프로그램으로 바인드시킬 때까지는 모듈을 실행할 수 없습니다. 이러한 프로그램에는 프로그램 입력 프로시듀어가 있어야 합니다. CL 컴파일러가 자신이 작성한 각 모듈에 프로그램 입력 프로시듀어를 생성합니다. 단일 모듈 ILE 프로그램은 CRTBNDCL(바인드된 CL 프로그램 작성) 명령을 사용하여 작성할 수 있습니다. CRTPGM(프로그램 작성) 명령은 ILE CL을 포함하여 서로 다른 ILE 컴파일러에 의해 생성된 모듈 오브젝트가 있는 ILE 프로그램을 작성하는 데 사용할 수 있습니다. ILE 모듈, ILE 프로그램, 서비스 프로그램 및 바인딩 디렉토리에 대한 자세한 정보는 ILE 개념  을 참조하십시오.

OPM CL 프로그램은 기본 프로그램 모델(OPM)을 준수하는 프로그램입니다. OPM CL 프로그램은 CRTCLPGM(CL 프로그램 작성) 명령을 사용하여 소스를 컴파일할 때 생기는 오브젝트입니다.

서비스 프로그램

서비스 프로그램은 하나 이상의 모듈을 포함하는 OS/400 오브젝트입니다. 서비스 프로그램의 프로시듀어가 필요없으면 서비스 프로그램에 바인드시키지 않은 프로그램을 실행할 수 있습니다. 단, 서비스 프로그램을 프로그램에 바인드시키지 않는 한, 서비스 프로그램의 프로시듀어를 실행할 수 없습니다. 서비스 프로그램의 프로시듀어를 호출하기 위해서는 프로시듀어명을 반드시 내보내야 합니다. 서비스 프로그램은 CRTSRVPGM(서비스 프로그램 작성) 명령을 사용하여 작성됩니다.

프로그램이 단 하나의 입력점을 갖는 반면에 서비스 프로그램은 복수 입력점을 가질 수 있습니다. 서비스 프로그램을 직접 호출할 수는 없습니다. 서비스 프로그램 안의 프로시듀어는 프로그램과 서비스 프로그램 안의 다른 프로시듀어로부터 호출될 수 있습니다.

명령 구문

명령어와 매개변수가 각 명령을 구성합니다. 일반적으로 명령어는 조치의 수신자를 나타내는 명사나 문구가 뒤에 오는 동사나 조치로 이루어집니다. 명령어는 보통 1 - 3자의 단축된 단어로 이루어집니다. 이것은 명령을 입력할 때 필요한 글자 수를 줄여줍니다. 한 예로 CL 명령 중 하나인 Send Message가 있습니다. 즉, 사용자로부터 메세지 대기행렬로 메세지를 송신할 때 SNDMSG라는 명령을 사용할 수 있습니다.

CL 명령에 사용되는 매개변수가 키워드 매개변수입니다. 키워드도 명령과 같이 줄여서 사용하며, 그 매개변수의 목적을 나타냅니다. 그러나 명령을 입력할 때 매개변수를 일정한 순서로 지정(위치 지정 스펙)하여 일부 키워드를 생략할 수도 있습니다.

CL 명령 및 키워드에 사용된 약어 리스트는 489 페이지의 부록 D『CL 명령어 및 키워드의 약어』부분을 참조하십시오.

CL 프로시듀어

CL 프로그램과 프로시듀어는 CL 명령으로 구성됩니다. 명령은 프로그램에 바인드할 수 있는 ILE 모듈이나 OPM 프로그램으로 컴파일되는데 이러한 프로그램은 CL이나 다른 언어로 작성된 모듈로 구성됩니다. CL 프로그램의 장점은 다음과 같습니다.

- CL 프로그램과 프로시듀어를 사용하는 것이 CL 명령을 개별적으로 입력하여 실행하는 것보다 더 빠릅니다.
- CL 프로그램과 프로시듀어는 같은 세트의 명령 및 논리에 대해 일관된 처리를 제공합니다.
- 어떤 기능은 CL 명령을 개별적으로 입력하지 못하게 하며, 반드시 CL 프로그램과 프로시듀어의 일부로 입력할 것을 요구합니다.
- 다른 고급 언어(HLL) 프로그램이나 프로시듀어와 마찬가지로 CL 프로그램과 프로시듀어도 테스트하고 디버그할 수 있습니다.
- 매개변수는 CL 프로그램과 프로시듀어로 전달되어 프로그램이나 프로시듀어에 의해 수행되는 조작을 사용할 때 필요한 특정 요구사항에 적용시킬 수 있습니다.
- CL 모듈을 다른 ILE 고급 언어(HLL) 모듈을 사용하여 프로그램에 바인드할 수 있습니다.

CL 프로그램과 프로시듀어는 여러 종류의 어플리케이션에 사용할 수 있습니다. 예를 들어, 다음과 같은 목적에 CL 프로시듀어를 사용할 수 있습니다.

- 프로그램이나 프로시듀어에서 사용되는 명령을 잘 이해하지 못한 채 어플리케이션 기능을 사용하는 대화식 어플리케이션 사용자에게 인터페이스를 제공하기 위해. 따라서 워크스테이션 사용자의 작업이 훨씬 쉬워지고 명령 입력 시 오류 발생률이 줄어듭니다.

- 어플리케이션에서 사용되는 변수(예: 날짜, 시간 및 외부 인디케이터)를 설정하고 어플리케이션에 사용될 라이브러리 리스트를 지정함으로써 어플리케이션의 조작을 제어하기 위해. 이것은 어플리케이션이 수행될 때 언제라도 이 조작이 확실히 실행되도록 보장합니다.
- 시스템 오퍼레이터에게 서브시스템을 시작하거나 파일의 백업 사본을 제공하거나 다른 오퍼레이팅 기능을 수행하는 프로시듀어와 같은 사전정의된 루틴을 제공하기 위해. CL 프로그램과 프로시듀어를 사용하면 오퍼레이터가 정기적으로 사용하는 명령의 수를 줄일 수 있고 시스템 조작을 일관성 있게 수행할 수 있습니다.

시스템이 제공하는 대부분의 CL 명령은 CL 프로그램 및 프로시듀어에서 사용할 수 있습니다. 어떤 명령은 특별히 CL 프로그램과 프로시듀어에서 사용하기 위해 설계된 것이므로 개별적으로 입력해서는 사용이 불가능합니다. 이러한 명령의 예로는 다음과 같은 것이 있습니다.

- 프로그램이나 프로시듀어를 실행할 때 존재하는 조건에 따라, 프로그램이나 프로시듀어가 수행하는 조작을 제어하는 데 사용할 수 있는 논리 제어 명령. 예를 들어, 특정 조건이 있을 *경우*, 특정 처리를 *실행하고*, *그 다음에* 다른 조작을 하는 것입니다. 이와 같은 논리 연산은 CL 프로그램과 프로시듀어 안에 조건 및 무조건 분기를 둘 다 제공합니다.
- 프로그램 또는 프로시듀어가 워크스테이션 사용자와 통신하는 방법을 제공하는 자료 연산. 자료 연산은 프로그램이나 프로시듀어가 워크스테이션으로 형식화된 자료를 송신하거나 워크스테이션에서 그러한 자료를 수신할 수 있도록 하며 데이터베이스에 대한 제한된 액세스를 허용합니다.
- 프로그램이나 프로시듀어가 화면 사용자에게 메세지를 송신할 수 있도록 하는 명령.
- 다른 프로그램과 프로시듀어가 송신한 메세지를 수신하는 명령. 이 메세지는 프로그램 간의 정상적인 통신을 제공하거나 오류나 다른 예외 상태가 발생했음을 표시합니다.
- 프로그램 간 및 프로그램 안에서의 명령 간에 정보를 전달하기 위한 변수와 매개변수의 사용.
- 다른 프로시듀어를 호출하는 명령(명령행이나 일괄처리 작업 스트림에서는 프로시듀어를 호출할 수 없음).

CL 프로그램과 프로시듀어를 사용하면 각 기능에 대한 별도의 프로그램이나 프로시듀어가 있는 어플리케이션을 설계할 수 있으며, 어플리케이션 안에서 어떤 프로그램이나 프로시듀어가 실행되는지를 제어하는 CL 프로그램과 프로시듀어를 가진 어플리케이션을 설계할 수도 있습니다. 어플리케이션은 CL 프로그램과 프로시듀어, 그리고 기타 HLL 프로그램과 프로시듀어로 구성될 수 있습니다. 이런 유형의 어플리케이션에서는 다음과 같은 목적으로 CL 프로그램이나 프로시듀어가 사용됩니다.

- 어플리케이션 안에서 수행될 프로그램 또는 프로시듀어를 판별하기 위해.
- 다른 HLL 언어로는 사용할 수 없는 시스템 기능을 제공하기 위해.

- 어플리케이션 사용자와의 대화 기능을 제공하기 위해.

CL 프로그램과 프로시듀어는 어플리케이션 사용자가 어떤 조작을 수행하고 어떤 필수 프로시듀어를 실행해야 하는지를 융통성 있게 선택할 수 있도록 합니다.

명령 정의

명령 정의를 통해 사용자는 특정 어플리케이션 요구를 만족시키는 추가 명령을 작성할 수 있습니다. 이 명령은 시스템 명령과 유사합니다.

시스템의 각 명령은 명령 정의 오브젝트와 명령 처리 프로그램(CPP)을 갖고 있습니다. 명령 정의 오브젝트는 다음과 같은 항목을 포함하여 명령을 정의합니다.

- 명령어
- CPP
- 명령에 유효한 매개변수와 값
- 명령 입력 시 명령을 검증하기 위하여 시스템이 사용하는 유효성 검사 정보
- 명령에 대한 프롬트가 요구될 때 표시될 프롬트 텍스트
- 온라인 도움말 정보

CPP는 명령이 입력될 때 호출되는 프로그램입니다. 명령이 입력되면 시스템이 유효성 검사를 실행하므로 CPP가 전달되는 매개변수를 항상 검사할 필요는 없습니다.

명령 정의 기능은 다음과 같은 목적으로 사용됩니다.

- CL 명령 사용자에 대한 일관성 있는 인터페이스를 유지시키면서 시스템 사용자가 필요로 하는 고유의 명령을 작성하기 위해.
- 시스템 사용자의 요구사항을 만족시키는 CL 명령들의 대체 버전을 정의하기 위해. 이 기능은 매개변수 값으로 다른 디폴트를 가지고 있거나, 어떤 매개변수는 입력하지 않아도 되도록 명령을 단순화시키고 있습니다. 이와 같은 매개변수에 대해서는 상수 값을 정의할 수 있습니다. IBM 제공 명령은 변경할 수 없습니다.

명령 정의에 대한 자세한 내용은 제 9 장『명령 정의』부분을 참조하십시오. 명령의 온라인 도움말 정보를 작성하는 방법에 대해서는 제 10 장『명령 문서화』부분을 참조하십시오.

메뉴

시스템은 사용자가 메뉴 옵션을 선택하기만 하면 많은 기능을 수행할 수 있는 여러 가지 메뉴를 제공합니다. 메뉴를 사용하여 시스템 타스크를 수행하면 다음과 같은 장점이 있습니다.

- 사용자가 CL 명령과 명령 구문을 이해해야 할 필요가 없습니다.
- 입력해야 할 양과 오류 발생 확률이 많이 줄어듭니다.

시스템 제공 메뉴처럼 사용할 수 있는 메뉴를 작성하는 데 대한 정보는 Application Display Programming  책을 참조하십시오.

명령 프롬터

명령 프롬터를 사용하여 명령 매개변수와 값을 프롬트할 수 있습니다. 프롬터는 직접 호출하거나 어플리케이션에서 호출할 수 있습니다. 프롬터를 사용하면 프롬터가 사용자를 대신하여 매개변수 키워드명과 매개변수 분리문자(예: 어포스트로피 및 괄호)를 삽입하므로 구문상 올바른 CL 명령 스트링을 쉽게 빌드할 수 있습니다. CL 프롬터는 명령, 매개변수, 매개변수 값, 명령 예 및 명령에 의해 발생하는 오류 메세지를 설명하는 데 사용할 수 있는 온라인 명령 도움말에 대한 액세스도 제공합니다.

iSeries Navigator는 클라이언트 PC에 사용되는 그래픽 CL 명령 프롬터를 제공합니다. 웹용 iSeries Access는 웹 브라우저에서 사용되는 HTML 양식 기반의 CL 명령 프롬터를 제공합니다.

OS/400은 F4를 눌러 명령행에서 사용할 수 있는 CL 명령 프롬터를 제공합니다. 또한 명령행 창 표시(QUSCMDLN) API를 사용하여 어플리케이션 내에서 명령행을 표시할 수 있습니다.

오브젝트와 라이브러리

오브젝트는 오브젝트를 설명하거나 때때로 자료를 설명하는 특성 세트로 구성되는 명명된 기억장치 공간입니다. 오브젝트는 기억장치에서 실제로 공간을 차지하고 있는 것으로, 오브젝트에 대해 조작을 수행할 수 있습니다. 오브젝트 속성에는 오브젝트명, 유형, 크기, 작성 날짜 및 오브젝트를 작성한 사용자가 입력한 설명 등이 포함됩니다. 오브젝트 값은 오브젝트에 저장된 정보의 콜렉션입니다. 예를 들면, 프로그램 값은 그 프로그램을 구성하는 코드입니다. 파일값은 그 파일을 구성하는 레코드의 콜렉션입니다. 오브젝트라는 개념은 그 항목이 무엇인지에 관계 없이 시스템 안에 저장될 수 있는 여러 가지 많은 항목들을 언급할 때 사용되는 용어입니다.

오브젝트

대부분의 CL 명령에 의해 실행되는 기능들은 오브젝트에 적용됩니다. 어떤 명령은 모든 유형의 오브젝트에서 사용될 수 있고, 어떤 명령은 특정한 유형의 오브젝트에만 적용됩니다.

시스템은 여러 가지 고유한 유형의 오브젝트를 지원합니다. 다음은 많은 자료 처리 시스템에서 공통되는 오브젝트 유형입니다.

- 파일
- 프로그램

- 명령
- 라이브러리
- 대기행렬
- 모듈
- 서비스 프로그램

그 밖에 다음과 같이 덜 친숙한 오브젝트 유형도 있습니다.

- 사용자 프로파일
- 작업 설명
- 서브시스템 설명
- 장치 설명

서로 다른 오브젝트 유형은 서로 다른 조작상의 특성을 가지고 있습니다. 이러한 차이로 인해 각 오브젝트 유형이 고유한 것입니다. 예를 들면, 파일은 자료가 들어 있는 오브젝트이므로 지시어가 들어 있는 프로그램과는 조작상 특성이 다를 수밖에 없습니다.

각 오브젝트에는 이름이 있습니다. 오브젝트명과 오브젝트 유형은 오브젝트를 식별하는 데 사용됩니다. 오브젝트명은 그 오브젝트를 작성하는 사용자가 할당합니다. 오브젝트 유형은 오브젝트를 작성하는 데 사용된 명령에 의해 결정됩니다. 예를 들어, 프로그램을 작성하여 이름을 OEUPDT(*주문 입력 갱신용*)로 정한 경우 항상 이 이름으로 프로그램을 참조할 수 있습니다. 시스템은 이 오브젝트명(OEUPDT)과 오브젝트 유형(프로그램)을 사용하여 오브젝트를 찾고 그에 관한 조작을 수행합니다. 여러 오브젝트들은 동일한 이름을 가질 수 있는데, 단 오브젝트 유형이 다르거나 서로 다른 라이브러리에 저장되어야만 합니다.

시스템은 오브젝트 유형에 따라 특정 기능이 잘못 사용되는 것을 방지함으로써 무결성을 유지합니다. 예를 들면, CALL 명령은 프로그램 오브젝트를 수행시키는 명령입니다. CALL을 지정하고 파일을 명명했다면, 프로그램이 같은 이름을 사용하지 않는 한 명령이 실패할 것입니다.

라이브러리

라이브러리는 관련 오브젝트들을 그룹으로 묶고 이 오브젝트들을 사용할 때 이름으로 찾기 위해 사용되는 오브젝트입니다. 따라서 라이브러리는 오브젝트 그룹의 디렉토리가 될 수 있습니다. 라이브러리는 오브젝트를 의미있는 콜렉션으로 그룹화하기 위해 사용될 수 있습니다. 예를 들면, 오브젝트를 보안, 백업 또는 처리 요구사항에 따라 그룹화할 수 있습니다. 사용할 수 있는 디스크 기억장치의 용량에 따라 라이브러리가 포함할 수 있는 오브젝트 수와 시스템의 라이브러리 수가 제한됩니다.

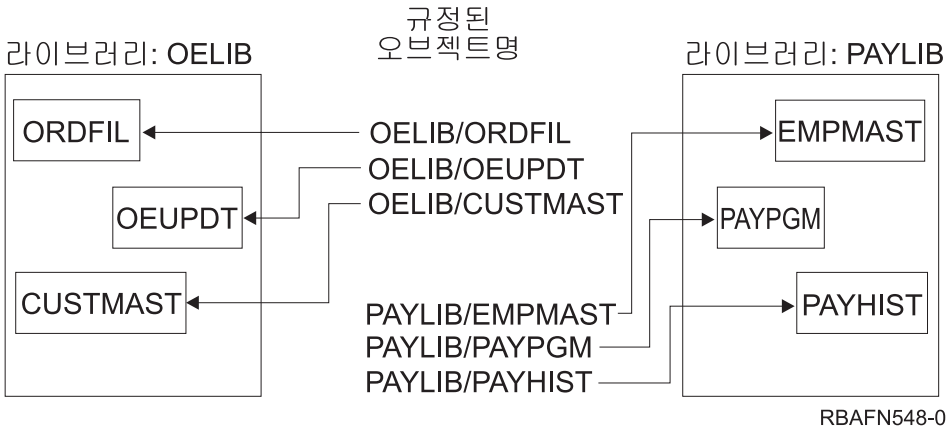
라이브러리에 의해 수행되는 오브젝트 그룹화는 논리적인 그룹화입니다. 라이브러리를 작성할 때 라이브러리를 작성할 사용자 보조 기억장치 풀(ASP)이나 독립 보조 기억장치

풀(독립 디스크 풀) 라이브러리를 지정할 수 있습니다. 라이브러리 내에 작성된 모든 오브젝트는 그 라이브러리와 ASP에 작성됩니다. 라이브러리 안의 오브젝트가 실제로 서로 인접해 있을 필요는 없습니다. 라이브러리나 다른 오브젝트의 *크기*는 기억장치 안에서 사용할 수 있는 인접 공간량에 제약을 받지 않습니다. 시스템은 오브젝트가 시스템 안에 저장되어 있기만 하면 오브젝트에 필요한 기억장치를 찾아냅니다. 독립 ASP에 대한 자세한 정보는 iSeries Information Center의 독립 디스크 풀 주제를 참조하십시오.

거의 모든 유형의 오브젝트가 작성 시 라이브러리에 위치합니다. CRTLIB의 AUT 매개변수가 라이브러리의 공용 권한을 정의합니다. CRTAUT 매개변수는 라이브러리 안으로 작성된 오브젝트의 디폴트 권한을 지정합니다. 오브젝트를 작성하는 명령이 AUT 매개변수에 대해 ***LIBCRTAUT**를 지정하면 오브젝트의 공용 권한은 그 라이브러리에 대해 지정된 작성 권한입니다. 대부분의 오브젝트 유형은 한 라이브러리에서 다른 라이브러리로 이동시킬 수 있으나 한 오브젝트가 동시에 둘 이상의 라이브러리에 들어 있을 수는 없습니다. 오브젝트를 다른 라이브러리로 이동시켜도 오브젝트는 기억장치에서 이동하지 않습니다. 그러나 이제부터는 신규 라이브러리를 통해 오브젝트를 찾아야 합니다. 또한 한 라이브러리에서 다른 라이브러리로 대부분의 오브젝트 유형의 이름을 변경하고 오브젝트 유형을 복사할 수 있습니다.

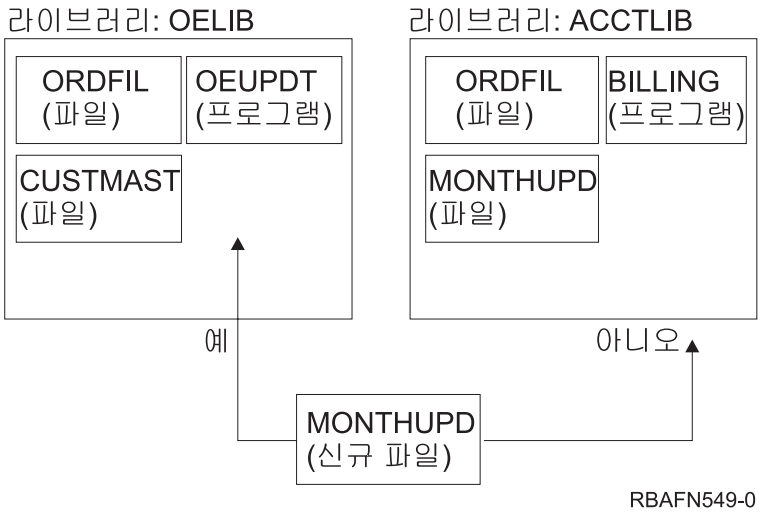
라이브러리명은 오브젝트명에 대한 또 다른 레벨의 식별을 제공하기 위해 사용될 수 있습니다. 앞서 설명한 것과 같이 오브젝트는 이름과 유형으로 식별됩니다. 라이브러리명은 더 나아가 오브젝트명을 규정합니다. 오브젝트명과 라이브러리명의 조합을 오브젝트의 *규정된 이름*이라 합니다. 규정된 이름은 오브젝트명과 오브젝트가 들어 있는 라이브러리명을 시스템에 알려줍니다.

다음의 다이어그램에서는 두 개의 라이브러리와 그 안에 들어 있는 오브젝트의 규정된 이름(qualified name)을 보여줍니다.



같은 이름과 유형을 가진 두 개의 오브젝트가 서로 다른 라이브러리에 존재할 수 있습니다. 같은 이름을 가진 두 개의 서로 다른 오브젝트는 유형이 다른 경우에만 같은 라이브러리에 존재할 수 있습니다. 그러므로 프로그램을 변경하지 않고 프로그램을 연속적으로 수행할 때 이름별로 오브젝트를 참조하는 프로그램이 서로 다른 오브젝트(같은

이름이지만 서로 다른 라이브러리에 저장된 오브젝트)에 대해 작업할 수 있는 것입니다. 또한 새 오브젝트를 작성하는 워크스테이션 사용자는 다른 라이브러리에 있는 오브젝트명을 몰라도 됩니다. 예를 들면, 아래의 다이어그램에서 명명된 MONTHUPD(월별 갱신)인 신규 파일의 경우 라이브러리 OELIB에는 추가될 수 있으나 라이브러리 ACCTLIB에는 추가될 수 없습니다. 이 파일을 ACCTLIB에 작성할 수 없는 이유는 명명된 MONTHUPD이고 유형이 파일인 오브젝트가 이미 라이브러리 ACCTLIB에 존재하기 때문입니다.



오브젝트는 라이브러리 안에서 오브젝트명과 유형으로 식별됩니다. 대부분의 CL 명령은 하나의 오브젝트 유형에만 적용되므로 오브젝트 유형을 명시적으로 식별할 필요는 없습니다. 여러 가지 오브젝트 유형에 적용되는 명령인 경우에는 오브젝트 유형을 명시적으로 식별해야 합니다.

오브젝트를 찾기 위해 라이브러리를 사용하는 자세한 방법에 대해서는 128 페이지의 『라이브러리 사용』 부분을 참조하십시오.

메세지

메세지는 사용자, 프로그램 또는 프로시듀어 간에 이루어지는 통신입니다. 대부분의 자료 처리 시스템은 시스템과 오퍼레이터 간의 통신을 제공하여 처리 과정중에 발생하는 오류나 기타 조건을 처리할 수 있도록 하며, 또한 OS/400은 프로그램과 시스템 사용자 간, 프로그램 간, 프로그램 안의 프로시듀어 간, 그리고 시스템 사용자 간의 양방향 통신을 지원하는 메세지 처리 기능을 제공합니다. 다음과 같은 두 가지 유형의 메세지가 지원됩니다.

- 즉시 메세지. 이 메세지는 메세지 송신시, 프로그램 또는 시스템 사용자가 작성하는 메세지로서 시스템 안에 영구히 저장되지 않는 메세지입니다.
- 사전정의 메세지(predefined message). 이 메세지는 사용에 앞서 미리 작성됩니다. 이 메세지는 작성 시 메세지 파일에 들어가서 사용 시 메세지 파일로부터 검색됩니다.

메세지는 프로그램 간, 프로그램 안의 프로시듀어 간, 프로그램과 사용자 간 통신을 제공하는 데 사용될 수 있으므로 어플리케이션 개발 시 OS/400 메세지 처리 기능을 사용하도록 고려해야 할 것입니다. 메세지 처리에 관한 다음의 개념은 어플리케이션 개발에 있어서 매우 중요합니다.

- 메세지는 메세지 파일에 정의할 수 있으며, 메세지 파일은 메세지를 사용하는 프로그램 외부에 있으며, 변수 정보는 메세지가 송신될 때 메세지 텍스트에서 제공될 수 있습니다. 메세지는 프로그램 외부에서 정의되기 때문에 메세지를 변경할 때 프로그램을 변경할 필요가 없습니다. 이와 같은 방식으로 접근하면 메세지를 다른 언어로 변환하는 것을 포함하여 같은 프로그램을 메세지 파일과 함께 사용할 수도 있습니다.
- 메세지는 시스템에서 독립된 오브젝트인 메세지 대기행렬로 송신되거나 메세지 대기행렬로부터 수신됩니다. 대기행렬로 송신된 메세지는 프로그램 또는 워크스테이션 사용자가 이 메세지를 명확히 수신할 때까지 대기행렬에 남아 있습니다.
- 프로그램은 사용자가 어떤 워크스테이션에 사인 온했는지에 관계 없이 프로그램을 요구한 사용자에게 메세지를 송신할 수 있습니다. 메세지는 특정 장치에 송신할 필요가 없으며 하나의 프로그램을 변경하지 않은 채 서로 다른 워크스테이션에서 사용할 수 있습니다.

메뉴, 메세지 및 메세지 설명에 대한 코드화 문자 세트 ID(CCSID)에 대한 정보는 iSeries Information Center의 **프로그래밍** 범주에 있는 *국제화* 주제를 참조하십시오.

메세지 설명

메세지 설명은 OS/400에 대한 메세지를 정의합니다. 메세지 설명에는 메세지의 텍스트 및 대체 변수에 관한 정보가 들어 있으며, 메세지가 송신될 때 메세지 송신자가 제공하는 변수 자료도 들어 있을 수 있습니다.

메세지 설명은 메세지 파일에 저장됩니다. 각 설명은 파일에 고유한 ID를 하나씩 가지고 있어야 합니다. 메세지가 송신될 때 메세지 파일과 메세지 ID는 시스템에 어떤 메세지 설명이 사용될 것인지를 알려줍니다.

메세지 대기행렬

메세지가 프로시듀어, 프로그램 또는 시스템 사용자로 송신되면 해당 프로시듀어, 프로그램 또는 사용자와 관련되는 **메세지 대기행렬**로 들어갑니다. 프로시듀어나 프로그램, 사용자는 대기행렬에서 메세지를 수신하여 그것을 보게 됩니다.

OS/400은 다음에 대해 메세지 대기행렬을 제공합니다.

- 시스템의 각 워크스테이션
- 시스템에 등록된 각 사용자
- 시스템 오퍼레이터
- 시스템 이력 기록부

특별한 어플리케이션 요구사항을 만족시키기 위하여 메세지 대기행렬이 추가로 작성될 수 있습니다. 메세지 대기행렬로 송신된 메세지는 보관되므로 메세지 리시버는 메세지를 즉시 처리하지 않아도 됩니다.

테스트 기능

시스템에는 프로그램이 실행될 때 수행되고 있는 작업을 프로그래머가 관찰할 수 있는 기능이 들어 있습니다. 이들 기능을 사용하면 의도한 대로 수행되지 않는 조작을 찾을 수 있습니다. 워크스테이션에서 일괄처리 작업이나 대화식 작업 모두에서 테스트 기능을 사용할 수 있습니다. 어느 경우에서나 관찰되고 있는 프로그램은 *디버그 모드*라고 하는 테스트 환경에 있어야 합니다.

테스트 기능은 프로시듀어의 소스문에서 찾기 어려운 오류의 탐색 범위를 축소시켜 줍니다. 실제 출력된 결과가 기대한 것이 아니라는 것만으로도 오류가 발생했음을 분명히 알 수 있습니다. 이러한 오류를 찾으려면 주어진 지점(*중단점*이라고 함)에서 중단한 다음 프로그램의 변수 정보를 검사하여 오류가 없는 지 확인해야 합니다. 프로그램을 계속 실행하기 전에 그러한 변수를 변경할 수도 있습니다.

기계 명령어는 알 필요도 없을 뿐만 아니라, 테스트 기능을 사용하기 위해 프로그램에 특별한 명령어를 포함시킬 필요도 없습니다. OS/400 테스트 기능을 사용하면 다음을 수행할 수 있습니다.

- 프로그램 소스문의 명명된 지점에서 프로그램 수행을 중단시킵니다.
- 프로그램을 중단시킬 수 있는 지점에서 프로시듀어 변수에 관한 정보를 표시합니다. 또한 프로시듀어 처리를 계속하기 전에 변수 정보를 변경할 수 있습니다.

통합 언어 환경(ILE) 프로그램 디버깅에 대한 자세한 정보는 419 페이지의 『ILE 프로그램 디버깅』 부분을 참조하고, OPM 프로그램 디버깅에 대한 자세한 정보는 444 페이지의 『OPM 프로그램 디버깅』 부분을 참조하십시오.

다른 ILE 언어를 사용하여 정보를 디버깅하는 방법에 관해서는 해당 ILE 지침서를 참조하십시오.

제 2 장 CL 프로그래밍

이 장의 중심 내용은 OPM이 아니라 ILE입니다. 이와 같은 이유로 인해, 여기에서는 '프로그램'보다 '프로시듀어'가 사용됩니다. 그러나 일반적으로 CL 명령에 관한 설명에는 '프로그램'이라는 말이 사용됩니다.

CL 프로시듀어는 입력이 오는 곳, 받은 입력을 처리하는 방법, 결과를 넣을 곳을 시스템으로 알려주는 CL 명령의 그룹입니다. 프로시듀어에는 다른 프로시듀어가 호출에 사용할 수 있거나 프로그램에 바인드하여 호출에 사용할 수 있는 이름이 할당됩니다. 다른 프로시듀어에서와 마찬가지로, 프로시듀어를 실행하기 위해서는 먼저 CL 프로시듀어 소스문을 입력한 후 컴파일 및 바인드해야 합니다.

CL 명령을 개별적으로 입력할 경우(예를 들면, '명령 입력' 화면으로부터 또는 입력 스트림 안의 개별 명령으로)에는 명령이 각각 처리됩니다. CL 프로시듀어에 대한 소스문으로 CL 명령을 입력하면 사용자의 선택에 따라 나중에 수정하기 위해 소스로 남겨지며, 명령이 모듈로 컴파일됩니다. 이 모듈은 다른 프로그램으로 바인드되어 실행이 가능한 영구 시스템 오브젝트로 남겨집니다. 즉, CL은 시스템 기능을 위한 실제적인 고급 프로그래밍 언어입니다. CL 프로시듀어는 명령 그룹이 일관성 있게 처리되도록 합니다. 명령을 개별적으로 입력하여 수행할 수 없는 기능은 CL 프로시듀어를 사용하여 수행할 수 있으며, CL 프로그램이나 프로시듀어는 수행 시 별도의 명령을 여러 개 처리하는 것보다 더 좋은 성능을 제공합니다.

CL 프로시듀어는 일괄처리 또는 대화식 처리에 사용될 수 있습니다. 특정 명령이나 기능은 일괄처리 또는 대화식 작업에만 한정됩니다.

CL 소스문은 CL 명령으로 이루어집니다. 모든 CL 명령은 CL 소스문으로 사용할 수 없으며 그 가운데 일부만 CL 프로시듀어나 OPM 프로그램에 사용할 수 있습니다.

CL 소스문은 워크스테이션에서 대화식으로 또는 장치에서 일괄처리 작업 입력 스트림 방식으로 데이터베이스 소스 멤버에 입력할 수 있습니다. CL 소스문을 사용하여 프로그램을 작성하려면 데이터베이스 소스 멤버에 소스문을 입력해야 합니다. 그러면 모듈로 소스 멤버를 컴파일하고, 프로그램 오브젝트로 모듈을 바인드하여 ILE 프로그램을 작성할 수 있습니다.

CL 프로시듀어는 다음을 포함하여 여러 목적을 위해 작성될 수 있습니다.

- 처리 순서를 제어하고 다른 프로그램이나 프로시듀어를 호출하기 위해
- 메뉴를 표시하고 이 메뉴에서 선택한 옵션에 따라 명령을 수행하기 위해. 이렇게 하면 워크스테이션 사용자의 작업이 쉬워지고 오류가 줄어듭니다.
- 데이터베이스 파일을 읽기 위해

- 특정 메세지를 모니터하여 명령, 프로그램 또는 프로시듀어에서 발행된 오류 상태를 처리하기 위해
- 날짜, 시간, 외부 인디케이터 등과 같이 어플리케이션에 사용되는 변수를 설정하여 어플리케이션의 조작을 제어하기 위해
- 시스템 오퍼레이터에게 서브시스템 시작 또는 파일 저장과 같은 사전정의 기능을 제공하기 위해. 그 결과 오퍼레이터가 정기적으로 사용해야 하는 명령의 수가 줄어들고 시스템 조작이 일관성 있게 수행됩니다.

어플리케이션에 대해 CL 프로시듀어를 사용하는 것에는 많은 장점이 있습니다. 예를 들어, 다음과 같은 경우가 있습니다.

- 명령은 프로그램이 작성될 때 처리될 수 있는 형식으로 저장되므로 프로그램을 사용하는 것이 각각의 명령을 입력하거나 실행시키는 것보다 더 빠릅니다.
- CL 프로시듀어에는 융통성이 있습니다. 프로시듀어에 의해 수행되는 조작을 특정 용도의 요구사항에 적용시키기 위하여 매개변수들은 CL 프로시듀어로 전달할 수 있습니다.
- 다른 고급 언어 프로그램과 프로시듀어처럼 CL 프로시듀어도 테스트하고 디버그할 수 있습니다.
- 명령을 개별적으로 입력할 때는 사용이 불가능한 조건 논리와 특수 기능을 CL 프로시듀어와 프로그램에 통합할 수 있습니다.
- CL 프로시듀어를 다른 언어의 프로시듀어와 바인드할 수 있습니다.

다음과 같은 작업에는 CL 프로시듀어를 사용할 수 없습니다.

- 데이터베이스 파일에 레코드를 추가하거나 갱신함
- 프린터나 ICF 파일을 사용함
- 화면 파일 안에서 서브파일을 사용함
- 프로그램 서술 화면 파일을 사용함

CL 프로그램 작성

모든 프로그램은 다음과 같은 단계로 작성됩니다.

1. 소스 작성. CL 프로시듀어는 CL 명령으로 구성됩니다. 대부분의 경우 소스문은 사용자의 어플리케이션 설계에 의해 결정된 논리 순서로 데이터베이스 파일에 입력됩니다.
2. 모듈 작성. CRTCLMOD(제어 언어 모듈 작성) 명령을 사용하여 이 소스는 시스템 오브젝트를 작성합니다. 작성된 CL 모듈은 프로그램으로 바인드될 수 있습니다. 하나의 CL 모듈에는 하나의 CL 프로시듀어가 들어 있습니다. 다른 HLL 언어에는 모듈 각각에 여러 개의 프로시듀어가 들어 있을 수 있습니다.

3. 프로그램 작성. CRTPGM(프로그램 작성) 명령을 사용하는 경우 프로그램을 작성하는 데 이 모듈(다른 모듈 및 서비스 프로그램과 함께)이 사용됩니다.

주: 하나의 CL 모듈로만 구성된 프로그램을 작성하려면 2와 3단계를 결합한 CRTBNDCL(바인드된 CL 프로그램 작성) 명령을 사용하십시오. CL 소스문에서 OPM CL 프로그램을 작성하는 데에는 CRTCLPGM(CL 프로그램 작성) 명령을 사용할 수 있습니다.

대화식 입력

OS/400은 프로그래머 메뉴, 명령 입력 화면, 명령 프롬트 화면 및 프로그래밍 개발 관리자(PDM) 메뉴를 포함하여 프로그래머를 지원하는 다양한 메뉴와 화면을 제공합니다.

보안 - 참조  에 설명된 OS/400 보안 기능을 사용하는 경우 사용자 프로파일에 제공된 권한으로 이러한 표시장치의 사용을 제어합니다. 일반적으로 사용자 프로파일은 시스템 보안 담당자에 의해 작성되고 유지보수됩니다.

가장 자주 사용되는 소스 입력 방식은 WebSphere Development Studio의 일부인 소스 입력 유틸리티(SEU)입니다. EDTF(파일 편집) 명령을 사용하여 데이터베이스 소스 파일에서 CL 명령을 입력하거나 변경할 수도 있습니다. 그러나 EDTF는 SEU에 포함된 통합 CL 명령 프롬트는 지원하지 않습니다.

일괄처리 입력

하나의 일괄처리 입력 스트림으로 CL 소스, CL 모듈 및 프로그램을 작성할 수 있습니다. 다음 예는 입력 스트림의 기본적인 부분을 보여줍니다. 입력은 SBMJOB(작업 제출) 명령을 사용하여 작업 대기행렬에 제출됩니다. 입력 스트림은 다음의 형식을 따라야 합니다.

```
// BCHJOB
CRTBNDCL PGM(QGPL/EDUPGM) SRCFILE(PERLIST)
// DATA FILE(PERLIST) FILETYPE(*SRC)
   .
   .              (CL Procedure Source)
   .
//
/*
// ENDINP
```

이 스트림은 인라인 소스로부터 프로그램을 작성합니다. 소스를 인라인으로 보관하려면 CPYF(파일 복사) 명령을 사용하여 소스를 데이터베이스 파일로 복사하면 됩니다. 그러면, 데이터베이스 파일을 사용하여 프로그램을 작성할 수 있습니다.

IBM 제공 장치 파일을 사용하여 테이프와 같은 외부 매체의 CL 소스에서 CL 모듈을 직접 작성할 수도 있습니다. IBM 제공 테이프 소스 파일은 QTAPSRC입니다. 예를 들어, CL 소스문이 PGMA라는 테이프의 소스 파일에 있는 것으로 가정해 보십시오.

첫 번째 단계는 LABEL 속성 대체 명령과 다음의 대체 명령을 함께 사용하여 테이프에서의 소스 위치를 판별하는 것입니다.

```
OVRTAPF FILE(QTAPSRC) LABEL(PGMA)
```

이제 QTAPSRC 파일을 CRTCLMOD(CL 모듈 작성) 명령에 대한 소스 파일로 간주할 수 있습니다. 다음 명령을 입력하여 테이프 파일의 소스 입력 기반 CL 모듈을 작성하십시오.

```
CRTCLMOD MODULE(QGPL/PGMA) SRCFILE(QTAPSRC)
```

CRTCLMOD 명령은 처리될 때 QTAPSRC 소스 파일을 데이터베이스 소스 파일로 간주합니다. 소스는 대체되어 테이프에 배치됩니다. PGMA가 QGPL에 작성되고 해당 모듈의 소스는 테이프에 그대로 남아 있습니다.

CL 프로시듀어의 각 부분

CL 프로시듀어의 일부로 입력된 각 소스문이 실제로 CL 명령이면, 일반적인 여러 가지 CL 프로시듀어에 사용되는 다음과 같은 기본 부분으로 이 소스를 나눌 수 있습니다.

PGM 명령

`PGM PARM(&A)`

프로시듀어를 시작하고 수신된 매개변수를 식별하는 선택이 가능한 PGM 명령.

선언 명령

`(DCL, DCLF, COPYRIGHT)`

변수 사용 시 프로시듀어 변수의 필수적인 선언(declaration). 선언 명령은 PGM 명령을 제외한 모든 명령 앞에 와야 합니다.

CL 처리 명령

`CHGVAR, SNDPGMMSG, OVRDBF, DLTF, ...`

상수나 변수를 처리하는 데 소스문으로 사용되는 CL 명령(이것은 리스트의 일부임).

논리 제어 명령

`IF, THEN, ELSE, DO, ENDDO, DOWHILE, DOUNTIL, DOFOR, LEAVE, ITERATE, GOTO, SELECT, ENDSELECT, WHEN, OTHERWISE`

CL 프로시듀어 안에서 처리를 제어하는 데 사용되는 명령.

내장 기능

`%SUBSTRING(%SST), %SWITCH, %BINARY(%BIN)`

산술(arithmetic), 관계(relational) 또는 논리식(logical expression)에 사용되는 내장 기능(built-in function)과 연산자(operator)

프로그램 제어 명령

CALL, RETURN

다른 프로그램으로 제어를 전달하는 데 사용되는 CL 명령.

프로시듀어 제어 명령

CALLPRC, RETURN

다른 프로시듀어로 제어를 전달하는 데 사용되는 CL 명령.

ENDPGM 명령

ENDPGM

선택이 가능한 프로그램 종료 명령.

이러한 구성요소의 순서, 조합 및 범위는 사용자 어플리케이션의 논리와 설계에 따라 다릅니다.

CL 프로시듀어는 프로시듀어가 작성될 때 명령이 처리될 때 또는 두 경우에 모두 반드시 존재해야 하는 다른 오브젝트를 참조할 수도 있습니다. 169 페이지의 『CL 프로그램 안의 오브젝트 액세스』와 여러 가지 오브젝트에 대해 설명하는 각 부분에서 이와 같은 차이를 알 수 있습니다. 경우에 따라서는 프로시듀어의 정상적인 수행에 다음 요소가 필요할 수 있습니다.

- 화면 파일. 화면 파일은 화면에서 정보를 구성할 때 사용됩니다. 사용자의 프로시듀어가 화면을 사용하는 경우 모듈을 작성하기 전에 CRTDSPF(화면 파일 작성) 명령을 사용하여 화면 파일과 레코드 형식을 입력 및 작성해야 합니다. 이 파일은 DCLF(파일 선언) 명령을 사용하여 선언 섹션에 프로시듀어로 선언해야 합니다. 자세히 알려면 173 페이지의 『CL 프로시듀어 안의 파일에 대한 작업』을 참조하십시오.
- 데이터베이스 파일. CL 프로시듀어가 데이터베이스 파일의 레코드를 읽을 수도 있습니다. 사용자의 프로시듀어가 데이터베이스 파일을 사용하면 모듈이 작성되기 전에 CRTPF(실제 파일 작성) 명령이나 CRTLF(논리 파일 작성) 명령을 사용하여 파일을 작성해야 합니다. 자료 설명 스펙(DDS), 구조화 조회 언어(SQL) 또는 대화식 자료 정의 유틸리티(IDDU)를 사용하여 파일에 있는 레코드 형식을 정의할 수도 있습니다. 또한 파일은 DCLF(파일 선언) 명령을 사용하여 DCL 섹션에서 프로시듀어로 반드시 선언되어야 합니다. 자세히 알려면 173 페이지의 『CL 프로시듀어 안의 파일에 대한 작업』을 참조하십시오.
- 다른 프로그램. CALL 명령을 사용할 경우 CALL 명령의 실행에 앞서 호출된 프로그램이 반드시 있어야 합니다. 호출 모듈을 컴파일할 때에는 그 프로그램이 반드시 존재하지 않아도 됩니다. 자세한 내용을 알려면 169 페이지의 『CL 프로그램 안의 오브젝트 액세스』와 제 3 장을 참조하십시오.
- 다른 프로시듀어. CALLPRC 명령을 사용할 경우 CRTPGM을 실행할 때 호출된 프로시듀어가 반드시 있어야 합니다. CRTCLMOD를 실행할 때는 없어도 됩니다.

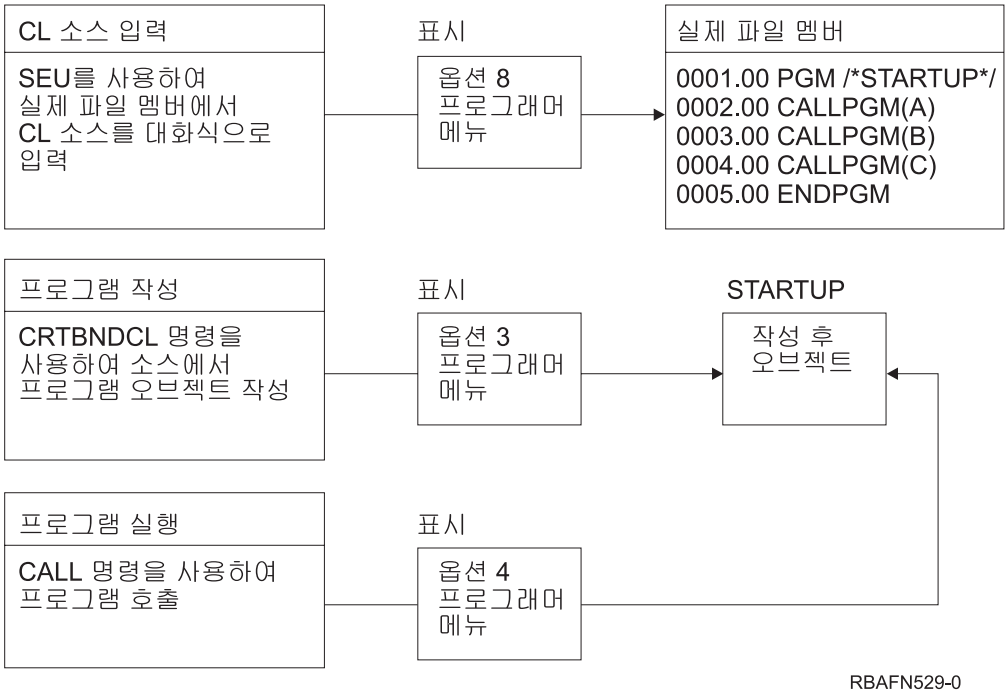
간단한 CL 프로그램의 예

CL 프로그램은 사용자의 필요에 따라 간단하게 또는 복잡하게 작성할 수 있습니다. 하루 일을 시작할 때 통상적으로 시스템 오퍼레이터가 처리하는 몇 가지 활동(예를 들어, 프로그램 A, B, C를 호출하는)을 통합 정리하기 위해 다음의 코드를 사용하여 CL 프로시듀어 STARTUP을 작성할 수 있습니다.

```
PGM  /*  STARTUP  */
CALL PGM(A)
CALL PGM(B)
CALL PGM(C)
    ENDPGM
```

이 예에서는 프로그래머 메뉴를 사용하여 프로그램을 작성했습니다. WebSphere Development Studio의 일부인 프로그래밍 개발 관리자(PDM)를 사용할 수도 있습니다.

이 프로그램을 입력, 작성 및 사용하려면 다음 단계를 수행하십시오.



CL 소스를 입력하려면 다음과 같이 하십시오.

- 프로그래머 메뉴에서 옵션 8(소스 편집)을 선택하고 매개변수(Parm) 필드에 STARTUP을 지정하십시오. (이 옵션은 STARTUP라는 소스 멤버를 작성합니다. 물론 프로그램이 이름을 지정하기도 합니다.)
- 유형 필드에 CLLE를 지정한 후 Enter 키를 누르십시오.
- SEU 화면에서 I(삽입) 행 명령을 사용하여 CL 명령을 입력하십시오. (CALL은 하나의 CL 명령입니다.)

```
 열.............:   1  71            편집                          QGPL/QCLSRC
찾기......: _____________________________________                    STARTUP
FMT A* .....A*. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
      ********************* 자료의 시작 ************************************
.......
.......
.......
.......
.......
.......
```

소스문을 입력했으면, 다음과 같이 하십시오.

- F3 키를 눌러 SEU에서 나가십시오.
- 나감 화면의 디폴트(옵션 2, 멤버 나감 및 갱신)를 적용한 후 Enter 키를 눌러 프로그래머 메뉴로 리턴하십시오.
- 옵션 3(오브젝트 작성)을 선택하여 사용자가 입력했던 소스문으로 프로그램을 작성하십시오. 이때 나오는 화면의 정보는 변경할 필요가 없습니다.

주: 프로그램 STARTUP을 작성할 때 참조 프로그램(A, B, C)은 없어도 됩니다.

프로그램이 작성되면 옵션 4(프로그램 호출)를 선택하고 매개변수(Parm) 필드에 STARTUP을 지정하여 프로그래머 메뉴로부터 이 프로그램을 호출할 수 있습니다. 이 샘플 프로그램을 실행하려면 CALL 명령을 실행할 때 참조 프로그램이 있어야 합니다.

CL 프로그램에 사용된 명령

하나의 CL 프로시듀어에는 하나의 CL 명령만 있을 수 있습니다. 이것은 IBM 제공 명령이나 사용자 정의 명령 모두 가능합니다. 일부 IBM 제공 명령들은 CL 프로시듀어에 사용할 수 없습니다. 개별 명령에 대한 설명과 CL 프로시듀어에서 이러한 명령을 적용할 수 있는지에 대한 자세한 정보는 특정 명령의 온라인 도움말을 참조하거나 **iSeries Information Center**의 *프로그래밍* 범주에 있는 *CL* 섹션을 참조하십시오.

RQSDTA와 CMD 매개변수에 입력되는 명령

작업 이전(TFRJOB) 및 작업 제출(SBMJOB)과 같은 특정 CL 명령은 다른 CL 명령을 매개변수 값으로 사용할 수 있는 RQSDTA 또는 CMD 매개변수를 가지고 있습니다. CL 프로시듀어 안에서만 사용될 수 있는 명령은 RQSDTA 또는 CMD 매개변수에서 값으로 사용될 수 없습니다.

CL 명령

다음은 CL 프로시듀어에 자주 사용되는 명령의 리스트입니다. 이 리스트를 사용하여 원하는 기능에 대한 적절한 명령을 선택할 수 있습니다. 필요한 명령을 결정하는 방법에 대한 정보는 **iSeries Information Center**의 *프로그래밍* 범주에 있는 *CL* 섹션을 참

조하십시오. 이 명령 기능에 친숙해지면 이 장의 뒷부분에 나오는 설명을 이해하는 데 도움이 될 것입니다. 위첨자 1은 CL 프로그램과 프로시듀어에서만 사용할 수 있는 명령을 나타냅니다.

시스템 기능	명령	명령 기능
프로시듀어 제어 변경	CALL(호출)	프로그램 호출
	CALLPRC(프로시듀어 호출) [1]	프로시듀어 호출
	RETURN(리턴)	프로그램이나 프로시듀어를 실행하게 만든 명령 다음의 명령으로 리턴
CL 프로시듀어 한계	PGM(프로그램) [1]	CL 프로시듀어 소스의 시작 표시
	ENDPGM(프로그램 종료) [1]	CL 프로시듀어 소스의 종료 표시
CL 프로시듀어 논리	IF(If) [1]	논리식의 값에 따라 명령 처리
	ELSE(Else) [1]	IF 명령의 else(거짓) 조건에 대한 조치 정의
	DO(Do) [1]	DO 그룹의 시작 표시
	DOWHILE(Do While)[1]	논리 표현식의 값이 참인 상태에서 명령 세트를 처리하는 Do 그룹의 시작을 나타냅니다.
	DOUNTIL(Do Until)[1]	논리 표현식의 값이 참이 될 때까지 명령 세트를 처리하는 Do 그룹의 시작을 나타냅니다.
	DOFOR(Do For)[1]	지정된 값을 기반으로 0회 이상 명령을 처리하는 Do 그룹의 시작을 나타냅니다.
	LEAVE(Leave) [1]	Do While, Do Until 또는 Do For 그룹에서 명령 처리를 종료합니다.
	ITERATE(Iterate) [1]	Do While, Do Until 또는 Do For 그룹에서 명령 처리를 종료하고 그룹 조건을 다시 평가합니다.
	ENDDO(실행 종료) [1]	DO 그룹의 종료 표시
	GOTO(찾아가기) [1]	다른 명령으로 분기
	SELECT(Select)[1]	명령 그룹의 조건부 처리를 허용하는 Select 그룹의 시작을 나타냅니다.
	WHEN(When)[1]	논리 표현식의 값이 참일 때 Select 그룹에서 명령을 처리합니다.
	OTHERWISE(Otherwise) [1]	Select 그룹에서 When 명령에 대해 참인 조건이 없는 경우 처리할 명령을 정의합니다.
	ENDSELECT(선택 종료) [1]	Select 그룹의 종료를 나타냅니다.
CL 프로시듀어 변수	CHGVAR(변수 변경)[1]	CL 변수의 값을 변경
	DCL(선언)[1]	변수 선언
변환	CHGVAR(변수 변경)[1]	CL 변수의 값을 변경
	CVTDAT(날짜 변환) [1]	날짜 형식 변경
자료 영역	CHGDTAARA(자료 영역 변경)	자료 영역 변경
	CRTDTAARA(자료 영역 작성)	자료 영역 작성
	DLTDTAARA(자료 영역 삭제)	자료 영역 삭제
	DSPDTAARA(자료 영역 표시)	자료 영역 표시
	RTVDTAARA(자료 영역 검색)[1]	자료 영역 내용을 CL 변수로 복사
파일	ENDRCV(수신 종료) [1]	RCVF, SNDF 또는 SNDRCVF 명령이 이전에 화면 파일로 발행했던 입력 요구를 취소
	DCLF(파일 선언) [1]	화면 파일이나 데이터베이스 파일을 선언
	RCVF(파일 수신) [1]	화면 파일이나 데이터베이스 파일로부터 레코드 읽기
	RTVMBRD(멤버 설명 검색) [1]	데이터베이스 파일 가운데 특정 멤버의 설명 검색
	SNDF(파일 송신) [1]	화면 파일에 레코드 기록
	SNDRCVF(파일 송신/수신) [1]	화면 파일에 레코드를 기록하고, 사용자가 응답한 후 그 레코드 읽기
	WAIT(대기) [1]	화면 파일에 발행된 SNDF, RCVF 또는 SNDRCVF 명령으로부터 자료가 수신되기를 기다림
Messages	MONMSG(메세지 모니터) [1]	프로그램의 메세지 대기행렬로 송신된 이탈(escape), 상태 및 통지 메세지를 모니터

시스템 기능	명령	명령 기능
	RCVMSG(메세지 수신) [1]	메세지 대기행렬에서 메세지를 CL 프로시듀어의 CL 변수로 복사
	RMVMSG(메세지 제거) [1]	지정 메세지 대기행렬에서 지정 메세지를 제거
	RTVMSG(메세지 검색) [1]	메세지 파일로부터 사전정의된 메세지를 CL 프로시듀어 변수로 복사
	SNDPGMMSG(프로그램 메세지 송신) [1]	프로그램 메세지를 메세지 대기행렬로 송신
	SNDRPY(응답 송신) [1]	응답 메세지를 조회 메세지의 송신자에게 송신
	SNDUSRMSG(사용자 메세지 송신)	정보 또는 조회 메세지를 화면이나 시스템 오퍼레이터에게 송신
기타 명령	CHKOBJ(오브젝트 검사)	오브젝트의 존재 여부 검사 및 오브젝트 사용에 대한 필수 권한 검사(선택사항)
	PRTCMDUSG(명령 사용 인쇄)	지정된 CL 프로시듀어 그룹에서 사용되는 지정 명령 그룹에 대한 상호 참조 리스팅을 작성합니다.
	RTVCFGSRC(구성 소스 검색)	기존 구성 오브젝트를 작성하기 위해 CL 명령 소스를 생성하며 소스 파일 멤버 안에 소스를 배치
	RTVCFGSTS(구성 상태 검색) [1]	어플리케이션이 세 가지의 구성 오브젝트인 행, 제어기 및 장치로부터 구성 상태를 검색할 수 있도록 함
	RTVJOBA(작업 속성 검색) [1]	하나 이상의 작업 속성값을 검색하여 이 값을 CL 변수에 넣기
	RTVSYSVAL(시스템 값 검색) [1]	시스템 값을 검색하여 그것을 CL 변수에 넣기
	RTVUSRPRF(사용자 프로파일 검색) [1]	사용자 프로파일 속성을 검색하여 그것을 CL 변수에 넣기
프로그램 작성 명령	CRTCLMOD(CL 모듈 작성)	CL 모듈 작성
	DLTMOD(모듈 삭제)	모듈 삭제
	DLTPGM(프로그램 삭제)	프로그램 삭제
	CRTBNDCL(바인드 제어 언어 프로그램 작성)	바인드된 CL 프로그램 작성
	CRTCLPGM(CL 프로그램 작성)	OPM CL 프로그램 작성
	CRTPGM(프로그램 작성)	하나 이상의 모듈로부터 하나의 프로그램 작성
	CRTSRVPGM(서비스 프로그램 작성)	하나 이상의 모듈로부터 하나의 서비스 프로그램 작성

CL 프로시듀어 사용

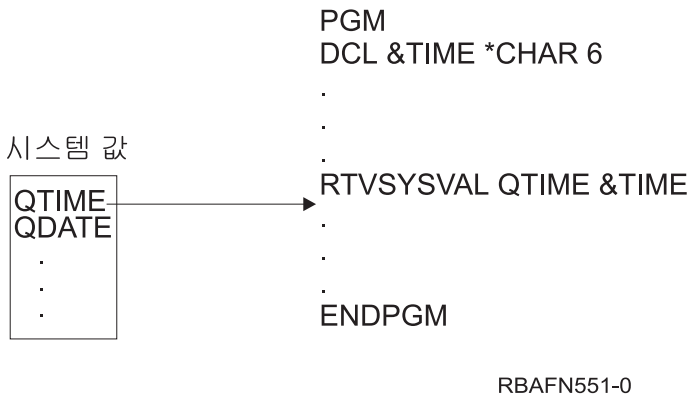
CL 프로그래밍은 다양한 작업을 수행할 수 있도록 해주는 융통성 있는 툴입니다. 아래의 각 사용법은 이 장의 뒤에 나오는 각 섹션에서 더 자세하게 설명됩니다. 일반적으로 사용자는 다음을 수행할 수 있습니다.

- CL 프로시듀어에 있는 변수, 논리 제어 명령, 표현식 및 내장 기능을 사용하여 자료를 조작하고 처리할 수 있습니다.

```
   PGM
DCL &C *LGL
DCL &A *DEC VALUE(22)
DCL &B *CHAR VALUE(ABCDE)
•
•
•
CHGVAR &A (&A + 30)
•
•
•
IF (&A < 50) THEN(CHGVAR &C '1')
•
DSPLIB ('Q' || &B)
•
```

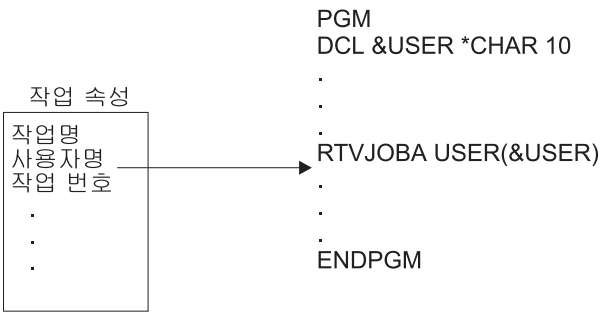
```
IF (%SST(&B 5 1)=E) THEN(CHGVAR &A 12)
•
•
•
ENDPGM
```

- CL 프로시듀어에 있는 변수로 시스템 값을 사용합니다.



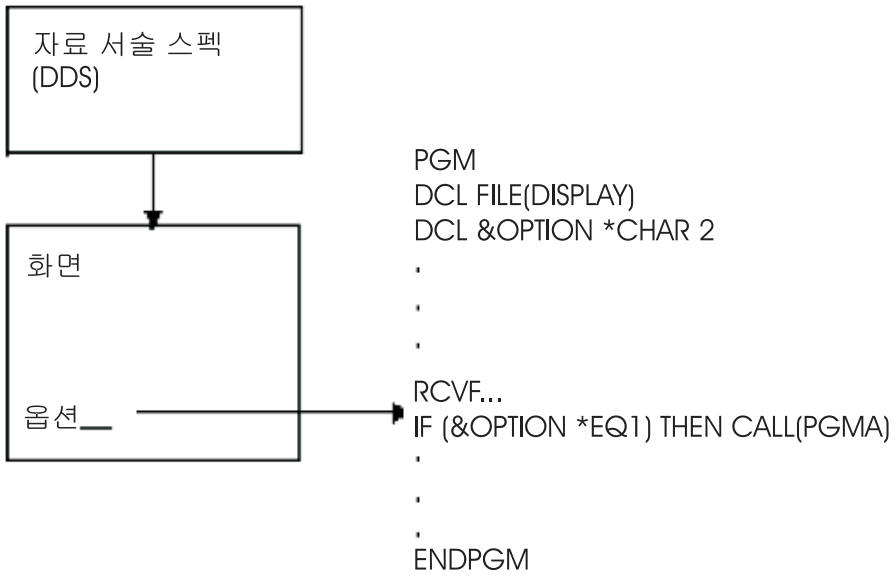
RBAFN551-0

- CL 프로시듀어에 있는 변수로 작업 속성을 사용합니다.



RBAFN552-0

- CL 프로시듀어를 사용하여 화면 파일로 자료를 송신하고 화면 파일로부터 수신합니다.

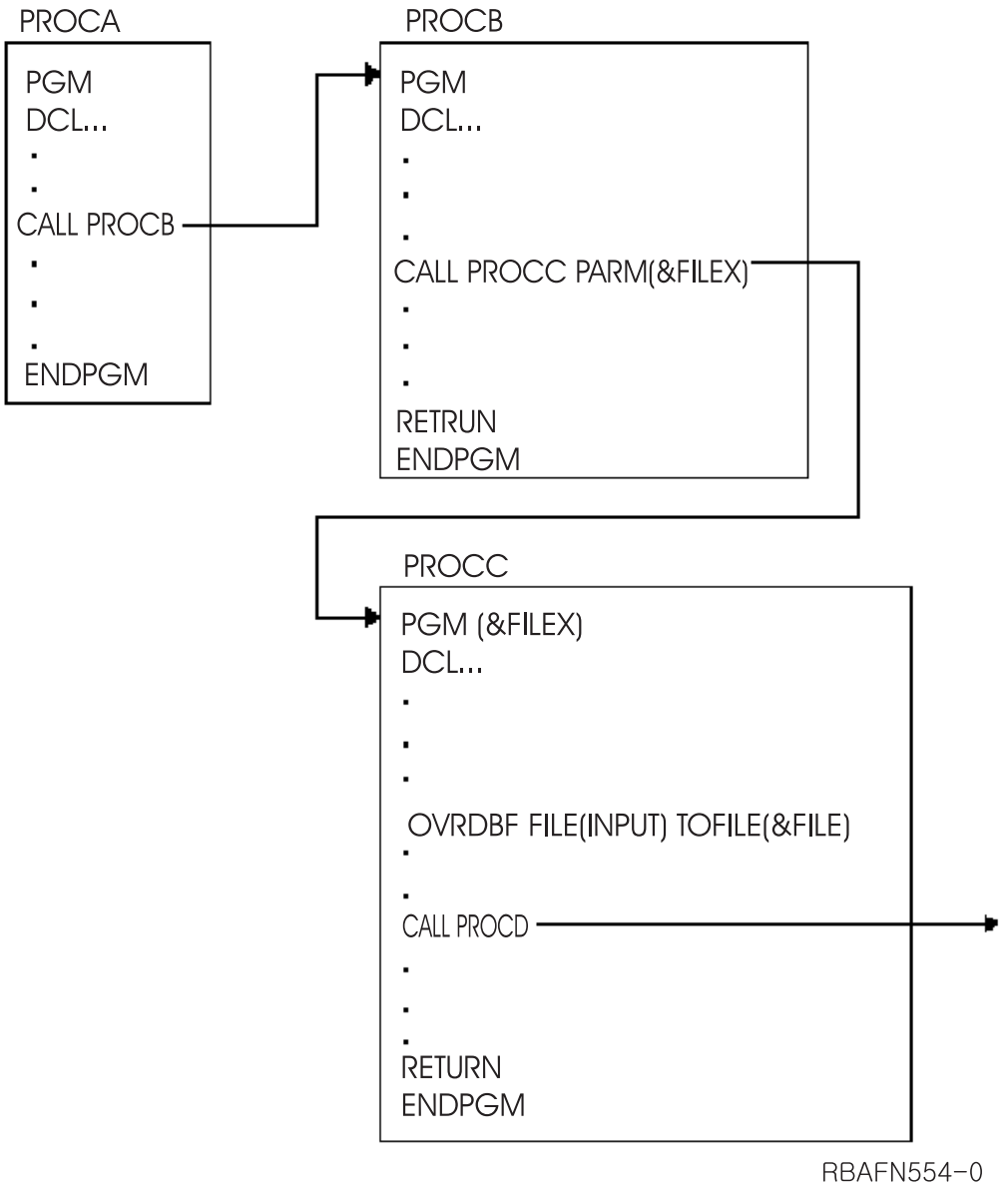


RBAFN553-0

- 작업에 대한 오류 메세지를 모니터하는 CL 프로시듀어를 작성하고, 필요한 경우 정정 조치를 수행할 수 있습니다.

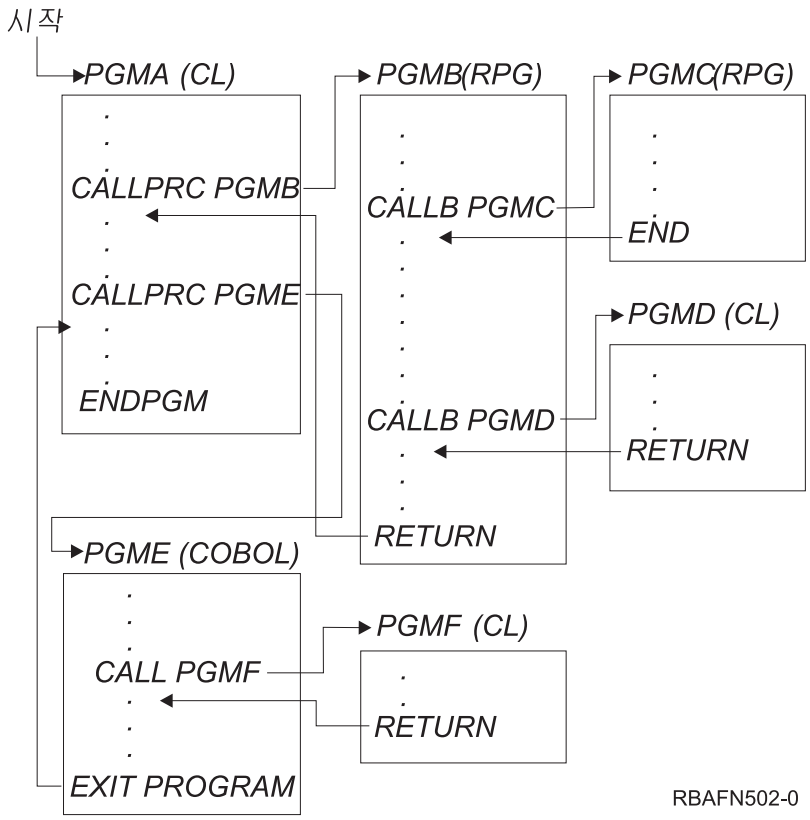
```
   PGM

        MONMSG MSGID(CPF0001) EXEC(GOTO ERROR)
        CALL PROGA
        CALL PROGB
          RETURN
ERROR:  SNDPGMMSG MSG('A CALL command failed') MSGTYPE(*ESCAPE)
ENDPGM
```

- 프로시듀어와 프로그램 간의 처리를 제어하고, CL 프로시듀어의 매개변수를 다른 프로시듀어나 프로그램으로 전달하여 파일을 대체할 수 있습니다.



제어 프로시듀어로 사용되는 CL 프로시듀어는 다른 언어로 작성된 프로시듀어를 호출할 수 있습니다. 앞의 일러스트레이션은 어플리케이션에서 CL 프로시듀어와 RPG IV* 및 ILE COBOL 프로시듀어 간에 제어 권한이 전달되는 방법을 보여줍니다. 어플리케이션을 사용하기 위해 워크스테이션 사용자는 전체 어플리케이션을 제어하는 프로그램 A를 요구합니다. 일러스트레이션은 PGMA와 함께 CALL 명령을 사용하여 호출된 단일 바운드 프로그램(PGMA)을 보여줍니다. PGMA는 다음과 같은 요소로 구성됩니다.

- RPG IV 프로시듀어(PGMB)를 호출하는 CL 프로시듀어(PGMA)
- 다른 RPG IV 프로시듀어(PGMC)를 호출하는 RPG IV 프로시듀어(PGMB)
- CL 프로시듀어(PGMD)를 호출하는 RPG IV 프로시듀어(PGMB)
- ILE COBOL 프로시듀어(PGME)를 호출하는 CL 프로시듀어(PGMA)
- CL 프로시듀어(PGMF)를 호출하는 ILE COBOL 프로시듀어(PGME)



다음 예와 같이 프로시듀어를 작성할 수 있습니다. 별도의 소스 멤버로부터 프로시듀어의 소스를 입력할 수 있습니다.

```
CRTCLMOD PGMA
CRTRPGMOD PGMB
CRTRPGMOD PGMC
CRTCLMOD PGMD
CRTCBLMOD PGME
CRTCLMOD PGMF
CRTPGM PGM(PGMA) +
       MODULE(PGMA PGMB PGMC PGMD PGME PGMF) +
       ENTMOD(*FIRST)
```

변수에 대한 작업

CL 프로시듀어는 CL 명령으로 구성되며, CL 명령 자체는 명령문, 매개변수 및 매개변수 값으로 구성됩니다.

매개변수 값은 변수, 상수 또는 표현식으로 표시할 수 있습니다. **변수**는 그 이름을 참조함으로써 액세스하거나 변경할 수 있는 이름을 가진 값입니다. CL 명령의 거의 모든 매개변수 값에 대해 변수를 하나의 대체값으로 사용할 수 있습니다. CL 변수가 매개변수 값으로 지정되고 그것이 들어 있는 명령이 실행되면 변수의 값이 매개변수 값으로 사용됩니다. 명령이 실행될 때마다 변수에 다른 값을 대체할 수 있습니다. 변수와 표현식은 CL 프로시듀어와 프로그램 안에서만 매개변수 값으로 사용될 수 있습니다.

변수는 라이브러리에 저장되지 않고, 오브젝트가 아니며, 그리고 변수가 들어 있는 프로시듀어가 더 이상 활동하지 않을 경우에는 변수 값이 파기됩니다. 변수를 값으로 사용하면 CL 프로그래밍에 특별한 융통성을 부여하게 되며, 그 이유는 특정 어플리케이션에 의해 그 내용이 변경될 수 있는 높은 수준의 오브젝트 조작이 가능하기 때문입니다. 예를 들어, 제어할 프로그램이나 워크스테이션을 지정하지 않고 다른 프로그램의 처리나 몇 개의 워크스테이션 조작을 지시하는 CL 프로시듀어를 작성할 수도 있습니다. 시스템은 이것을 CL 프로시듀어에서 변수로 식별합니다. CL 프로시듀어를 실행할 때 변수 값을 정의(지정)할 수 있습니다.

모든 변수가 프로시듀어에서 사용하기 위해서는 먼저 CL 프로시듀어로 선언(정의)이 이루어져야 합니다.

- 변수 선언. 변수 정의는 DCL(CL 변수 선언) 명령을 사용하며, 변수의 속성 정의로 이루어집니다. 속성에는 유형, 길이 및 초기값 등이 있습니다.

  ```
  DCL  VAR(&AREA)  TYPE(*CHAR)  LEN(4)  VALUE(BOOK)
  ```

- 파일 선언. 사용자의 CL 프로시듀어가 파일을 사용할 경우 DCLF(파일 선언) 명령의 FILE 매개변수에 파일명을 지정해야 합니다. 파일에는 파일 안의 레코드와 레코드 안의 필드에 관한 설명(형식)이 포함되어 있습니다. 컴파일하는 동안, DCLF 명령이 파일에서 정의된 필드와 인디케이터에 대한 CL 변수를 내재적으로 선언합니다.

  파일의 DDS가 두 개의 필드(F1과 F2)와 함께 한 개의 레코드를 갖고 있으면 두 개의 변수 &F1과 &F2가 자동으로 프로그램에 선언됩니다.

  ```
  DCLF  FILE(MCGANN/GUIDE)
  ```

  파일이 DDS 없이 작성된 실제 파일이면, 하나의 변수가 전체 레코드에 대해 선언됩니다. 변수는 파일명과 동일하며 변수의 길이도 파일의 레코드 길이와 동일합니다.

선언 명령은 프로시듀어에서 다른 모든 명령(PGM 명령을 제외한)보다 앞에 있어야 하지만, 임의의 순서로 혼합이 가능합니다.

여기에서 설명한 것 이외의 다음 경우에서도 변수를 사용할 수 있습니다.

- 프로시듀어와 작업 간의 정보 전달. 81 페이지의 제 3 장 『프로그램과 프로시듀어 간의 흐름 제어 및 통신』 부분을 참조하십시오.
- 프로시듀어와 장치 화면 간의 정보 전달. 184 페이지의 『복수 장치 화면 파일에 대한 작업』 부분을 참조하십시오.
- 조건부 명령 처리. 35 페이지의 『CL 프로시듀어 안에서의 처리 제어』 부분을 참조하십시오.
- 오브젝트 작성. 오브젝트명이나 라이브러리명 또는 두 가지를 모두 대신하여 변수를 사용할 수 있습니다. 다음 예에서 첫 번째 행은 CRTPF(실제 파일 작성) 명령을 사용하여 라이브러리를 지정하였고, 두 번째 행은 라이브러리명을 변수로 대체한 것을 보여줍니다.

```
CRTPF  FILE(DSTPRODLB/&FILE)
CRTPF  FILE(&LIB/&FILE)
```

명령어 또는 키워드를 변경하거나 CALLPRC 명령에 프로시듀어명을 지정하는 작업에는 변수를 사용할 수 없습니다. 그러나 CL 프로시듀어 처리 시 프롬트 기능을 사용하여 명령 매개변수를 변경할 수 있습니다. 자세히 알려면 197 페이지의 『수행 시 CL 명령의 사용자 변경 허용』 부분을 참조하십시오.

또한 명령에 대한 키워드 및 매개변수를 어셈블하거나 QCAPCMD API 또는 QCMDEXC API를 사용하여 이를 처리할 수도 있습니다. 추가 정보에 대해서는 191 페이지의 『QCAPCMD 프로그램 사용』 및 191 페이지의 『QCMDEXC 프로그램 사용』 부분을 참조하십시오.

변수 선언

DCL(CL 변수 선언) 명령에는 가장 간단한 형식으로 다음 매개변수가 있습니다.

DCL 명령을 사용할 때는 반드시 아래의 규칙을 준수해야 합니다.

- CL 변수명은 최대 10자가 뒤에 오는 앰퍼샌드(&)로 시작되어야 합니다. & 뒤에 오는 첫 번째 문자는 영문자, 나머지 문자들은 영숫자여야 합니다. 예를 들면, &PART와 같습니다.
- CL 변수 값은 다음 중 하나여야 합니다.
  - 5000자까지의 문자 스트링
  - 소수 자릿수(decimal position)가 9자리까지 가능하며 총 자릿수가 최대 15자리인 팩 10진값(packed decimal value).
  - '0' 또는 '1'의 논리값. 여기서 '0'은 OFF, 거짓, 아니오를 의미하고, '1'은 ON, 참, 예를 의미합니다. 논리 변수는 '0'이나 '1'이어야 합니다.
  - 2바이트 또는 4바이트의 정수 값. 이 값은 TYPE 매개변수에 *INT이 지정된 경우 음수일 수 있습니다.
- 초기값(initial value)을 지정하지 않으면 다음 값으로 간주됩니다.
  - 10진 변수의 경우 '0'
  - 문자 변수의 경우 공백
  - 논리 변수의 경우 '0'
  - 정수 변수의 경우 '0'

  10진수와 문자 유형의 경우 초기 값은 지정하고 LEN 매개변수를 지정하지 않으면 디폴트 길이가 초기 값의 길이가 됩니다. *CHAR 유형의 경우 LEN 매개변수를 지

정하지 않으면 스트링의 최대 길이는 5000자입니다. *INT 또는 *UINT 유형의 경우 LEN 매개변수를 지정하지 않으면 디폴트 길이는 4입니다.

- 프로그램 DCL문에서 매개변수는 변수로 선언되어야 합니다.

리스트 또는 규정된 이름을 지정하기 위해 변수 사용

매개변수의 값은 리스트일 수 있습니다. 예를 들면, CHGLIBL(라이브러리 리스트 변경) 명령은 LIBL 매개변수에 각각 공백으로 분리되는 라이브러리 리스트를 필요로 합니다. 이 리스트 안의 요소들은 변수일 수 있습니다.

```
CHGLIBL LIBL(&LIB1 &LIB2 &LIB3)
```

리스트 안의 요소를 지정하는 데 변수가 사용되는 경우 각 요소가 각각 별도로 선언되어야 합니다.

```
DCL VAR(&LIB1) TYPE(*CHAR) LEN(10) VALUE(QTEMP)
DCL VAR(&LIB2) TYPE(*CHAR) LEN(10) VALUE(QGPL)
DCL VAR(&LIB3) TYPE(*CHAR) LEN(10) VALUE(DISTLIB)
CHGLIBL LIBL(&LIB1 &LIB2 &LIB3)
```

변수 요소는 리스트에서 문자 스트링으로 지정될 수 없습니다.

틀린 예:

```
DCL   VAR(&LIBS) TYPE(*CHAR) LEN(20) +
      VALUE('QTEMP QGPL DISTLIB')
CHGLIBL LIBL(&LIBS)
```

하나의 문자 스트링으로 표시되는 경우 시스템이 리스트를 개별 요소가 모인 리스트로 보지 않기 때문에 오류가 발생합니다.

각 규정자가 개별 변수로 선언된 경우에는 규정된 이름을 지정하는 데 변수를 사용할 수 있습니다.

```
DCL VAR(&PGM) TYPE(*CHAR) LEN(10)
DCL VAR(&LIB) TYPE(*CHAR) LEN(10)
CHGVAR VAR(&PGM) VALUE(MYPGM)
CHGVAR VAR(&LIB) VALUE(MYLIB)
.
.
.
DLTPGM PGM(&LIB/&PGM)
ENDPGM
```

위의 예에서는 프로그램명과 라이브러리명이 분리되어 선언되어 있습니다. 아래의 예에서와 같이 프로그램명과 라이브러리명을 하나의 변수로 지정할 수는 없습니다.

틀린 예:

```
DCL VAR(&PGM) TYPE(*CHAR) LEN(11)
CHGVAR VAR(&PGM) VALUE('MYLIB/MYPGM')
DLTPGM PGM(&PGM)
```

여기서, 시스템은 이 값을 두 개의 오브젝트(하나의 라이브러리와 하나의 오브젝트)로 간주하지 않고 하나의 문자 스트링으로 간주합니다. 규정된 이름이 문자 스트링 값을 가진 하나의 변수로 처리되어야 한다면, 내장 기능(built-in function)인 %SUBSTRING과 *TCAT 연결 기능(concatenation function)을 사용하여 오브젝트와 라이브러리명을 개별 변수로 할당할 수 있습니다. %SUBSTRING 기능을 사용한 예를 보려면 55 페이지의 『%SUBSTRING 내장 기능 사용』 부분과 제 9 장을 참조하십시오.

변수 안의 소문자

변수로 사용이 가능한 *LIBL과 같은 예약된 값은 특히 어포스트로피로 묶인 문자 스트링을 나타내는 경우 항상 대문자로 표현해야 합니다. 예를 들면, 명령어상의 라이브러리명을 변수로 대체하기 위한 코드는 다음과 같습니다.

```
DCL VAR(&LIB) TYPE(*CHAR) LEN(10) VALUE('*LIBL')
DLTPGM &LIB/MYPROG;
```

그러나 VALUE 매개변수를 다음과 같이 지정하면 *틀립니다*.

```
DCL VAR(&LIB) TYPE(*CHAR) LEN(10) VALUE('*libl')
```

어포스트로피가 없으면 이 VALUE 매개변수는 자동적으로 대문자로 변환되므로 VALUE 매개변수를 어포스트로피로 묶어서는 안됩니다. 매개변수가 화면에서 프로시듀어나 프로그램에 대한 입력으로 문자 스트링에 전달되고 화면에 소문자로 입력되는 경우에는 이 오류가 빈번히 발생합니다.

주: 위 단락에서는 대문자로 변환하는 것이 언어에 따라 다르다는 점을 고려하지 않았습니다. **주의**: 시스템에 의존하여 값을 대문자로 변환하면 예기치 않은 결과가 발생할 수 있습니다.

예약된 값이나 숫자 매개변수 값을 대체하는 변수

일부 CL 명령은 특정 매개변수에 숫자 또는 사전정의(예약) 값을 모두 허용합니다. 이 경우 문자 변수를 사용하여 명령 매개변수에 값을 표시할 수 있습니다.

명령의 각 매개변수는 일정한 유형의 값만 허용합니다. 매개변수는 정수, 문자 스트링, 예약된 값, 지정된 유형의 변수 또는 그 조합을 값으로 허용합니다. 어떤 유형의 값은 매개변수에 반드시 필요한 것이기도 합니다. 매개변수에 숫자값이 허용되고(명령에 *INT2, *INT4, *UINT2, *UINT4 또는 *DEC로 값이 정의되면) 또한 예약된 값(별표(*)가 앞에 오는 문자 스트링)이 허용되면 변수를 매개변수에 대한 값으로서 사용할 수 있습니다. 예약된 값을 사용하려면 변수를 TYPE(*CHAR)으로 선언해야 합니다.

예를 들면, CHGOUTQ(출력 대기행렬 변경) 명령에는 숫자(0에서 9까지) 또는 사전정의된 디폴트 *SAME을 가질 수 있는 작업 분리자(JOBSEP) 매개변수가 있습니다. 숫자와 사전정의 값 모두 허용이 가능하므로 JOBSEP 값을 문자 변수로 대체하는 CL 프로시듀어를 작성할 수도 있습니다.

```
   PGM
      DCL &NRESP *CHAR LEN(6)
      DCL &SEP *CHAR LEN(4)
      DCL &FILNAM *CHAR LEN(10)
      DCL &FILLIB *CHAR LEN(10)
      DCLF.....
      .
      .
      .
LOOP: SNDRCVF.....
      IF (&SEP *EQ IGNR) GOTO END
      ELSE IF (&SEP *EQ NONE) CHGVAR &NRESP '0'
      ELSE IF (&SEP *EQ NORM) CHGVAR &NRESP '1'
      ELSE IF (&SEP *EQ SAME) CHGVAR &NRESP '*SAME'
      CHGOUTQ OUTQ(&FILLIB/&FILNAM) JOBSEP(&NRESP)
         GOTO LOOP
END:  RETURN
ENDPGM
```

앞의 예에서 표시장치 사용자는 지정된 출력 대기행렬에 필요한 작업 분리자의 수를 설명하는 정보를 화면에 입력합니다. 변수 &NRESP는 숫자와 사전정의 값을 조작하는 문자 변수입니다.(어포스트로피의 사용에 주의하십시오.) CHGOUTQ 명령의 JOBSEP 매개변수는 이 값이 숫자값이나 사전정의 값으로 입력된 것처럼 인식합니다. 이 프로그램에서 사용된 화면 파일에 대한 DDS는 VALUES 키워드를 사용하여 사용자 응답을 IGNR, NONE, NORM 또는 SAME으로 제한합니다.

매개변수에 숫자 유형의 값(*INT2, *INT4, *UINT2, UINT4 또는 *DEC)이 허용되는 경우 예약된 값(예: *SAME)을 입력하지 않으면 이 매개변수에 10진 또는 정수 변수를 사용할 수 있습니다.

이 책의 제 9 장 부분에 명령 매개변수에 허용되는 값 유형에 대한 정보가 나와 있습니다. 자세한 정보는 **iSeries Information Center**의 *프로그래밍* 범주에서 *CL* 섹션을 참조하십시오.

이 기능의 다른 대안은 CL 프로시듀어 안에서 프롬터를 사용하는 것입니다.

변수 값의 변경

CHGVAR(변수 변경) 명령을 사용하면 CL 변수의 값을 변경할 수 있습니다. 값은 다음과 같이 변경될 수 있습니다.

- 상수로 변경.

  ```
  CHGVAR   VAR(&INVCMPLT)   VALUE(0)
  ```

  또는

  ```
  CHGVAR   &INVCMPLT  0
  ```

  &INVCMPLT는 0으로 설정됩니다.

- 다른 변수의 값으로 변경.

```
CHGVAR   VAR(&A)   VALUE(&B)
```

또는

```
CHGVAR   &A  &B
```

&A는 변수 &B의 값으로 설정됩니다.

- 평가가 이루어진 후 표현식의 값으로 변경.

```
CHGVAR   VAR(&A)   VALUE(&A + 1)
```

또는

```
CHGVAR   &A  (&A + 1)
```

&A의 값은 1씩 커집니다.

- 내장 기능 %SST에 의해 산출된 값으로 변경(자세한 내용은 55 페이지의 『%SUBSTRING 내장 기능 사용』 부분 참조).

```
CHGVAR   VAR(&A)   VALUE(%SST(&B 1 5))
```

&A는 변수 &B의 값 중 처음 5자로 설정됩니다.

- 내장 기능 %SWITCH에 의해 산출된 값으로 변경(자세히 알려면 58 페이지의 『%SWITCH 내장 기능 사용』 부분 참조).

```
CHGVAR  VAR(&A)  VALUE(%SWITCH(0XX111X0))
```

작업 스위치 1과 8이 0이고 작업 스위치 4, 5, 6이 1인 경우 &A가 1로 설정되며, 그렇지 않으면 &A가 0으로 설정됩니다.

- 내장 기능 %BIN에 의해 산출된 값으로 변경(자세히 알려면 53 페이지의 『%BINARY 내장 기능 사용』 부분 참조).

```
CHGVAR  VAR(&A)  VALUE(%BIN((%B 1 4))
```

변수 &B의 처음 4자는 등가의 10진수로 변환되어 변수 &A에 저장됩니다.

CHGVAR 명령을 사용하여 로컬 자료 영역을 검색하고 변경할 수 있습니다. 예를 들어, 다음 명령은 로컬 자료 영역에 10바이트를 공백으로 만들고, 로컬 자료 영역 부분을 검색합니다.

```
CHGVAR %SST(*LDA 1 10) ' '

CHGVAR &A %SST(*LDA 1 10)
```

다음 표에서는 값(리터럴 또는 변수)에서 변수로 지정한 유효한 항목을 보여줍니다.

표 1. 값에서 변수로 지정한 유효한 지정 항목

	논리 값	문자 값	십진 값	부호화된 정수 값	부호 없는 정수 값
논리 변수	X				
문자 변수	X	X	X	X	X
십진 변수		X	X	X	X
부호화된 정수 변수		X	X	X	X
부호 없는 정수 변수		X	X	X	X

주:

1. 문자 변수에 숫자 값을 지정할 때는 다음 사항에 유의해야 합니다.
   - 문자 변수의 값은 우측으로 정렬되고, 필요한 경우 선행 0이 앞에 채워집니다.
   - 문자 변수는 필요 시 소수점과 빼기 부호(-)가 들어갈 수 있을 만큼의 충분한 길이여야 합니다.
   - 빼기 부호 사용 시, 이 부호는 값의 맨 왼쪽에 놓입니다.

   예를 들어, &A는 10진 변수 &B의 값으로 변경될 문자 변수입니다. &A의 길이는 6입니다. &B의 길이와 소수 자릿수는 각각 5와 2입니다. &B의 현재값은 123입니다. &A의 결과값은 123.00입니다.
2. 숫자 변수에 문자 값을 지정할 때는 다음 사항에 유의해야 합니다.
   - 소수점은 문자 값의 소수점 위치에 따라 판별됩니다. 문자 값에 소수점이 없으면, 소수점은 값의 가장 오른쪽 위치에 놓입니다.
   - 문자 값에는 값 왼쪽에 빼기 부호(-)나 더하기 부호(+)가 있을 수 있으며, 사이 공백은 허용되지 않습니다. 문자 값에 부호가 없으면 양수로 간주됩니다.
   - 숫자 변수에 포함될 수 있는 자릿수보다 더 많은 자릿수가 문자 값의 소수점 오른쪽에 있는 경우 그 값이 십진 변수이면 자릿수가 절단되고 정수 변수이면 반올림됩니다. 그러나 초과된 자릿수가 소수점 왼쪽에 있으면 자릿수가 절단되지 않고 오류가 발생합니다.

     예를 들면, &C는 문자 변수 &D의 값으로 변경될 10진 변수이고, &C의 길이는 소수 자릿수 2를 포함하여 5입니다. &D의 길이는 10이고 현재 값은 +123.1bbbb(b는 공백을 의미)입니다. 이 경우 &C의 결과 값은 123.10입니다.

명령 매개변수상의 후미 공백

어떤 명령 매개변수는 매개변수 값 VARY(*YES)로 정의됩니다. 이 매개변수 값에 의해 전달될 값의 길이는 어포스트로피 안의 문자 수입니다. 이렇게 정의된 매개변수 값을 지정하기 위해 CL 변수를 사용할 때 시스템은 후미 공백을 제거한 후 명령 프로세

서 프로그램으로 전달될 변수의 길이를 결정합니다. 후미 공백이 존재하고 이 공백이 매개변수에 중요한 의미가 있다면, 전달되는 길이에 이 공백이 포함될 수 있도록 특별한 조치를 수행해야 합니다. 대부분의 명령 매개변수들은 이와 같은 상태가 발생하지 않는 방식으로 정의되어 사용됩니다. 이러한 상태가 발생할 수 있는 매개변수가 정의된 예로는 OVRDBF 명령에서 POSITION 매개변수의 키값 요소를 들 수 있습니다.

이러한 상태가 발생하면 어포스트로피로 매개변수 값을 분리하는 명령 스트링을 구성하고, QCMDEXC 또는 QCAPCMD로 스트링을 전달하여 처리함으로써, 원하는 매개변수 결과를 얻을 수 있습니다.

다음은 후미 공백이 키값의 일부로 포함되도록 OVRDBF 명령을 실행하는 데 사용될 수 있는 프로그램의 예입니다. 매개변수 VARY(*YES)를 사용하여 매개변수를 정의한 다른 명령에도 이것과 동일한 방법을 사용할 수 있으며, 후미 공백은 매개변수와 함께 전달되어야 합니다.

```
            PGM        PARM(&KEYVAL &LEN)
/*   PROGRAM TO SHOW HOW TO SPECIFY A KEY VALUE WITH TRAILING        */
/*      BLANKS AS PART OF THE POSITION PARAMETER ON THE OVRDBF       */
/*      COMMAND IN A CL PROGRAM.                                     */
/*   THE KEY VALUE ELEMENT OF THE POSITION PARAMETER OF THE OVRDBF   */
/*      COMMAND IS DEFINED USING THE VARY(*YES) PARAMETER.           */
/*      THE DESCRIPTION OF THIS PARAMETER ON THE ELEM COMMAND        */
/*      DEFINITION STATEMENT SPECIFIES THAT IF A PARAMETER           */
/*      DEFINED IN THIS WAY IS SPECIFIED AS A CL VARIABLE THE        */
/*      LENGTH IS PASSED AS THE VARIABLE WITH TRAILING BLANKS        */
/*      REMOVED. A CALL TO QCMDEXC USING APOSTROPHES TO DELIMIT      */
/*      THE LENGTH OF THE KEY VALUE CAN BE USED TO CIRCUMVENT        */
/*      THIS ACTION.                                                 */
/*    PARAMETERS--                                                   */
            DCL        VAR(&KEYVAL) TYPE(*CHAR) LEN(32)  /*  THE VALUE +
                         OF THE REQUESTED KEY. NOTE IT IS DEFINED AS +
                         32 CHAR.  */
            DCL        VAR(&LEN) TYPE(*INT)              /*  THE LENGTH +
                         OF THE KEY VALUE TO BE USED. ANY VALUE OF  +
                         1 TO 32 CAN BE USED  */
/*   THE STRING TO BE FINISHED FOR THE OVERRIDE COMMAND TO BE        */
/*        PASSED TO QCMDEXC (NOTE 2 APOSTROPHES TO GET ONE).         */
            DCL        VAR(&STRING) TYPE(*CHAR) LEN(100) +
                         VALUE('OVRDBF FILE(X3) POSITION(*KEY 1 FMT1  '' ')
/*   POSITION MARKER  123456789 123456789 123456789 123456789       */
            DCL        VAR(&END) TYPE(*DEC) LEN(15 5)  /*  A VARIABLE +
                         TO CALCULATE THE END OF THE KEY IN &STRING  */

            CHGVAR     VAR(%SST(&STRING 40 &LEN)) VALUE(&KEYVAL)  /*  +
                         PUT THE KEY VALUE INTO COMMAND STRING FOR +
                         QCMDEXC IMMEDIATELY AFTER THE APOSTROPHE.  */
            CHGVAR     VAR(&END) VALUE(&LEN + 40)  /*  POSITION AFTER +
                         LAST CHARACTER OF KEY VALUE  */
            CHGVAR     VAR(%SST(&STRING &END 2)) VALUE(')')  /*  PUT +
                         A CLOSING APOSTROPHE & PAREN TO END  +
                         PARAMETER  */
            CALL       PGM(QCMDEXC) PARM(&STRING 100)  /*  CALL TO +
                         PROCESS THE COMMAND   */
    ENDPGM
```

주: VARY(*YES)와 RTNVAL(*YES)를 사용하고 CL 변수를 전달하면 CL 변수 안의 자료 길이가 아닌 변수 길이가 전달됩니다.

CL 프로시듀어에 주석 작성

CL 프로시듀어에 주석을 작성하거나 프로시듀어의 명령에 주석을 추가하고자 할 경우에는 문자쌍 /*와 */를 사용하십시오. 두 개의 이 기호 사이에 주석이 작성됩니다.

명령 스트링의 맨 처음 2자리에 /*가 없으면, 주석 시작 분리문자 /*에 3자리가 필요합니다. 맨 처음 2자리에 /*가 있을 경우에는 명령 앞에 공백이 없어도 /*를 사용할 수 있습니다.

3자로 된 주석 시작 분리문자로는 다음 중 어느 것이나 가능합니다(b는 공백을 의미).

```
  /*b
b/*
/**
```

그러므로 주석 시작 분리문자는 네 가지 방법으로 입력이 가능합니다. 주석 시작 분리문자 /*는 다음과 같이 사용할 수 있습니다.

- 명령 스트링의 맨 처음에서 시작할 수 있습니다.
- 앞에 공백이 올 수 있습니다.
- 뒤에 공백이 올 수 있습니다.
- 뒤에 별표(*)가 올 수 있습니다(/**).

주: 주석은 주석 안에 내장시킬 수 없습니다.

다음 프로시듀어는 메뉴 옵션 세트에 사용할 수 있는 사용자 응답을 설명하기 위해 주석을 작성한 예입니다.

```
        PGM        /* ORD040C ORDER DEPT GENERAL MENU */
       DCLF FILE(ORD040CD)
START: SNDRCVF RCDFMT(MENU)
        SELECT
          WHEN (&RESP=1) THEN(CALL CUS210)   /* CUSTOMER INQUIRY     */
          WHEN (&RESP=2) THEN(CALL ITM210)   /* ITEM INQUIRY         */
          WHEN (&RESP=3) THEN(CALL CUS210)   /* CUSTOMER NAME SEARCH */
          WHEN (&RESP=4) THEN(CALL ORD215)   /* ORDERS BY CUST       */
          WHEN (&RESP=5) THEN(CALL ORD220)   /* EXISTING ORDER       */
          WHEN (&RESP=6) THEN(CALL ORD410C)  /* ORDER ENTRY          */
          WHEN (&RESP=7) THEN(RETURN)
        ENDSELECT
       GOTO START
ENDPGM
```

CL 프로시듀어 안에서의 처리 제어

CL 프로시듀어의 명령은 연속적인 순서로 처리됩니다. 각 명령은 위치한 순서에 따라 차례로 처리됩니다. 사용자는 프로시듀어에 있는 논리 흐름 변경 명령을 사용하여 이러한 연속적 처리를 변경할 수 있습니다. 이러한 명령은 조건부이거나 무조건적일 수 있습니다.

무조건 분기는 분기 명령이 처리되는 시점에서 어떤 조건이 존재하는지에 상관없이 처리가 프로시듀어 안에서 임의의 위치에 있는 명령 또는 명령 세트로 분기되도록 사용자가 지시할 수 있다는 의미입니다. 무조건적 처리 명령은 다음과 같습니다.

- GOTO
- ITERATE

- LEAVE

조건 분기는 지정된 특정 조건하의 프로시듀어에서 연속되지 않은 섹션이나 명령으로 처리가 분기된다는 의미입니다. 분기는 프로시듀어 안의 어떤 명령문으로나 가능합니다. 이것은 지정 조건이 참(true)일 때만 분기가 발생하므로 이를 조건부 처리라고 합니다. 조건부 처리는 일반적으로 IF 명령과 관련이 있습니다. ELSE 명령을 사용하면 조건이 참이 아닌 경우 대체 처리를 지정할 수 있습니다. 단순 DO 명령을 사용하면 지정된 조건에서 항상 그룹 형태로 함께 처리되는 명령 그룹을 작성할 수 있습니다. 조건부 처리 명령은 다음과 같습니다.

- IF 및 THEN
- SELECT, WHEN 및 OTHERWISE
- DOFOR
- DOWHILE
- DOUNTIL

GOTO 명령 및 레이블 사용

GOTO 명령은 무조건 분기를 처리합니다. GOTO 명령이 있으면 GOTO 명령을 만날 때마다 프로시듀어의 다른 부분(레이블로 식별됨)으로 처리가 지정됩니다. 이 분기는 표현식의 계산 결과에 따라 이루어지는 것이 아닙니다. 레이블된 명령문으로 분기한 후 그 명령문에서 처리가 시작되어 연속적인 순서로 계속되며, 다른 명령어가 특별히 처리 방향을 바꾸지 않는 한 처리는 GOTO 명령으로 리턴되지 않습니다. 분기는 앞이나 뒤 어느 쪽으로나 가능합니다. GOTO를 사용하여 프로시듀어 외부의 레이블로 진행할 수는 없습니다. GOTO 명령에는 하나의 매개변수가 있으며, 이 매개변수에 분기될 명령문 레이블이 들어 있습니다.

```
GOTO CMDLBL(label)
```

레이블은 GOTO 명령으로 처리가 넘어가게 될 프로시듀어 안의 명령문을 가리킵니다. GOTO 명령을 사용하려면 분기할 명령에 레이블이 있어야 합니다.

```
            PGM
          .
          .
          .
START: SNDRCVF RCDFMT(MENU)
          IF (&RESP=1) THEN(CALL CUS210)
          .
          .
          .
       GOTO START
          .
          .
          .
    ENDPGM
```

앞의 예에서 레이블은 START입니다. 레이블은 10자 이하이며, 바로 다음에 콜론(:)이 와야 하지만, 레이블과 명령어 사이에는 공백도 가능합니다.

IF 명령 사용

IF 명령은 참인 경우 프로시듀어에서 실행될 일부 명령문이나 명령문 그룹을 지정하는 조건을 서술하는 데 사용됩니다. ELSE 명령은 IF 명령과 함께 사용되어 IF 명령으로 표현된 조건이 거짓일 경우에 실행될 명령문이나 명령문 그룹을 지정합니다.

이 명령에는 테스트될(참 또는 거짓) 표현식과 표현식이 참인 경우의 조치를 지정하는 THEN 매개변수가 포함되어 있습니다. 다음과 같이 IF 명령을 사용할 수 있습니다.

```
IF COND(logical-expression) THEN(CL-command)
```

COND 매개변수상의 논리식은 하나의 논리 변수나 상수여야 하며, 그 식이 참 또는 거짓이라고 평가할 수 있는 둘 또는 그 이상의 피연산자 간의 관계를 설명하는 것이어야 합니다. 논리식의 구성에 대해 자세히 알려면 49 페이지의 『*AND, *OR 및 *NOT 연산자 사용』 부분을 참조하십시오.

논리식의 조건이 참으로 평가되면 프로시듀어는 THEN 매개변수상의 CL 명령을 처리합니다. 이 명령은 하나의 명령일 수도 있고 명령 그룹일 수도 있습니다(42 페이지의 『DO 명령과 DO 그룹 사용』 부분 참조). 조건이 참이 아니면, 프로시듀어는 다음과 같이 순차적으로 명령을 수행합니다.

명령어상에서 COND와 THEN은 모두 키워드이며, 위치상의 항목인 경우 생략할 수 있습니다. 다음은 명령 구문을 올바로 사용한 예입니다.

```
IF COND(&RESP=1) THEN(CALL CUS210)
IF (&A *EQ &B) THEN(GOTO LABEL)
IF (&A=&B) GOTO LABEL
```

명령어(IF)와 키워드(COND) 또는 값(&A) 사이에 공백이 있어야 합니다. 키워드 사이에는 공백이 허용되지 않으며, 키워드와 값을 둘러싼 왼쪽 괄호 사이에는 공백이 없어야 합니다.

아래의 예에서는 IF 명령을 사용한 조건부 처리를 보여주고 있습니다. 분기 처리 방법은 IF 명령의 논리식 결과에 따라 결정됩니다. 예를 들어, 다음 코드의 처음에 있는 &A의 값이 2이고 &C의 값이 4라고 가정해 보겠습니다.

```
             IF (&A=2) THEN(GOTO FINAL)
       IF (&A=3) THEN(CHGVAR &C 5)
       .
       .
       .
FINAL: IF (&C=5) CALL PROGA
    ENDPGM
```

이 경우 프로시듀어는 FINAL로 분기하기 위해 중간 코드를 건너뛰기 전에 첫 번째 IF 명령을 처리합니다. 두 번째 IF 명령으로는 리턴하지 않습니다. FINAL에서 &C=5에 대

해 테스트가 실패하므로 PROGA가 호출되지 않습니다. 그러면 프로시듀어는 프로시듀어 끝을 신호하는 다음 명령 ENDPGM을 처리하고 호출 프로시듀어로 제어를 리턴시킵니다.

동일한 코드를 사용해도 변수의 초기값이 다르면, 논리 처리가 달라집니다. 예를 들어, 이 코드의 처음에 있는 &A의 값이 3이고 &C의 값이 4이면, 첫 번째 IF문은 거짓으로 평가됩니다. GOTO FINAL 명령을 처리하는 대신, 프로시듀어는 첫 번째 IF문을 무시하고 다음의 IF문으로 이동합니다. 두 번째 IF문은 참으로 평가되고 &C의 값은 5로 변경됩니다. 그런 다음, 여기에서는 생략되어 있는 후속 명령문이 연속적으로 처리됩니다. 처리가 마지막 IF문에 도달하면 조건 &C=5가 참으로 평가되어 PROGA가 호출됩니다.

일련의 연속적인 IF문이 독립적으로 수행됩니다. 다음 예를 보십시오.

```
            PGM /*  IFFY  */
      DCL &A..
      DCL &B..
      DCL &C..
      DCL &D..
      DCL &AREA *CHAR LEN(5) VALUE(YESNO)
      DCL &RESP..
      IF (&A=&B) THEN(GOTO END)   /*  IF #1  */
      IF (&C=&D) THEN(CALL PGMA)   /*  IF #2  */
      IF (&RESP=1) THEN(CHGVAR &C 2)  /*  IF #3  */
      IF (%SUBSTRING(&AREA 1 3) *EQ YES) THEN(CALL PGMB)  /*  IF #4  */
      CHGVAR &B &C
      .
      .
      .
END:        ENDPGM
```

이 예에서 &A가 &B와 같지 않으면 그 다음 명령문을 실행합니다. &C가 &D와 같으면, PGMA가 호출됩니다. PGMA로부터 리턴한 뒤에 세 번째 IF문을 처리합니다. 그리고 계속 이런 식으로 진행됩니다. 이러한 단순한 순차 IF문 및 ELSE와 함께 IF를 사용하는 경우 또는 이 장의 뒷부분에 설명되어 있는 내장 IF 명령을 사용하는 경우에서의 논리와 처리상의 차이점에 주목하십시오(39 페이지의 『ELSE 명령 사용』 부분과 41 페이지의 『내장 IF 명령 사용』 부분 참조).

내장 명령은 다른 명령의 매개변수에 완전히 포함되는 명령입니다. 아래의 예에서 CHGVAR 명령과 DO 명령은 내장 명령입니다.

```
IF (&A *EQ &B) THEN(CHGVAR &A (&A+1))


IF (&B *EQ &C) THEN(DO)
                     .
                     .
                     .
                   ENDDO
```

ELSE 명령 사용

ELSE 명령은 관련 IF 명령의 조건이 거짓인 경우 대체 처리를 지정하는 방법입니다.

IF 명령은 ELSE 명령 없이도 사용될 수 있습니다.

```
IF (&A=&B) THEN(CALLPRC PROCA)
CALLPRC PROCB
```

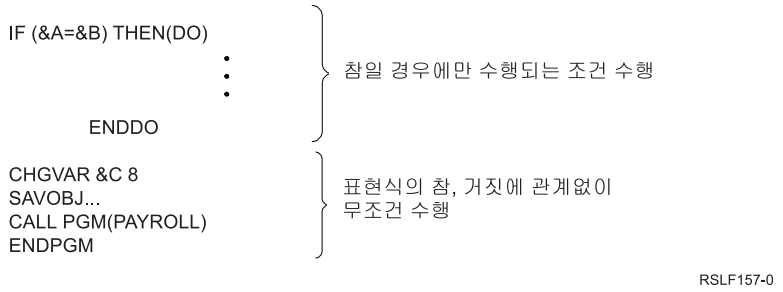
이 경우 PROCA는 &A=&B일 때만 호출되지만, PROCB는 항상 호출됩니다.

이 프로시듀어에서 ELSE 명령을 사용하면 논리 처리가 변경됩니다. 다음 예에서 &A=&B이면 PROCA가 호출되고, PROCB는 호출되지 않습니다. 표현식 &A=&B가 참이 아니면, PROCB가 호출됩니다.

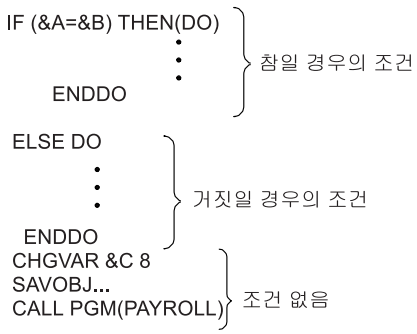
```
IF (&A=&B) THEN(CALLPRC PROCA)
ELSE CMD(CALLPRC PROCB)
CHGVAR &C 8
```

IF 표현식의 평가가 거짓일 때 명백한 분기(즉, 배타적인 선택 분기)를 해야 할 경우에는 ELSE 명령을 사용해야 합니다.

ELSE 명령은 DO 그룹과 결합하여 가장 유효하게 사용됩니다. 아래의 예에서 DO 그룹은 IF 표현식의 평가에 따라 수행되지 않을 수도 있으나 나머지 명령은 항상 처리됩니다.



ELSE 명령을 사용하면 표현식이 참이 아닐 경우에만 명령이 처리되어 논리적 대체가 완료되도록 지정할 수 있습니다.



RV2W275-1

각 ELSE 명령은 그 앞에 관련 IF 명령이 있어야 합니다. IF 명령이 내포된 경우 ELSE 명령은 아직 다른 ELSE 명령과 대응되지 않은 맨 안쪽의 IF 명령과 대응됩니다.

```
IF ... THEN ...
                    IF ...THEN(DO)
                    IF ...THEN(DO)
                    .
                    .
                    .
 ENDDO
        ELSE DO
                    IF ...THEN(DO)
                        .
                        .
                        .
 ENDDO
        ELSE DO
                        .
                        .
                        .
 ENDDO
 ENDDO
ELSE IF ... THEN ...
IF ... THEN ...
IF ... THEN ...
```

대응 ELSE 명령에 대한 프로시듀어 검토 시, 항상 가장 안쪽 세트부터 시작하십시오.

ELSE 명령은 일련의 상호 배타적인 옵션의 테스트에도 사용할 수 있습니다. 다음 예에서는 첫 번째의 성공적인 IF 테스트 이후 내장 명령이 처리되고 프로시듀어가 RCLRSC 명령을 처리합니다.

```
IF  COND(&OPTION=1) THEN(CALLPRC PRC(ADDREC))
 ELSE    CMD(IF COND(&OPTION=2) THEN(CALLPRC PRC(DSPFILE)))
  ELSE   CMD(IF COND(&OPTION=3) THEN(CALLPRC PRC(PRINTFILE)))
    ELSE CMD(IF COND(&OPTION=4) THEN(CALLPRC PRC(DUMP)))
RCLRSC
          RETURN
```

내장 IF 명령 사용

IF 명령은 또 다른 IF 명령에 내장될 수 있습니다. 이는 평가가 참일 때 처리되는 명령(THEN 매개변수에 있는 CL 명령) 자체가 또 다른 IF 명령인 경우입니다.

```
IF (&A=&B) THEN(IF (&C=&D) THEN(GOTO END))
       GOTO START
```

이러한 내장 명령은 명령이나 명령 그룹이 수행되기 전에 몇 가지 조건이 충족되어야 할 경우 유용하게 사용됩니다. 앞의 예에서 첫 번째 표현식이 참이면, 시스템은 첫 번째 THEN 매개변수를 읽고, 여기서, &C=&D 표현식이 참으로 평가되면 시스템은 두 번째 THEN 매개변수에 있는 명령인 GOTO END를 처리합니다. GOTO END 명령을 처리하려면 두 개의 표현식이 모두 참이어야 합니다. 둘 중 하나라도 거짓이면, GOTO START 명령이 수행됩니다. 표현식과 명령을 구성하고 있는 괄호의 사용에 주의하십시오.

CL 프로그래밍에서는 이러한 내장 처리를 최대 25단계까지 허용합니다.

내장 단계가 증가하고 논리가 점점 더 복잡해지면, 관계를 보다 분명히 표시하기 위해 자유 양식으로 코드를 입력할 수 있습니다.

```
   PGM
DCL &A *DEC 1
DCL &B *CHAR 2
DCL &RESP *DEC 1
IF (&RESP=1) +
      IF (&A=5) +
             IF (&B=NO) THEN(DO)
                        .
                        .
                        .
 ENDDO
CHGVAR &A VALUE(8)
CALL PGM(DAILY)
    ENDPGM
```

앞에 나온 일련의 IF는 하나의 내장 명령으로 처리됩니다. IF 조건 중 하나라도 거짓이면, 처리는 코드의 나머지 부분(CHGVAR과 후속 명령)으로 분기됩니다. 이 코드의 목적이 DO 그룹이 처리되기 위해 모두 참이어야 하는 일련의 조건을 누적하기 위한 것이면, 한 명령에서 여러 개의 표현식에 *AND를 사용하여 보다 간단히 할 수 있습니다. 49 페이지의 『*AND, *OR 및 *NOT 연산자 사용』 부분을 참조하십시오.

그러나 때때로 조건이 거짓인 경우를 따라 분기해야 하는 경우도 있습니다. 각 내장 IF 명령에 ELSE 명령을 추가해서 이를 수행할 수 있습니다.

```
   PGM
DCL &A ...
DCL &B ...
DCL &RESP ...
IF (&RESP=1) +
      IF (&A=5) +
             IF (&B=NO) THEN(DO)
```

```
                              .
                              .
                              .
                              SNDPGMMSG ...
                              .
                              .
                              .
 ENDDO
ELSE CALLPRC PROCA
      ELSE CALLPRC PROCB
CHGVAR &A 8
CALLPRC PROC(DAILY)
    ENDPGM
```

여기에서 모든 조건이 참이면, SNDPGMMSG 명령이 처리된 후 CHGVAR 명령이 처리됩니다. 첫 번째와 두 번째 조건(&RESP=1 및 &A=5)이 참이지만, 세 번째 조건(&B=NO)이 거짓이면, PROCA가 호출됩니다. PROCA가 리턴되면 CHGVAR 명령이 처리됩니다. 두 번째 조건이 거짓이면, PROCB가 호출되고(&B=NO가 테스트되지 않음), 그 다음에 CHGVAR 명령이 호출됩니다. 마지막으로 &RESP가 1이 아니면 CHGVAR 명령이 곧바로 처리됩니다. 이 예에서는 ELSE 명령을 사용하여 각 테스트마다 서로 다른 분기를 제공합니다.

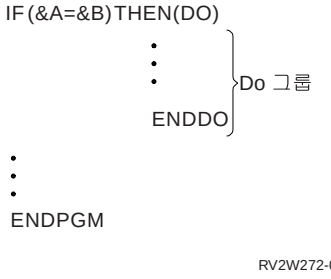
주: 아래의 세 가지 예는 앞의 예에 있는 내장 IF 명령과 구문상 기능이 같은 명령입니다.

```
IF (&RESP=1) THEN(IF (&A=5) THEN(IF (&B=NO) THEN(DO)))

IF (&RESP=1) THEN +
           (IF (&A=5) THEN +
                        (IF (&B=NO) THEN(DO)))

IF (&RESP=1) +
         (IF (&A=5) +
               (IF (&B=NO) THEN(DO)))
```

DO 명령과 DO 그룹 사용

DO 명령은 명령 그룹을 함께 처리할 수 있게 합니다. 그룹이란 DO 명령과 해당 ENDDO 명령 사이의 모든 명령을 의미합니다.

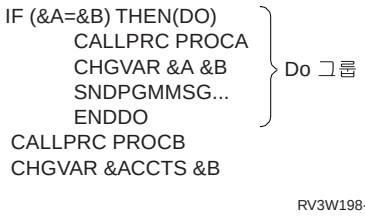
일반적으로 그룹의 처리는 관련된 명령의 평가 결과에 따라 다릅니다. DO 그룹은 거의 대부분 IF, ELSE 또는 MONMSG 명령과 연관되어 사용됩니다. 다음 예를 보십시오.



RV2W272-0

논리식(&A=&B)이 참이면, DO 그룹이 처리됩니다. 표현식이 참이 아니면 ENDDO 명령 이후부터 처리가 시작됩니다. 즉, DO 그룹을 건너뜁니다.

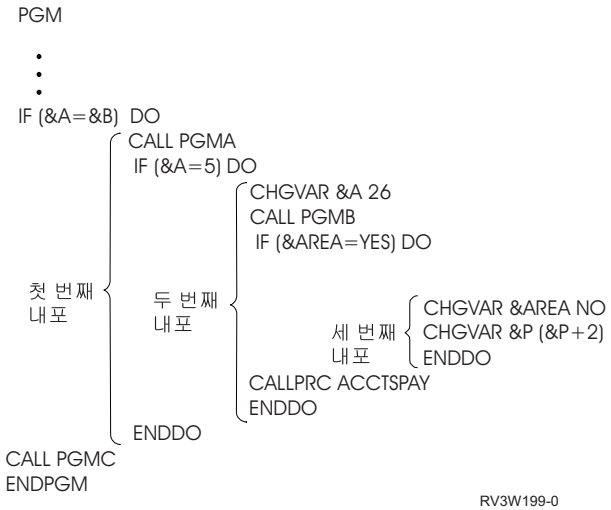
다음 프로시듀어에서 &A가 &B와 같지 않으면 시스템은 PROCB를 호출합니다. PROCA는 호출되지 않으며 DO 그룹 안에 있는 다른 명령도 처리되지 않습니다.



RV3W198-0

DO 그룹은 최대 25단계의 내포 레벨까지 다른 DO 그룹 내에서 내포될 수 있습니다.

다음의 예에서는 3단계까지 내포되어 있습니다. 각각의 DO 그룹이 ENDDO 명령에 의해 완료되는 방법에 주의하십시오.



RV3W199-0

이 예에서 1단계 내포의 &A가 5가 아니면, PGMC가 호출됩니다. &A가 5이면 두 번째 DO 그룹의 명령문이 처리됩니다. 두 번째 DO 그룹의 &AREA가 YES가 아니면 처리가 DO 그룹 이후의 다음 명령으로 이동하므로 프로시듀어 ACCTSPAY가 호출됩니다.

CL 컴파일러는 DO 그룹의 시작과 끝을 표시하지 않습니다. CL 컴파일러가 균형이 맞지 않는 조건을 오류로 지적해도, 실제 오류를 검출하기는 쉽지 않습니다.

DOUNTIL 명령 사용

Do Until(DOUNTIL) 명령은 CL 명령 그룹을 1회 이상 처리합니다. 명령 그룹은 DOUNTIL 및 일치하는 ENDDO 명령 사이의 명령을 의미합니다.

명령 그룹을 처리한 후, 명시된 조건을 평가합니다. 조건이 참이면 DOUNTIL 그룹은 종료되고 연관된 ENDDO 뒤에 나오는 다음 명령의 처리를 재개합니다. 조건이 거짓이면 그룹은 그룹의 첫 번째 명령을 처리합니다.

COND 매개변수상의 논리식은 하나의 논리 변수나 상수여야 하며, 그 식이 참 또는 거짓이라고 평가할 수 있는 둘 또는 그 이상의 피연산자 간의 관계를 설명하는 것이어야 합니다. 논리식의 구성에 대한 자세한 내용은 49 페이지의 『*AND, *OR 및 *NOT 연산자 사용』 부분을 참조하십시오.

다음은 DOUNTIL 명령을 사용한 조건부 처리의 예입니다.

```
DOUNTIL (&LGL)
  .
  .
  .
  CHGVAR &INT (&INT + 1)
  IF (&INT *GT 5) (CHGVAR &LGL '1')
 ENDDO
```

DOUNTIL 그룹의 본문은 1회 이상 실행됩니다. &INT 변수의 초기 값이 5 이상이면 &LGL은 처음부터 참으로 설정되고 그룹 종료 시 표현식이 평가될 때 ENDDO 뒷부분을 처리합니다. 초기 값이 5 미만이면 그룹 본문은 &INT 값이 5보다 커질 때까지 계속 반복하고 &LGL 값은 참으로 변경됩니다.

LEAVE 명령은 DOUNTIL 그룹을 종료하고 ENDDO 뒤에서 실행을 재개하는 데 사용할 수 있으며 ITERATE 명령은 그룹의 나머지 명령을 건너뛰고 명시된 조건을 즉시 평가하는 데 사용할 수 있습니다.

DOWHILE 명령 사용

DOWHILE 명령을 사용하면 논리식의 값이 참인 때에 명령 그룹을 0회 이상 처리할 수 있습니다. 참인 경우 DOWHILE 명령은 프로시듀어에서 실행할 명령이나 명령 그룹을 지정하는 조건을 서술하는 데 사용됩니다. 명령 그룹은 DOWHILE 및 일치하는 ENDDO 명령 사이의 명령을 의미합니다.

명령 그룹을 처리한 후, 명시된 조건을 평가합니다. 조건이 거짓이면 DOWHILE 그룹은 종료되고 연관된 ENDDO 뒤에 나오는 다음 명령의 처리를 재개합니다. 조건이 참이면 그룹은 그룹의 첫 번째 명령을 처리합니다. ENDDO 명령에 도달하면 조건을 다시 평가하기 위해 DOWHILE 명령으로 제어가 리턴됩니다.

COND 매개변수상의 논리식(logical expression)은 하나의 논리 변수나 상수여야 하며, 그 식이 참 또는 거짓이라고 평가할 수 있는 둘 또는 그 이상의 피연산자 간의 관계를 설명하는 것이어야 합니다. 논리식의 구성에 대한 자세한 내용은 49 페이지의 『*AND, *OR 및 *NOT 연산자 사용』 부분을 참조하십시오.

다음은 DOWHILE 명령을 사용한 조건부 처리의 예입니다.

```
DOWHILE (&LGL)
  .
  .
  .
  IF (&INT *EQ 2) (CHGVAR &LGL '0')
                   ENDDO
```

DOWHILE 그룹이 처리되면 명시된 조건이 평가됩니다. 조건이 참이면 DOWHILE 그룹의 명령 그룹이 처리됩니다. 조건이 거짓이면 연관된 ENDDO 명령 뒤에 나오는 명령을 처리합니다.

&LGL 값이 참이면 DOWHILE 그룹의 명령은 &INT가 2와 같아질 때까지 실행되어 &LGL 변수 값을 거짓으로 설정합니다.

LEAVE 명령은 DOWHILE 그룹을 종료하고 뒤의 ENDDO 처리를 재개하는 데 사용할 수 있으며 ITERATE 명령은 그룹의 나머지 명령을 건너뛰고 명시된 조건을 즉시 평가하는 데 사용할 수 있습니다.

DOFOR 명령 사용

DOFOR 명령을 사용하면 명령 그룹을 지정된 횟수만큼 처리할 수 있습니다.

DOFOR 명령은 변수, 해당 초기 값, 증가량 또는 감소량 및 단말 값 조건을 지정합니다. DOFOR 명령의 형식은 다음과 같습니다.

```
DOFOR VAR(정수-변수) FROM(초기-값) TO(종료-값) BY(정수-상수)
```

DOFOR 그룹 처리가 시작되면 VAR 매개변수에 지정된 정수-변수는 FROM 매개변수에 지정된 초기-값으로 초기화됩니다. 정수-변수 값은 TO 매개변수에 지정된 종료-값과 비교됩니다. BY 매개변수의 정수-상수가 양수이면 비교 시 정수-변수가 종료-값보다 큰지 여부를 검사하고 BY 매개변수의 정수-상수가 음수이면 비교 시 정수-변수가 종료-값보다 작은지 여부를 검사합니다.

조건이 참이 아니면 DOFOR 그룹의 본문이 처리됩니다. ENDDO에 도달하면 BY 매개변수의 정수-상수가 정수-값에 추가되고 조건이 다시 평가됩니다.

다음은 DOFOR 명령을 사용한 조건부 처리의 예입니다.

```
CHGVAR &INT2 0
DOFOR VAR(&INT) FROM(2) TO(4) BY(1)
  .
  .
```

```
  .
  CHGVAR &INT2 (&INT2 + &INT)
 ENDDO
/* &INT2 = 9 after running the DOFOR group 3 times */
```

DOFOR 그룹을 처리할 때 &INT는 2로 초기화되고 4보다 큰지 확인하기 위해 &INT 값이 검사됩니다. 값이 4보다 작으면 그룹의 본문이 처리됩니다. 그룹을 두 번째 반복할 때 &INT에 1이 더해져서 검사가 반복됩니다. 값이 4보다 작으면 DOFOR 그룹이 다시 처리됩니다. ENDDO에 두 번째로 도달하면 &INT 값에 다시 1이 증가합니다. 이제 &INT 값은 4입니다. &INT가 4보다 작거나 같으면 DOFOR 그룹이 다시 처리됩니다. ENDDO에 세 번째로 도달하면 &INT 값에 다시 1이 증가합니다. 이번에는 값이 5이고 ENDDO 뒤의 명령을 처리합니다.

LEAVE 명령은 DOFOR 그룹을 종료하고 ENDDO 뒷부분의 처리를 재개하는 데 사용할 수 있으며 ITERATE 명령은 그룹의 나머지 명령을 건너뛰고 제어 변수를 증가시키고 종료 값 조건을 즉시 평가하는 데 사용할 수 있습니다.

ITERATE 명령 사용

ITERATE 명령은 활동 DOWHILE, DOUNTIL 또는 DOFOR 그룹의 나머지 명령을 건너뛰는 데 사용할 수 있습니다. ITERATE는 단순 DO 명령 그룹에서는 유효하지 않습니다.

레이블이 없는 ITERATE 명령은 가장 안쪽에 있는 활동 Do 그룹의 ENDDO로 건너뜁니다. 레이블을 지정하면 레이블과 연관된 DO의 ENDDO로 건너뜁니다.

다음은 ITERATE 명령을 사용하는 예입니다.

```
DO_1:
DO_2:DOWHILE &LGL
DO_3:  DOFOR &INT FROM(0) TO(99)
         .
         .
         .
         IF (&A *EQ 12) THEN (ITERATE DO_1)
         .
         . /* Not processed if &A equals 12        */
         .
         IF (&A *GT 12) ITERATE
         .
         .  /* Not processed if &A greater than 12 */
         .
       ENDDO
         .
         .
         .
       IF (&A *LT 0) (ITERATE DO_1)
         .
         . /* Not processed if &A less than zero   */
         .
 ENDDO
```

이 예에서 레이블 DO_1 및 DO_2는 DOWHILE 그룹과 연관되어 있습니다. 이러한 레이블은 DOWHILE 또는 DOFOR 그룹에 나타나는 ITERATE 명령에 지정할 수 있습니다. &A가 12이면 ITERATE DO_1 명령이 실행됩니다. DOWHILE 명령과 연관된 ENDDO에서 처리가 계속됩니다. &LGL 값을 평가하여 참이면 DOWHILE 뒤에 나오는 DOFOR를 처리합니다. &LGL이 거짓이면 두 번째 ENDDO 뒤에 나오는 CL 명령을 처리합니다.

&A가 12는 아니지만 12보다 큰 경우 DOFOR 그룹의 ENDDO를 처리합니다. &INT 값이 증가하고 종료 값 99와 비교됩니다. &INT가 99보다 작거나 같은 경우 DOFOR 명령 뒤의 첫 번째 명령을 처리합니다. &INT가 99보다 크면 첫 번째 ENDDO 뒤의 다음 명령을 처리합니다.

세 번째 IF 명령이 처리되고 &A가 0보다 작으면 두 번째 ENDDO를 처리합니다. &LGL 값을 평가하여 거짓이면 ENDDO 뒤의 명령으로 제어 권한이 전달되고 참이면 DOWHILE 뒤에 나오는 DOFOR 명령의 처리를 재개합니다.

LEAVE 명령 사용

LEAVE 명령은 활동 DOWHILE, DOUNTIL 또는 DOFOR 그룹을 종료하는 데 사용할 수 있습니다. 이 명령은 GOTO 명령을 사용하지 않고 활동 그룹을 나가는 구조화된 방법을 제공합니다. LEAVE는 단순 DO 명령 그룹에서는 유효하지 않습니다.

레이블이 없는 LEAVE 명령은 가장 안쪽에 있는 활동 Do 그룹을 나갑니다. 레이블을 지정하면 처리에서 하나 이상의 주변 그룹을 일시 중단할 수 있습니다.

다음은 LEAVE 명령의 사용 예입니다.

```
DO_1:
DO_2:DOWHILE &LGL
DO_3:  DOFOR &INT FROM(0) TO(99)
          .
          .
          .
          IF (&A *EQ 12) THEN(LEAVE DO_1)
          .
          . /* Not processed if &A equals 12        */
          .
          IF (&A *GT 12) LEAVE
          .
          .  /* Not processed if &A greater than 12 */
          .
 ENDDO
          .
          .
          .
        IF (&A *LT 0) (LEAVE DO_1)
          .
          . /* Not processed if &A less than zero   */
          .
                    ENDDO
```

이 예에서 레이블 DO_1 및 DO_2는 DOWHILE 그룹과 연관되어 있습니다. 이러한 레이블은 DOWHILE 또는 DOFOR 그룹에 나타나는 LEAVE 명령에 지정할 수 있습니다. &A가 12이면 LEAVE DO_1 명령이 실행되고 두 번째 ENDDO 뒤의 CL 명령을 처리합니다.

&A가 12는 아니지만 12보다 큰 경우 DOFOR 그룹이 종료되고 첫 번째 ENDDO 뒤의 다음 명령을 처리합니다.

세 번째 IF 명령이 처리되고 &A가 0보다 작으면 첫 번째 ENDDO 뒤의 다음 명령을 처리합니다.

SELECT 명령 및 SELECT 그룹 사용

SELECT 명령은 해당 조건이 참일 때 처리할 하나 이상의 조건 및 연관된 명령 그룹을 식별하는 데 사용됩니다. 특정 명령 그룹을 명시된 조건이 모두 거짓일 때 처리되도록 지정할 수도 있습니다. WHEN 또는 OTHERWISE 명령에 의해 식별되는 하나의 명령 그룹만이 그룹 내에서 처리됩니다.

SELECT 명령의 일반 구조는 다음과 같습니다.

```
SELECT
  WHEN (condition-1) THEN(command-1)
  .
  .
  .
  WHEN (condition-n) THEN(command-n)
  OTHERWISE command-x
ENDSELECT
```

SELECT 그룹은 하나 이상의 WHEN 명령을 지정해야 합니다. WHEN 명령에는 테스트된(참 또는 거짓) 표현식과 조건이 참인 경우에 수행해야 하는 조치를 지정하는 선택적 THEN 매개변수가 포함되어 있습니다.

COND 매개변수의 논리식은 하나의 논리 변수나 상수여야 하며 아니면 피연산자 둘 이상 간의 관계를 설명하여 그 표현식이 참 또는 거짓으로 평가되도록 해야 합니다. 논리식의 구성에 대한 자세한 내용은 49 페이지의 『*AND, *OR 및 *NOT 연산자 사용』 부분을 참조하십시오.

논리식의 조건이 참으로 평가되면 프로시듀어는 THEN 매개변수상의 CL 명령을 처리합니다. 이 프로시듀어는 DO, DOWHILE, DOUNTIL 또는 DOFOR 명령에 의해 지정된 단일 명령이나 명령 그룹일 수 있습니다. 조건이 참이 아니면 SELECT 그룹에서 다음 WHEN 명령에 지정된 조건이 평가됩니다. 뒤에 WHEN 명령이 없으면 OTHERWISE 명령(있는 경우)이 식별하는 명령이 처리됩니다. 다음 WHEN 및 OTHERWISE 명령이 없으면 연관된 ENDSELECT 명령 뒤의 다음 명령을 처리합니다.

```
SELECT
  WHEN (&LGL)
  WHEN (&INT *LT 0) THEN(CHGVAR &INT 0)
  WHEN (&INT *GT 0) (DOUNTIL (&INT *EQ 0))
                       CHGVAR &INT (&INT - 1)
 ENDDO
  OTHERWISE (CHGVAR &LGL '1')
ENDSELECT
```

&LGL의 초기 값이 참('1')이면 THEN 매개변수가 없기 때문에 ENDSELECT 뒤의 명령을 처리합니다.

&LGL의 초기 값이 거짓('0')이면 두 번째 WHEN의 COND가 평가됩니다. &INT가 0보다 작으면 CHGVAR이 처리되고 &INT 값을 0으로 설정합니다. 그런 다음 ENDSELECT 뒤의 명령을 처리합니다.

처음 두 조건이 충족되지 않으면 0보다 큰지 판별하기 위해 &INT 값을 검사합니다. 값이 0보다 크면 DOUNTIL 그룹이 입력되고 0에 도달할 때까지 &INT 값이 감소합니다. &INT가 0에 도달하면 DOUNTIL 그룹이 종료되고 ENDSELECT 뒤의 다음 명령을 처리합니다.

WHEN 명령의 조건이 모두 거짓으로 평가되면 OTHERWISE 명령의 CMD 매개변수에 지정된 CHGVAR이 처리됩니다. &LGL이 참으로 설정된 동안에는 &INT 값이 변경되지 않습니다. 그런 다음 ENDSELECT 뒤의 다음 명령을 처리합니다.

*AND, *OR 및 *NOT 연산자 사용

*AND와 *OR은 논리식에서 피연산자들 간의 관계를 지정하는 데 사용되는 예약된 논리 연산자 값입니다. 앰퍼샌드 기호(&)는 예약된 값 *AND를 대체할 수 있으며, 수직 막대(|)는 *OR을 대체할 수 있습니다. 예약된 값 앞뒤에는 공백이 있어야 합니다. 논리식의 피연산자는 논리 연산자에 의해 분리되는 관계식, 논리 변수 또는 상수로 이루어집니다. *AND 연산자는 (연산자 양쪽에 있는) 두 피연산자가 모두 참이어야 그 결과를 참으로 나타냅니다. *OR 연산자는 피연산자 중 하나만 참이어도 그 결과를 참으로 나타냅니다.

주: 앰퍼샌드(&)기호나 수직 막대를 사용할 경우 모든 코드 페이지에 있어서 기호들이 같은 코드점에 있는 것이 아니므로 문제가 발생할 수 있습니다. 이와 같은 문제는 기호 대신 *AND 및 *OR을 사용하여 방지할 수 있습니다.

표현식에서 피연산자에 수행할 조치 또는 피연산자 간의 관계를 나타내려면 논리 연산자가 아닌 연산자를 사용하십시오. 논리 연산자 이외에도 세 가지 종류의 연산자가 있습니다.

- 산술(+, -, *, /)
- 문자(*CAT, ||, *BCAT, |>, *TCAT, |<)

- 관계(*EQ, =, *GT, >, *LT, <, *GE, >=, *LE, <=, *NE, ¬=, *NG, ¬>, *NL, ¬<)

이들 연산자에 대한 정보는 **iSeries Information Center**의 *프로그래밍* 범주에서 *CL* 섹션을 참조하십시오.

다음은 논리식(logical expression)의 예입니다.

```
((&C *LT 1) *AND (&TIME *GT 1430))
(&C *LT 1 *AND &TIME *GT 1430)
((&C< 1) & (&TIME>1430))
((&C< 1) & (&TIME>1430))
```

각각의 경우에 논리식은 세 부분, 즉 두 개의 피연산자(operand)와 한 개의 연산자(operator)(*AND 또는 *OR 또는 그것들의 기호)로 되어 있습니다. 표현식을 논리적으로 특징지우는 것은 피연산자의 유형이 아니라 연산자의 유형(*AND나 *OR)입니다. 논리식에서 피연산자란 논리 변수 또는 다른 표현식(예: 관계식)입니다. (관계식은 >, < 또는 = 기호나 대응되는 예약된 값으로 특징지워집니다.) 다음 예를 보십시오.

```
((&C *LT 1) *AND (&TIME *GT 1430))
```

위의 경우 논리식 전체가 괄호로 묶여 있고, 두 피연산자는 관계식이면서 또한 각각 분리되어 괄호로 묶여 있습니다. 논리식의 두 번째 예와 같이 각 피연산자를 분리시켜 괄호로 묶을 필요는 없으나 괄호로 묶으면 더욱 명확해집니다. *AND와 *OR은 우선순위가 다르기 때문에 괄호가 필요없습니다. *AND가 항상 *OR보다 우선합니다. 우선순위가 동일한 연산자는 괄호를 사용하여 연산의 실행 순서를 제어할 수 있습니다.

간단한 관계식은 하나의 조건으로 이루어진 명령으로 작성될 수 있습니다.

```
IF (&A=&B) THEN(DO)
              .
              .
              .
                   ENDDO
```

이 관계식의 피연산자는 상수일 수도 있습니다.

조건을 둘 이상 지정하려면 관계식으로 된 논리식을 피연산자로 사용할 수 있습니다.

```
IF ((&A=&B) *AND (&C=&D)) THEN(DO)
                                .
                                .
                                .
                   ENDDO
```

41 페이지의 『내장 IF 명령 사용』에서 예로 인용된 종속 IF 명령 시리즈는 다음과 같이 코드화할 수 있습니다.

```
   PGM
DCL &RESP *DEC 1
DCL &A *DEC 1
DCL &B *CHAR 2
```

```
IF ((&RESP=1) *AND (&A=5) *AND (&B=NO)) THEN(DO)
                                          .
                                          .
                                          .
 ENDDO
CHGVAR &A VALUE(8)
CALLPRC PROC(DAILY)
    ENDPGM
```

여기에서 논리 연산자가 관계식 사이에서 다시 사용됩니다.

논리식에서는 피연산자로 다른 논리식을 사용할 수도 있기 때문에 논리가 매우 복잡해질 수 있습니다.

```
IF (((&A=&B) *OR (&A=&C)) *AND ((&C=1) *OR (&D='0'))) THEN(DO)
```

이 경우 &D는 논리 변수로 정의됩니다.

관계식이나 논리식의 평가 결과는 '1' 또는 '0'(참 또는 거짓)입니다. 종속 명령은 모든 표현식의 평가가 참('1')일 때만 처리됩니다. 아래의 명령은 이러한 의미로 해석됩니다.

```
IF   ((&A = &B) *AND (&C = &D)) THEN(DO)

     ((true'1') *AND (not true'0'))
          (not true '0')
```

표현식이 최종적으로 참이 아닌 것('0')으로 평가되었기 때문에 DO가 처리되지 않았습니다. 이런 평가가 나오게 된 과정을 알려면 이 섹션의 뒷부분에 나오는 매트릭스를 참조하십시오.

아래의 예에서도 논리 변수를 사용한 논리식을 계산하기 위해 같은 프로세스가 사용됩니다.

```
   PGM
DCL &A *LGL
DCL &B *LGL
IF (&A *OR &B) THEN(CALL PGM(PGMA))
.
.
.
    ENDPGM
```

여기에서 조건식은 &A 또는 &B의 값이 '1'(참)인지를 보기 위해 평가됩니다. 둘 중 하나가 참이면 전체 표현식은 참이 되며 PGMA가 호출됩니다.

이러한 모든 논리식의 예에서 도달한 마지막 평가는 *OR 또는 *AND 연산자 아래의 두 값(여기에서는 &A와 &B)을 비교하는 표준 매트릭스에 기초하고 있습니다.

논리 변수나 상수를 포함하여 *OR을 사용할 때는 다음의 매트릭스를 사용하십시오.

&A가

'0' '0' '1' '1'이고

&B가

'0' '1' '0' '1'이면

OR 표현식은

'0' '1' '1' '1'입니다

간단히 말해, 논리 변수나 상수를 포함한 복수 OR 연산자의 경우 모든 값이 거짓일 때만 표현식은 거짓('0')이 됩니다. 즉, 어떤 값이 하나라도 참이면 표현식은 참('1')이 됩니다.

```
   PGM
DCL &A *LGL VALUE('0')
DCL &B *LGL VALUE('1')
DCL &C *LGL VALUE('1')
IF (&A *OR &B *OR &C) THEN(CALL PGMA)
.
.
.
    ENDPGM
```

여기에서는 값들이 모두 거짓이 아니므로 표현식은 참이 되고 PGMA가 호출됩니다.

논리 변수나 상수를 포함한 논리식 *AND를 사용할 때는 다음의 매트릭스를 사용하십시오.

&A가

'0' '0' '1' '1'이고

&B가

'0' '1' '0' '1'이면

AND 표현식은

'0' '0' '0' '1'입니다

논리 변수나 상수를 포함한 복수 AND 연산자의 경우 어떤 값이 하나만 거짓이어도 표현식은 거짓('0')이 되며 모든 값이 참일 때만 표현식이 참이 됩니다.

```
   PGM
DCL &A *LGL VALUE('0')
DCL &B *LGL VALUE('1')
DCL &C *LGL VALUE('1')
IF (&A *AND &B *AND &C) THEN(CALL PGMA)
.
.
.
    ENDPGM
```

여기에서는 값들이 모두 참이 아니므로 표현식은 거짓이 되고 따라서 PGMA가 호출되지 않습니다.

앞의 예에서와 같이 이러한 논리 연산자는 피연산자가 논리값일 경우 표현식 *안*에서만 사용될 수 있습니다. 논리 변수 이외의 변수에 OR이나 AND를 사용해서는 *안됩니다*. 다음 예를 보십시오.

```
   PGM
DCL &A *CHAR 3
DCL &B *CHAR 3
DCL &C *CHAR 3
```

틀린 예: `IF (&A *OR &B *OR &C = YES) THEN...`

위의 예를 올바르게 코딩하면 다음과 같습니다.

```
IF ((&A=YES) *OR (&B=YES) *OR (&C=YES)) THEN...
```

이 경우 관계식 사이에 OR이 생깁니다.

논리 연산자 *NOT(또는 ¬)은 논리 변수나 상수를 부정하는 데 사용됩니다. 모든 *NOT 연산자는 *AND나 *OR 연산자보다 먼저 평가되어야 합니다. *NOT 연산자 다음의 모든 값은 피연산자 간의 논리 관계보다 먼저 평가되어야 합니다.

```
   PGM
DCL &A *LGL '1'
DCL &B *LGL '0'
IF (&A *AND *NOT &B) THEN(CALL PGMA)
```

이 예에서는 값이 모두 참이므로 표현식은 참이 되고 따라서 PGMA가 호출됩니다.

```
   PGM
DCL &A *LGL
DCL &B *CHAR 3 VALUE('ABC')
DCL &C *CHAR 3 VALUE('XYZ')
CHGVAR &A VALUE(&B *EQ &C)
IF (&A) THEN(CALLPRC PROCA)
```

이 예에서는 값이 거짓이므로 PROCA가 호출되지 않습니다.

논리식 및 관계식에 관한 자세한 정보는 **iSeries Information Center**의 *프로그래밍* 범주에서 *CL* 섹션을 참조하십시오.

%BINARY 내장 기능 사용

2진 내장 기능(%BINARY 또는 %BIN)은 지정된 CL 문자 변수의 내용을 부호화 2진 정수로 해석합니다. 지정 위치에서 시작 위치가 시작하며 그 길이는 2자 또는 4자까지입니다.

2진 내장 기능의 구문은 다음과 같습니다.

```
%BINARY(character-variable-name starting-position length)
```

또는

```
%BIN(character-variable-name starting-position length)
```

시작 위치와 길이는 생략할 수 있습니다. 그러나 시작 위치와 길이가 *지정되지 않으면*, 1의 시작 위치에 지정된 문자 변수의 길이가 사용됩니다. 그와 같은 경우에는 문자 변수의 길이를 반드시 2나 4로 선언해야 합니다.

시작 위치가 *지정된* 경우 상수의 길이도 2 또는 4로 지정해야 하며, 시작 위치는 1 이상의 양수여야 합니다. 시작 위치와 길이의 합계가 문자 변수보다 크면 오류가 발생합니다. (CL 10진 변수 또는 정수 변수를 시작 위치에 사용할 수도 있습니다.)

IF 및 CHGVAR 명령 모두와 2진 내장 기능을 함께 사용할 수 있습니다. 내장 기능은 단독으로 사용되거나 산술식 또는 논리식의 일부로 사용할 수 있습니다. 또한 EXPR(*YES)과 함께 숫자(*DEC, *INT2, *INT4, *UINT2 또는 *UINT4의 유형)로 정의된 명령 매개변수에도 2진 내장 기능을 사용할 수 있습니다.

2진 내장 기능이 IF 명령상의 조건(COND) 매개변수와 함께 또는 CHGVAR(변수 변경) 명령상의 VALUE 매개변수와 함께 사용되면 문자 변수의 내용이 2진에서 10진으로 변환된 것으로 해석됩니다.

2진 내장 기능이 CHGVAR 명령에서 VAR 매개변수와 함께 사용되면 VALUE 매개변수 안의 10진값이 부호화 2바이트나 4바이트의 2진 정수로 변환되어, 지정 시작 위치에서 문자 변수 안에 결과가 저장됩니다. 소수 부분은 잘립니다.

시스템은 호출 프로시듀어가 부호화 2진 정수로 호출된 프로시듀어를 리턴하도록 나타내기 위해 CALLPRC 명령의 RTNVAL 매개변수에 2진 내장 기능을 사용합니다.

2바이트 문자 변수는 -32,768에서 32,767까지 부호화 2진 정수값을 보유할 수 있습니다. 4바이트 문자 변수는 -2,147,483,648에서 2,147,483,647까지 부호화 2진 정수값을 보유할 수 있습니다.

다음은 2진 내장 기능의 예입니다.

- ```
  DCL    VAR(&B2)  TYPE(*CHAR)  LEN(2)    VALUE(X'001C')
  DCL    VAR(&N)   TYPE(*DEC)   LEN(3 0)
  CHGVAR   &N   %BINARY(&B2)
  ```

  변수 &B2의 내용이 부호화 2바이트 2진 정수로 취급되어 등가의 10진수 28로 변환됩니다. 그런 다음 10진 변수 &N에 할당됩니다.

- ```
  DCL    VAR(&N)   TYPE(*DEC)   LEN(5 0)  VALUE(107)
  DCL    VAR(&B4)  TYPE(*CHAR)  LEN(4)
  CHGVAR   %BIN(&B4)    &N
  ```

  10진 변수 &N의 값이 부호화 4바이트 2진수로 변환되어 문자 변수 &B4에 놓이고, 변수 &B4는 X'0000006B'라는 값을 갖게 됩니다.

- ```
  DCL    VAR(&P)  TYPE(*CHAR)  LEN(100)
  DCL    VAR(&L)  TYPE(*DEC)   LEN(5 0)
  CHGVAR   &L  VALUE(%BIN(&P 1 2) * 5)
  ```
  ```

변수 &P의 처음 2개의 문자는 부호화 2진 정수로 취급되어 등가의 10진수로 변환되고 5가 곱해집니다. 산출된 결과는 십진 변수 &L에 할당됩니다.

- ```
  DCL    VAR(&X)   TYPE(*CHAR)  LEN(50)
  CHGVAR   %BINARY(&X 15 2)  VALUE(122.56)
  ```

  숫자 122.56은 정수 122로 절단된 후 2바이트 부호화 2진 정수로 변환되어 문자 변수 &X의 15와 16 위치에 놓여집니다. 변수 &X의 15와 16 위치에는 등가의 16진수 X'007A'가 들어가게 됩니다.

- ```
  DCL    VAR(&B4)  TYPE(*CHAR)  LEN(4)
  CHGVAR   %BIN(&B4)  VALUE(-57)
  ```

  값 -57은 4바이트 부호화 2진 정수로 변환되어 문자 변수 &B4에 할당됩니다. 변수 &B4에는 값 X'FFFFFFC7'이 들어가게 됩니다.

- ```
  DCL    VAR(&B2)  TYPE(*CHAR)  LEN(2)   VALUE(X'FF1B')
  DCL    VAR(&C5)  TYPE(*CHAR)  LEN(5)
  CHGVAR   &C5  %BINARY(&B2)
  ```

  변수 &B2의 내용이 부호화 2바이트 2진 정수로 취급되어 등가의 10진수 -229로 변환됩니다. 이 숫자는 문자 형식으로 변환되어 변수 문자 &C5에 저장됩니다. 그러면 문자 변수 &C5에 '-0229' 값이 포함됩니다.

- ```
  DCL    VAR(&C5)  TYPE(*CHAR)  LEN(5)   VALUE(' 1253')
  DCL    VAR(&B2)  TYPE(*CHAR)  LEN(2)
  CHGVAR   %BINARY(&B2)  VALUE(&C5)
  ```

  문자 변수 &C5 안의 문자 수(character number) 1253이 10진수로 변환됩니다. 그런 다음, 십진수 1253은 2바이트 부호화 2진 정수로 변환되어 변수 &B2에 저장됩니다. 변수 &B2에는 값 X'04E5'가 들어가게 됩니다.

- ```
  DCL    VAR(&S)  TYPE(*CHAR)  LEN(100)
  IF     (%BIN(&S 1 2) *GT 10)
         THEN( SNDPGMMSG MSG('Too many in list.') )
  ```

  문자 변수 &S의 처음 2바이트는 숫자 10에 비교될 때 부호화 2진 정수로 취급됩니다. 2진수에 10보다 더 큰 값이 있으면 SNDPGMMSG(프로그램 메세지 송신) 명령이 수행됩니다.

- ```
  DCL    VAR(&RTNV) TYPE(*CHAR) LEN(4)
  CALLPRC PRC(PROCA) RTNVAL(%BIN(&RTNV 1 4))
  ```

  프로시듀어 PROCA가 변수 &RTNV에 저장되어 있는 4바이트 정수를 리턴합니다.

%SUBSTRING 내장 기능 사용

서브스트링 내장 기능(%SUBSTRING 또는 %SST)은 기존 문자 스트링의 서브세트인 문자 스트링을 작성하며 CL 프로시듀어 안에서만 사용될 수 있습니다. CHGVAR 명령에서 %SST 기능은 변경될 변수(VAR 매개변수) 또는 변경될 값(VALUE 매개변수)을 대신하여 지정될 수 있습니다. IF 명령에서는 %SST 기능을 표현식에 지정할 수 있습니다.

서브스트링 내장 기능의 형식은 다음과 같습니다.

```
%SUBSTRING(character-variable-name starting-position length)
```

또는

```
%SST(character-variable-name starting-position length)
```

서브스트링 기능이 로컬 자료 영역의 내용에 대해 수행됨을 나타내기 위해 문자 변수명을 대신하여 *LDA를 코드화할 수 있습니다.

서브스트링 기능은 지정 CL 문자 변수나 로컬 자료 영역의 내용으로부터 서브스트링을 작성합니다. 서브스트링은 지정 시작 위치(변수명일 수도 있음)에서 시작하여 지정 길이(변수명일 수도 있음)만큼 계속됩니다. 시작 위치나 길이는 0이나 음수가 될 수 없습니다. 서브스트링의 시작 위치와 길이의 합계가 전체 변수의 길이나 로컬 자료 영역의 길이보다 크면, 오류가 발생합니다. 로컬 자료 영역의 길이는 1,024입니다.

다음은 서브스트링 내장 기능의 예입니다.

- 문자 변수 &NAME의 처음 두 자리가 IN이면 프로그램 INV210이 호출됩니다. &NAME의 전체 값이 INV210으로 전달되며 &ERRCODE의 값은 변경되지 않습니다. 그렇지 않으면 &ERRCODE 값이 99로 설정됩니다.

  ```
  DCL &NAME *CHAR VALUE(INVOICE)
  DCL &ERRCODE *DEC (2 0)
  IF (%SST(&NAME 1 2) *EQ 'IN') +
  THEN(CALL INV210 &NAME)
  ELSE CHGVAR &ERRCODE 99
  ```

- &A의 처음 두 자리가 &B의 처음 두 자리와 일치하면 프로그램 CUS210이 호출됩니다.

  ```
  DCL &A *CHAR VALUE(ABC)
  DCL &B *CHAR VALUE(DEF)
  IF (%SST(&A 1 2) *EQ %SUBSTRING(&B 1 2)) +
  CALL CUS210
  ```

- 위치와 길이도 변수가 될 수 있습니다. 다음 예에서는 길이 &Z의 위치 &Y에서 시작하는 &X의 값을 123으로 변경합니다.

  ```
  CHGVAR %SST(&X &Y &Z) '123'
  ```

- 아래의 CHGVAR 명령이 수행되기 전에 &A가 ABCDEFG이면 &A는 다음과 같습니다.

  ```
  CHGVAR %SST(&A 2 3) '123'
  ```

  이 명령이 수행된 후 A123EFG가 됩니다.

- 이 예에서는 서브스트링의 길이가 5이므로 이와 비교될 피연산자인 YES의 길이보다 깁니다. 따라서 피연산자는 공백으로 채워져 YESNO와 YESbb(b는 공백을 의미)가 비교됩니다. 조건은 거짓이 됩니다.

```
DCL VAR(&NAME) TYPE(*CHAR) LEN(5) VALUE(YESNO)
.
.
.
IF (%SST (&NAME 1 5) *EQ YES) +
  THEN(CALL PROGA)
```

서브스트링의 길이가 피연산자보다 짧으면 비교 시 서브스트링이 공백으로 채워집니다. 예를 들어, 다음과 같은 경우가 있습니다.

```
DCL VAR(&NAME) TYPE(*CHAR) LEN(5) VALUE(YESNO)
.
.
.
IF (%SST(&NAME 1 3 ) *EQ YESNO) THEN(CALL PROG)
```

YESbb(bb는 2개의 공백을 의미)가 YESNO와 같지 않으므로 이 조건은 거짓이 됩니다.

- 변수 &A의 값이 로컬 자료 영역의 1에서 10의 위치에 놓입니다.

  ```
  CHGVAR %SST(*LDA 1 10) &A
  ```

- 로컬 자료 영역의 1에서 3까지의 위치와 상수 'XYZ'를 연결한 값이 변수 &A와 같으면, PROCA가 호출됩니다. 예를 들면, 로컬 자료 영역의 1에서 3까지의 위치에 'ABC'가 들어 있고, 변수 &A의 값이 ABCXYZ이면, 테스트 결과는 참이 되며 PROCA가 호출됩니다.

  ```
  IF (((%SST*LDA 1 3) *CAT 'XYZ') *EQ &A) THEN(CALLPRC PROCA)
  ```

- 이 프로시듀어는 문자 변수 &NUMBER를 검색하여 선행 0을 공백으로 변경합니다. 이것은 메세지에 표시되기 전에 필드를 간단하게 편집하기 위해 사용할 수 있습니다.

  ```
            DCL &NUMBER *CHAR LEN(5)
      DCL &X *DEC LEN(3 0) VALUE(1)
      .
      .
  LOOP:IF (%SST(&NUMBER &X 1) *EQ '0') DO
      CHGVAR (%SST(&NUMBER &X 1)) ' '  /* Blank out */
      CHGVAR &X (&X + 1)  /* Increment */
      IF (&X *NE 4) GOTO LOOP
                  ENDDO
  ```

다음 프로시듀어는 서브스트링 내장 기능을 사용하여 50자 필드 &INPUT의 첫 문장을 찾고 남은 텍스트를 필드 &REMAINDER에 넣습니다. 이때 문장에는 최소한 2자가 들어 있으며 마침표가 내장되지 않은 것으로 가정한 것입니다.

```
          PGM (&INPUT &REMAINDER)   /*  SEARCH  */
    DCL &INPUT *CHAR LEN(50)
    DCL &REMAINDER *CHAR LEN(50)
    DCL &X *INT  /* INDEX            */
    DCL &L *INT  /* REMAINING LENGTH */

    DOFORL:
    DOFOR &X 3 50
      IF (%SST(&INPUT &X 1) *EQ '.') THEN(DO)
```

```
            CHGVAR &L (50-&X)
          CHGVAR &X (&X+1)
            CHGVAR &REMAINDER %SST(&INPUT &X &L)
            LEAVE
 ENDDO
 ENDDO
ENDPGM
```

프로시듀어는 맨 처음에 세 번째 위치가 마침표인지를 검사하는 것부터 시작됩니다. 서브스트링 기능이 세 번째 위치부터 길이 1만큼, 즉 세 번째 위치(길이는 0이 될 수 없음)에서만 &INPUT을 검사한다는 점에 유의해야 합니다. 세 번째 위치가 마침표이면 &INPUT의 나머지 길이가 계산됩니다. &X의 값은 나머지 부분의 맨 처음으로 가고, &INPUT의 나머지 부분은 &REMAINDER로 이동합니다.

위치 3이 마침표가 아니면, 프로시듀어는 마침표가 49번째 위치에 있는지 검사합니다. 49번째 위치에 있으면 50번째 위치가 마침표인 것으로 간주하여 리턴합니다. 49번째 위치에 없으면 프로시듀어는 &X를 네 번째 위치로 올려 프로세스를 반복합니다.

%SWITCH 내장 기능 사용

스위치 내장 기능(%SWITCH)이 8개의 스위치 중 하나 이상을 작업에 이미 설정된 8개의 스위치 설정값과 비교하여 논리값 '0' 또는 '1'을 리턴합니다. 작업에 대한 스위치의 초기값은 우선 CRTJOBD(작업 설명 작성) 명령에 의해 결정됩니다. 디폴트 값은 00000000입니다. 필요한 경우 SBMJOB, CHGJOB 또는 JOB 명령에서 SWS 매개변수를 사용하여 이것을 변경할 수 있습니다. 이것의 디폴트는 작업 설명 설정입니다. 다른 고급 언어에서도 작업 스위치를 설정할 수 있습니다.

%SWITCH 값을 작업값과 비교할 때 모든 스위치가 같으면 논리값 '1'(참)이 리턴됩니다. 테스트할 스위치에 지시된 값이 없으면 결과는 '0'(거짓)이 됩니다.

%SWITCH 내장 기능의 구문은 다음과 같습니다.

```
%SWITCH(8-character-mask)
```

8자 마스크를 사용하여 테스트할 작업 스위치 및 테스트할 각 스위치의 값을 지정할 수 있습니다. 마스크 안의 각 위치는 한 작업의 8개 작업 스위치 중 하나와 대응됩니다. 위치 1은 작업 스위치 1과 대응되고 위치 2는 스위치 2와 대응되는 등 이런 식으로 계속됩니다. 마스크의 각 위치는 0, 1, X의 세 가지 값 중 하나로 지정될 수 있습니다.

0 대응되는 작업 스위치가 0(OFF)인지 테스트함.

1 대응되는 작업 스위치가 1(ON)인지 테스트함.

X 대응되는 작업 스위치를 테스트하지 않음. 스위치 값이 %SWITCH의 결과에 영향을 미치지 않음.

%SWITCH(0X111XX0)가 지정되면 작업 스위치 1과 8이 0인지 테스트하고, 스위치 3, 4, 5가 1인지 테스트하며, 스위치 2, 6, 7은 테스트하지 않습니다. 각 작업 스위치 안에 마스크에 표시된 값(1 또는 0만)이 들어 있으면 %SWITCH의 결과는 참 '1'입니다.

프로시듀어의 흐름을 제어하기 위해서는 CL 프로시듀어에서 스위치를 테스트할 수 있습니다. 이 기능은 CL 프로시듀어에서 IF 및 CHGVAR 명령과 함께 사용됩니다. CHGJOB(작업 변경) 명령을 사용하면 CL 프로시듀어에서 스위치를 변경할 수 있습니다. CL 프로시듀어의 경우 이 변경 내용이 즉시 적용됩니다.

IF 명령을 사용한 %SWITCH

IF 명령에서 %SWITCH는 테스트될 논리식으로서 COND 매개변수에 지정될 수 있습니다. 아래의 예에서 0X111XX0은 미리 결정된 작업 스위치 설정과 비교됩니다.

```
IF  COND(%SWITCH(0X111XX0))  THEN(GOTO C)
```

작업 스위치 1, 3, 4, 5, 8에 각각 0, 1, 1, 1, 0이 들어 있으면, 결과는 참이 되고 프로시듀어는 C라는 레이블이 있는 명령으로 분기됩니다. 테스트된 스위치 중 하나 이상이 마스크에서 지시한 값을 갖고 있지 않으면 결과는 거짓이 되고 분기되지 않습니다.

아래의 예에서 스위치는 두 개의 프로시듀어에서 조건부 처리를 제어합니다.

```
SBMJOB  JOB(APP502) JOBD(PAYROLL) CMD(CALL APP502)
    SWS(11000000)

PGM  /* CONTROL */
IF (%SWITCH(11XXXXXX)) CALLPRC PROCA
IF (%SWITCH(10XXXXXX)) CALLPRC PROCB
IF (%SWITCH(01XXXXXX)) CALLPRC PROCC
IF (%SWITCH(00XXXXXX)) CALLPRC PROCD
ENDPGM

PGM  /* PROCA */
CALLPRC TRANS
IF (%SWITCH(1XXXXXXX)) CALLPRC CUS520
ELSE CALLPRC CUS521
    ENDPGM
```

CHGVAR 명령을 사용한 %SWITCH

CHGVAR 명령에 %SWITCH를 지정하여 논리 변수의 값을 변경할 수 있습니다. 논리 변수의 값은 사용자의 %SWITCH 설정을 작업 스위치 설정과 비교한 결과에 따라 결정됩니다. 비교 결과가 참이면 논리 변수는 '1'로 설정됩니다. 결과가 거짓이면 변수는 '0'으로 설정됩니다. 예를 들어, 작업 스위치를 10000001로 설정하여 프로시듀어를 처리하면 다음과 같이 하십시오.

```
   PGM
DCL &A *LGL
CHGVAR VAR(&A) VALUE(%SWITCH(10000001))
.
.
.
    ENDPGM
```

변수 &A는 값 '1'을 가집니다.

MONMSG(메세지 모니터) 명령 사용

이탈 메세지는 CL 프로시듀어의 명령과 CL 프로시듀어가 호출하는 프로그램 및 프로시듀어에 의해 CL 프로시듀어로 송신됩니다. 이탈 메세지는 오류가 검출되어 요구된 기능이 수행되지 않았음을 프로시듀어에 알려주기 위해 송신되는 것입니다. CL 프로시듀어는 이탈 메세지가 도착했는지 모니터할 수 있으며, 사용자는 명령을 통해 메세지를 처리하는 방법을 지정할 수 있습니다. 예를 들어, 이미 삭제된 자료 영역을 CL 프로시듀어가 이동하려고 시도하면 '오브젝트 없음'이라는 이탈 메세지가 MOVOBJ(오브젝트 이동) 명령에 의해 프로시듀어로 송신됩니다.

바로 앞의 명령이 처리되는 동안 특정 오류가 발생할 경우 MONMSG(메세지 모니터) 명령을 사용하여 미리 지정해 둔 조치를 프로시듀어에 지시할 수 있습니다. MONMSG 명령은 MONMSG 명령이 사용된 프로시듀어의 호출 스택으로 송신된 이탈, 통지 또는 상태 메세지를 모니터하는 데 사용됩니다. MONMSG 명령은 다음과 같은 매개변수를 가집니다.

```
MONMSG   MSGID(message-identifier) CMPDTA(comparison-data) +
         EXEC(CL-command)
```

특정 오류에 대해 송신되는 각 메세지는 고유한 ID를 가지고 있습니다. MSGID 매개변수에는 50개까지의 메세지 ID를 입력할 수 있습니다. (메세지와 ID에 대해서는 온라인 도움말을 참조하십시오.) CMPDTA 매개변수를 사용하면 메세지의 MSGDTA 부분에서 특정 문자 스트링을 검사할 수 있기 때문에 오류 메세지를 더 광범위하게 스펙할 수 있습니다. EXEC 매개변수에 CL 명령(프로그램 호출(CALL), 수행(DO) 또는 찾아가기(GOTO)와 같은)을 지정할 수 있으며, 이 명령들은 프로시듀어에 오류 복구를 수행하도록 지시합니다.

아래의 예에서 MONMSG 명령은 RCVF(파일 수신) 명령 뒤에 오고 따라서 RCVF 명령이 송신한 메세지만 모니터합니다.

```
READLOOP:  RCVF                          /* Read a file record  */
           MONMSG  MSGID(CPF0864)  EXEC(GOTO CMDLBL(EOF))
           /*  Process the file record  */
           GOTO  CMDLBL(READLOOP)       /*  Get another record  */
EOF:       /*  End of file processing  */
```

이탈 메세지인 CPF0864는 읽을 파일에 레코드가 더 이상 없을 때 프로시듀어의 호출 대기행렬로 송신됩니다. 이는 예에서 이 조건에 대해 MONMSG 모니터인 MSGID(CPF0864)를 지정했기 때문입니다. 메세지를 수신할 때 GOTO CMDLBL(EOF) 명령이 실행됩니다.

또한 MONMSG 명령을 사용하여 CL 프로시듀어 안의 임의의 명령이 송신한 메세지를 모니터할 수 있습니다. 다음의 예에는 두 개의 MONMSG 명령이 들어 있습니다. 첫 번째 MONMSG 명령은 메세지 CPF0001과 CPF1999를 모니터하며 프로시듀어에

서 이후에 실행되는 명령이 이들 메세지를 송신할 수 있습니다. 프로시듀어에서 실행 중인 임의의 명령으로부터 메세지가 수신되는 경우 레이블 EXIT2로 식별되는 명령으로 제어가 분기됩니다.

두 번째 MONMSG 명령은 메세지 CPF2105와 MCH1211을 모니터합니다. EXEC 매개변수에 대해서 코드화된 명령이 없으므로 이 메세지는 무시됩니다.

```
   PGM
DCL
MONMSG MSGID(CPF0001 CPF1999) EXEC(GOTO EXIT2)
MONMSG MSGID(CPF2105 MCH1211)
.
.
.
ENDPGM
```

메세지 CPF0001은 메세지에서 식별된 명령에 오류가 있음을 말해줍니다. CHGPGMVAR(프로그램 변수 변경) 같은 대부분의 디버깅 명령에 의해 송신될 수 있는 메세지 CPF1999는 명령에서 오류가 발생했다는 것을 알려주지만 메세지 안에서 명령을 식별하지는 않습니다.

EXEC 매개변수가 지정되어 있는 MONMSG 명령(CPF0001 또는 CPF1999)에 의해 모니터되는 모든 오류 상태는 EXIT2에서 동일한 방법으로 처리되며 오류 다음에 나오는 명령문으로 리턴할 수 없습니다. 각 명령 다음에는 특정 조건을 모니터하여 해당되는 오류 정정 프로시듀어로 분기함으로써 이러한 상태를 방지할 수 있습니다.

EXEC 매개변수가 지정되지 않은 MONMSG 명령(CPF2105 또는 MCH1211)에 의해 모니터되는 모든 오류 상태는 무시되며 프로시듀어 처리가 다음 명령으로 계속 진행됩니다.

IF 명령에 표현식을 평가할 때 오류가 발생하면 조건은 거짓(false)으로 간주됩니다. 아래의 예에서 MCH1211(0으로 나눔)이 IF 명령에서 발생할 수 있습니다. 이 조건은 거짓으로 간주되며 PROCA가 호출될 것입니다.

```
IF(&A / &B *EQ 5) THEN(DLTF ABC)
ELSE CALLPRC PROCA
```

CL 프로시듀어의 시작 부분에 MONMSG 명령을 코드화하면 사용자가 지정한 메세지는 어떤 명령이 이러한 메세지를 발생시키든간에 프로그램 전체를 통하여 모니터됩니다. EXEC 매개변수가 사용되면 GOTO 명령만 지정할 수 있습니다.

프로시듀어 레벨 또는 명령 레벨 MONMSG 명령에는 동일한 메세지 ID를 지정할 수 있습니다. 명령 레벨의 MONMSG 명령은 프로시듀어 레벨의 MONMSG 명령보다 우선합니다. 아래의 예에서 메세지 CPF0001이 CMDB에서 수신되면 CMDC가 수행됩니다. 메세지 CPF0001이 프로시듀어 안의 다른 명령에서 수신되면 프로시듀어는 EXIT2로 분기됩니다. 메세지 CPF1999가 CMDB를 포함한 다른 어떤 명령에서 수신되면 프로시듀어는 EXIT2로 분기됩니다.

```
   PGM
MONMSG MSGID(CPF0001 CPF1999) EXEC(GOTO EXIT2)
CMDA
CMDB
MONMSG MSGID(CPF0001) EXEC(CMDC)
CMDD
EXIT2:  ENDPGM
```

많은 이탈 메세지가 하나의 프로시듀어로 송신될 수 있으므로 모니터하여 처리하고자 하는 메세지를 결정해야 합니다. 이러한 대부분의 메세지는 프로시듀어에 오류가 있을 경우에만 프로시듀어에 송신됩니다. 다른 메세지는 프로시듀어의 외부 조건으로 인하여 송신됩니다. 일반적으로, CL 프로시듀어는 기본 기능에 관련된 메세지와 적합하게 처리할 수 있는 그와 같은 메세지를 모니터해야 합니다. 다른 모든 메세지의 경우 OS/400은 오류가 발생했다는 가정하에 기본적인 적절한 조치를 수행합니다.

CL 프로시듀어의 메세지 처리에 관해 자세히 알려면 제 7 장 및 제 8 장을 참조하십시오.

변수로 사용할 수 있는 값

시스템 값 검색

시스템 값은 시스템의 특정 부분의 작업을 위한 제어 정보를 포함하고 있습니다. IBM® 에서는 여러 가지 유형의 시스템 값을 제공합니다. 예를 들어, QDATE와 QTIME는 날짜와 시간 시스템 값으로서, OS/400이 시작될 때 이를 설정하게 됩니다.

시스템 값을 프로시듀어로 가져와서 RTVSYSVAL(시스템 값 검색) 명령을 사용하여 변수로 조작할 수 있습니다.

```
RTVSYSVAL  SYSVAL(system-value-name)  RTNVAR(CL-variable-name)
```

RTNVAR 매개변수는 시스템 값을 수신하게 될 CL 프로시듀어 안의 변수명을 지정합니다.

변수 유형은 시스템 값의 유형과 일치해야 합니다. 문자 시스템 값과 논리 시스템 값인 경우에 CL 변수의 길이는 값의 길이와 같아야 합니다. 10진값인 경우에는 변수의 길이가 시스템 값의 길이보다 *크거나* 같아야 합니다. 시스템 값 속성은 iSeries Information Center에서 정보의 **시스템 관리** 범주에 정의되어 있습니다.

QTIME 시스템 값

다음 예에서는 QTIME를 수신하여 변수로 넣은 다음 이를 다른 변수와 비교합니다.

```
PGM
   DCL  VAR(&PWRDNTME)  TYPE(*CHAR)  LEN(6)  VALUE('162500')
   DCL  VAR(&TIME)  TYPE(*CHAR)  LEN(6)
   RTVSYSVAL  SYSVAL(QTIME)  RTNVAR(&TIME)
   IF  (&TIME *GT &PWRDNTME)  THEN(DO)
   SNDBRKMSG('Powering down in 5 minutes.  Please sign off.')
```

```
      PWRDWNSYS  OPTION(*CNTRLD)  DELAY(300)  RESTART(*NO)  +
                 IPLSRC(*PANEL)

 ENDDO
ENDPGM
```

시스템 값 리스트와 이를 변경 및 표시할 수 있는 방법에 대해서는 iSeries Information Center에 있는 정보의 **시스템 관리** 범주를 참조하십시오.

QDATE 시스템 값

많은 어플리케이션의 경우 시스템 값 QDATE를 검색하여 변수에 넣음으로써 프로시듀어에 현재 날짜를 사용할 수 있습니다. 또한 프로시듀어에서 사용하기 위해 그 날짜의 형식을 변경할 수도 있습니다. CL 프로시듀어에서 날짜 형식을 변환하려는 경우 CVTDAT(날짜 변환) 명령을 사용하십시오.

시스템 날짜 형식은 시스템 값 QDATFMT입니다. 제공되는 QDATFMT의 값은 국가 또는 지역마다 다릅니다. 예를 들어, 062488은 1988년 6월 24일에 대한 MDY(월일년) 형식입니다. 이 형식을 YMD, DMY 또는 JUL(율리우스력) 형식으로 변경할 수도 있습니다. 율리우스력 형식에서 QDAY 값은 001에서 366까지의 3자로 된 값입니다. 이것은 두 날짜 간의 일 수를 판별하는 경우에 사용됩니다. 또한 CVTDAT 명령을 사용하여 날짜 분리자를 삭제하거나 날짜 분리자로 사용된 문자를 변경할 수 있습니다.

CVTDAT 명령에 대한 형식은 다음과 같습니다.

```
CVTDAT    DATE(date-to-be-converted) TOVAR(CL-variable) +
          FROMFMT(old-format) TOFMT(new-format) +
          TOSEP(new-separators)
```

DATE 매개변수를 사용하면 변환될 상수나 변수를 지정할 수 있습니다. 날짜가 변환되면 그 날짜는 TOVAR 매개변수에서 지정된 변수명에 놓입니다. 다음 예에서는 MDY로 형식화된 변수 &DATE가 DMY 형식으로 변경되어 변수 &CVTDAT에 들어갑니다.

```
CVTDAT    DATE(&DATE) TOVAR(&CVTDAT) FROMFMT(*MDY) TOFMT(*DMY)
          TOSEP(*SYSVAL)
```

날짜 분리자는 시스템 값 QDATSEP에 지정된 대로 남아 있습니다.

CVTDAT 명령은 오브젝트를 작성하거나 날짜를 멤버명의 일부로 사용하는 멤버를 추가할 때 유용합니다. 예를 들어, 멤버는 현재의 시스템 날짜를 사용하여 파일에 추가되어야 하고, 현재 날짜는 MDY 형식에서 율리우스력 형식으로 변환된다고 가정해 보겠습니다.

```
   PGM
DCL &DATE6 *CHAR  LEN(6)
DCL &DATE5  *CHAR  LEN(5)
RTVSYSVAL  QDATE  RTNVAR(&DATE6)
CVTDAT  DATE(&DATE6)  TOVAR(&DATE5)  TOFMT(*JUL)  TOSEP(*NONE)
ADDPFM  LIB1/FILEX  MBR('MBR'  *CAT  &DATE5)
```

```
.
.
.
ENDPGM
```

현재 날짜가 1988년 1월 5일이면 추가되는 멤버의 이름은 MBR88005입니다.

날짜를 변환할 때에는 다음 사항에 유의해야 합니다.

- DATE 매개변수 값의 길이와 TOVAR 매개변수의 변수 길이는 날짜 형식과 호환되는 것이어야 합니다. TOVAR 매개변수의 변수 길이는 적어도 다음과 같아야 합니다.
-

  1. 2자리 연도를 갖는 율리우스력이 아닌 날짜의 경우
     a. 분리자 없이 사용할 때는 6자를 사용합니다.

        1978년 7월 28일은 072878로 씁니다.

     b. 분리자를 사용할 때는 8자를 사용합니다.

        1978년 7월 28일은 07-28-78로 씁니다.
  2. 4자리 연도를 갖는 율리우스력이 아닌 날짜의 경우
     a. 분리자 없이 사용할 때는 8자를 사용합니다.

        1978년 7월 28일은 07281978로 씁니다.

     b. 분리자를 사용할 때는 10자를 사용합니다.

        1978년 7월 28일은 07-28-1978로 씁니다.
  3. 2자리 연도를 갖는 율리우스력 날짜의 경우
     a. 분리자 없이 사용할 때는 5자를 사용합니다.

        1996년 12월 31일은 96365로 씁니다.

     b. 분리자를 사용할 때는 6자를 사용합니다.

        1996년 12월 31일은 96-365로 씁니다.
  4. 4자리 연도를 갖는 율리우스력 날짜의 경우
     a. 분리자를 사용하지 않을 때는 7자가 필요합니다.

        1997년 2월 4일은 1997035로 씁니다.

     b. 분리자를 사용할 때는 8자가 필요합니다.

        1997년 2월 4일은 1997-035로 씁니다.

  변환된 문자가 변수에 적합하지 않으면 오류 메세지가 송신됩니다. 변환된 날짜가 변수 길이보다 짧으면 오른쪽이 공백으로 채워집니다.
- 율리우스력 형식을 제외한 모든 날짜 형식에서 월, 일은 그 값에 관계 없이 각각 2바이트 필드입니다. 연도는 2바이트나 4바이트 필드일 수 있습니다. 변환된 모든 값은 우측 정렬되며, 필요한 경우 앞부분 빈칸은 0으로 채워집니다.

- 율리우스력 형식인 경우 일은 3바이트 필드이고 연도는 2바이트나 4바이트 필드입니다. 변환된 모든 값은 우측 정렬되며, 필요한 경우 앞부분 빈칸은 0으로 채워집니다.

다음은 날짜를 율리우스력 형식으로 변환하기 위해 ILE에 바인드할 수 있는 API인 CEELOCT(현지 시간 사용) 명령을 사용하는 대체 프로그램입니다. 이 프로그램을 작성하려면 CRTBNDCL(바인드 제어 언어 프로그램 작성) 명령만을 사용하거나 CRTCLMOD(제어 언어 모듈 작성) 명령과 CRTPGM(프로그램 작성) 명령을 함께 사용해야 합니다.

```
   PGM
DCL  &LILDATE  *INT   LEN(4)
DCL  &PICTSTR  *CHAR  LEN(5)  VALUE(YYDDD)
DCL  &JULDATE  *CHAR  LEN(5)
DCL  &SECONDS  *CHAR  8       /*  Seconds from CEELOCT  */
DCL  &GREG     *CHAR  23      /*  Gregorian date from CEELOCT  */
                              /*                                  */
CALLPRC  PRC(CEELOCT)         /*  Get current date and time    */ +
         PARM(&LILDATE        /* Date in Lilian format           */ +
              &SECONDS        /* Seconds field will not be used  */ +
              &GREG           /* Gregorian field will not be used */ +
              *OMIT)          /* Omit feedback parameter         */ +
                              /*  so exceptions are signalled    */

CALLPRC  PRC(CEEDATE) +
         PARM(&LILDATE        /* Today's date     */ +
                &PICTSTR      /*  How to format               */ +
                &JULDATE      /*  Julian date                 */ +
              *OMIT)

ADDPFM   LIB1/FILEX  MBR('MBR'  *CAT   &JULDATE')

ENDPGM
```

ILE API에 대한 자세한 정보는 iSeries Information Center의 **프로그래밍** 범주를 참조하십시오.

구성 소스 검색

RTVCFGSRC(구성 소스 검색) 명령을 사용하면 기존 구성 오브젝트를 작성하기 위해 CL 명령 소스를 생성하여 그 소스를 소스 파일 멤버 안에 놓을 수 있습니다. 생성된 CL 명령 소스는 다음 작업에 사용될 수 있습니다.

- 시스템 간의 구성 이동
- 온사이트 구성의 유지보수
- 구성의 저장(SAVSYS 명령을 사용하지 않고)

구성 상태 검색

RTVCFGSTS(구성 상태 검색) 명령을 사용하면 세 가지의 구성 오브젝트인 행, 제어기, 장치로부터 구성 상태를 검색하는 능력을 어플리케이션에 부여할 수 있습니다. CL 프로시듀어에서 RTVCFGSTS 명령을 사용하면 구성 설명 상태를 검사할 수 있습니다.

네트워크 속성 검색

RTVNETA(네트워크 속성 검색) 명령을 사용하면 시스템의 네트워크 속성을 검색할 수 있습니다. 이와 같은 속성들은 CHGNETA(네트워크 속성 변경) 명령을 사용하여 변경할 수 있고 DSPNETA(네트워크 속성 표시) 명령을 사용하여 화면에 표시할 수 있습니다. 네트워크 속성에 대한 자세한 정보는 iSeries Information Center에 있는 정보의 **프로그래밍** 범주를 참조하십시오.

RTVNETA 예

다음 예에서는 디폴트 네트워크 출력 대기행렬과 이 대기행렬을 포함한 라이브러리가 검색되어 QGPL/QPRINT로 변경되며 나중에 다시 이전 값으로 변경됩니다.

```
   PGM
DCL  VAR(&OUTQNAME) TYPE(*CHAR) LEN(10)
DCL  VAR(&OUTQLIB) TYPE(*CHAR) LEN(10)
RTVNETA  OUTQ(&OUTQNAME) OUTQLIB(&OUTQLIB)
CHGNETA  OUTQ(QGPL/QPRINT)
.
.
.
CHGNETA  OUTQ(&OUTQLIB/&OUTQNAME)
ENDPGM
```

작업 속성 검색

작업 속성을 검색한 후에는 그 값을 CL 변수에 넣어 어플리케이션을 제어할 수 있습니다.

작업 속성은 RTVJOBA(작업 속성 검색) 명령을 사용하여 검색됩니다. RTVJOBA 명령을 사용해 모든 작업 속성이나 여러 가지로 조합된 속성을 검색할 수 있습니다.

다음 CL 프로시듀어에서는 RTVJOBA 명령이 프로시듀어를 호출한 사용자의 이름을 검색합니다.

```
   PGM
         /* ORD410C Order entry program  */
        DCL &CLKNAM TYPE(*CHAR) LEN(10)
        DCL &NXTPGM TYPE(*CHAR) LEN(3)
        .
        .
        .
        RTVJOBA USER(&CLKNAM)
BEGIN:  CALL ORD410S2 PARM(&NXTPGM &CLKNAM)
         /* Customer prompt */
        IF (&NXTPGM *EQ 'END') THEN(RETURN)
        .
        .
        .
```

먼저, 사용자명이 전달될 변수 &CLKNAM을 DCL 명령을 사용하여 선언합니다. RTVJOBA 명령은 선언 명령 다음에 옵니다. 프로그램 ORD410S2가 호출되면 두 개

의 변수 &NXTPGM과 &CLKNAM이 프로그램으로 전달됩니다. &NXTPGM은 공백으로서 전달되지만 ORD410S2에 의해 변경될 수 있습니다.

RTVJOBA 예

다음 CL 프로시듀어에서 대화식 작업이 CL 프로시듀어를 포함한 프로그램을 일괄처리로 제출한다고 가정해 보겠습니다. RTVJOBA(작업 속성 검색) 명령은 작업 완료 메세지를 송신할 메세지 대기행렬명을 검색하고 작업을 제출한 사용자와의 통신을 위해 메세지 대기행렬을 사용합니다.

```
   PGM
DCL &MSGQ *CHAR 10
DCL &MSGQLIB *CHAR 10
DCL &MSGKEY *CHAR 4
DCL &REPLY *CHAR 1
DCL &ACCTNO *CHAR 6
.
.
.
RTVJOBA SBMMSGQ(&MSGQ) SBMMSGQLIB(&MSGQLIB)
IF (&MSGQ *EQ '*NONE') THEN(DO)
   CHGVAR &MSGQ 'QSYSOPR'
   CHGVAR &MSGQLIB 'QSYS'
 ENDDO
.
.
.
IF (. . . ) THEN(DO)
 SNDMSG:SNDPGMMSG MSG('Account number ' *CAT &ACCTNO *CAT 'is +
                  not valid. Do you want to cancel the update +
                  (Y or N)?') TOMSGQ(&MSGQLIB/&MSGQ) MSGTYPE(*INQ) +
                  KEYVAR(&MSGKEY)
 RCVMSG MSGQ(*PGMQ) MSGTYPE(*RPY) MSGKEY(&MSGKEY) +
          MSG(&REPLY) WAIT(*MAX)
 IF (&REPLY *EQ 'Y') THEN(RETURN)
 ELSE IF (&REPLY *NE 'N') THEN(GOTO SNDMSG)
 ENDDO
.
.
.
```

두 변수 &MSGQ와 &MSGQLIB를 선언하여 사용될 메세지 대기행렬의 이름과 라이브러리를 수신합니다. 메세지 대기행렬명과 라이브러리명을 검색하기 위해 RTVJOBA 명령이 사용됩니다. 메세지 대기행렬이 작업용으로 지정되지 않을 수 있으므로 메세지 대기행렬명이 *NONE 값과 비교됩니다. 비교 결과 같다면, 메세지 대기행렬이 지정되지 않은 것이며, 라이브러리 QSYS 안의 메세지 대기행렬 QSYSOPR을 사용할 수 있도록 변수가 변경됩니다. 프로시듀어에서 이후에 오류 조건이 검출되면 조회 메세지가 지정 메세지 대기행렬로 송신되고 응답이 수신되어 처리됩니다. RTVJOBA 명령은 아래와 같이 다른 목적으로 사용될 수도 있습니다.

- 하나 이상의 작업 속성(예: 출력 대기행렬이나 라이브러리 리스트)을 검색하여 이 속성을 일시적으로 변경시킨 후 초기 값으로 복원시킵니다.

- SBMJOB 명령을 사용하기 위해 하나 이상의 작업 속성을 검색하여 피제출 작업과 제출 작업의 속성을 같게 만듭니다.

오브젝트 설명 검색

RTVOBJD(오브젝트 설명 검색) 명령을 사용하면 특정 오브젝트의 설명을 CL 프로시듀어로 리턴시킬 수 있습니다. 설명을 리턴시키기 위해서는 변수가 사용됩니다. 이러한 설명은 사용되지 않는 오브젝트를 검출하는 데 도움을 줍니다. 오브젝트 설명 검색에 대해 더 자세히 알려면 144 페이지의 『오브젝트 설명 검색』 부분을 참조하십시오.

QUSROBJD(오브젝트 설명 검색) 어플리케이션 프로그래밍 인터페이스(API)를 사용하면 특정 오브젝트의 설명을 프로시듀어로 리턴시킬 수도 있습니다. 시스템은 변수를 사용하여 설명을 리턴시킵니다. 자세한 정보는 **iSeries Information Center**의 *프로그래밍* 범주에 있는 *API* 섹션을 참조하십시오.

사용자 프로파일 속성 검색

RTVUSRPRF(사용자 프로파일 속성 검색) 명령을 사용하면 사용자 프로파일의 속성(암호는 제외)을 검색하여 그 값을 CL 변수에 놓고 어플리케이션을 제어할 수 있습니다. 이 명령에는 10자의 사용자 프로파일명이나 *CURRENT를 지정할 수 있습니다.

RTVUSRPRF 명령을 수행한 후에는 이탈 메세지를 모니터할 수도 있습니다. 이 정보에 대해서는 **iSeries Information Center**의 *프로그래밍* 범주에서 *CL* 섹션을 참조하십시오.

RTVUSRPRF 예

다음 CL 프로시듀어에서는 RTVUSRPRF 명령이 프로시듀어를 호출한 사용자의 이름과 그 사용자에 대한 메세지를 송수신할 메세지 대기행렬의 이름을 검색합니다.

```
DCL  &USR  *CHAR  10
DCL  &USRMSGQ  *CHAR  10
DCL  &USRMSGQLIB  *CHAR  10
.
.
.
RTVUSRPRF  USRPRF(*CURRENT)  RTNUSRPRF(&USR)  +
           MGSQ(&USRMSGQ)  MSGQLIB(&USRMSGQLIB)
```

다음의 정보가 프로시듀어로 리턴됩니다.

- &USR은 프로그램을 호출한 사용자의 사용자 프로파일명을 가지고 있습니다.
- &USRMSGQ는 사용자 프로파일에 지정된 메세지 대기행렬의 이름을 가지고 있습니다.
- &USRMSGQLIB는 사용자 프로파일과 연관된 메세지 대기행렬을 포함한 라이브러리의 이름을 가지고 있습니다.

멤버 설명 정보 검색

RTVMBRD(멤버 설명 검색) 명령을 사용하면 어플리케이션에 사용하기 위한 데이터베이스 파일의 멤버에 대한 정보를 검색할 수 있습니다.

RTVMBRD 예

다음 CL 프로시듀어에서 RTVMBRD 명령은 특정 멤버의 설명을 검색합니다. MFILE이라는 데이터베이스 파일이 현재 라이브러리(MYLIB) 안에 존재하며 세 개의 멤버(AMEMBER, BMEMBER, CMEMBER)를 가지고 있다고 가정하겠습니다.

```
DCL &LIB TYPE(*CHAR) LEN(10)
DCL  &MBR      TYPE(*CHAR) LEN(10)
DCL  &SYS      TYPE(*CHAR) LEN(4)
DCL  &MTYPE    TYPE(*CHAR) LEN(5)
DCL  &CRTDATE  TYPE(*CHAR) LEN(13)
DCL  &CHGDATE  TYPE(*CHAR) LEN(13)
DCL  &TEXT     TYPE(*CHAR) LEN(50)
DCL  &NBRRCD   TYPE(*DEC)  LEN(10 0)
DCL  &SIZE     TYPE(*DEC)  LEN(10 0)
DCL  &USEDATE  TYPE(*CHAR) LEN(13)
DCL  &USECNT   TYPE(*DEC)  LEN(5 0)
DCL  &RESET    TYPE(*CHAR) LEN(13)
.
.
.
RTVMBRD    FILE(*CWeb siteIB/MYFILE) MBR(AMEMBER *NEXT) +
           RTNLIB(&LIB) RTNSYSTEM(&SYS) RTNMBR(&MBR) +
           FILEATR(&MTYPE) CRTDATE(&CRTDATE) TEXT(&TEXT) +
           NBRCURRCD(&NBRRCD) DTASPCSIZ(&SIZE) USEDATE(&USEDATE) +
           USECOUNT(&USECNT) RESETDATE(&RESET)
```

다음의 정보가 프로시듀어로 리턴됩니다.

- 현재 라이브러리명(MYLIB)이 CL 변수명 &LIB에 놓여 있습니다.
- MYFILE이 발견된 시스템이 CL 변수명 &SYS에 놓여 있습니다(*LCL은 파일이 로컬 시스템에서 발견되었음을 의미하며, *RMT는 파일이 리모트 시스템에서 발견되었음을 의미함).
- BMEMBER가 이름순 멤버 리스트(*NEXT)에서 AMEMBER 바로 다음에 나오기 때문에 멤버명(BMEMBER)이 CL 변수 &MBR에 놓여 있습니다.
- MYFILE의 파일 속성이 CL 변수명 &MTYPE에 놓여 있습니다(*DATA는 멤버가 자료 멤버임을 의미하며. *SRC는 파일이 소스 멤버임을 의미함).
- BMEMBER의 작성 날짜가 호출된 CL 변수 &CRTDATE에 놓여 있습니다.
- BMEMBER를 설명하는 데 사용된 텍스트가 호출된 CL 변수 &TEXT에 놓여 있습니다.
- BMEMBER에 있는 레코드의 현재 번호가 호출된 CL 변수 &NBRRCD에 놓여 있습니다.
- BMEMBER의 자료 공간 크기(바이트)가 호출된 CL 변수 &SIZE에 놓여 있습니다.

- BMEMBER가 마지막으로 사용된 날짜가 호출된 CL 변수 &USEDATE에 놓여 있습니다.
- BMEMBER가 사용된 일 수가 호출된 CL 변수 &USECNT에 놓이고, 이 계수의 시작 날짜가 호출된 CL 변수 &RESET에 놓여 있습니다.

CL 프로시듀어에 대한 작업

CL 소스 프로시듀어는 실행에 앞서 모듈로 컴파일하여 프로그램에 바인드해야 합니다.

한 번의 단계로 CL 프로그램을 작성하려는 경우 CRTBNDCL(바인드 제어 언어 프로그램 작성) 명령을 사용할 수 있으며 모듈이 하나인 바인드 프로그램을 작성할 수 있습니다.

또한 CRTCLMOD(제어 언어 모듈 작성) 명령을 사용하여 모듈을 작성할 수도 있습니다. 그런 다음 모듈이 CRTPGM(프로그램 작성) 또는 CRTSRVPGM(서비스 프로그램 작성) 명령을 사용하여 프로그램이나 서비스 프로그램으로 바인드되어야 합니다.

다음 예에서는 모듈 ORD040C를 작성하여 라이브러리 DSTPRODLB에 놓습니다.

```
CRTCLMOD     MODULE(DSTPRODLB/ORD040C) SRCFILE(QCLSRC)
             TEXT('Order dept general menu program')
```

ORD040C에 대한 소스 명령은 소스 파일 QCLSRC에 들어 있으며 소스 멤버명은 ORD040C입니다. 디폴트로 컴파일러 리스팅이 작성됩니다.

CRTBNDCL(바인드 제어 언어 프로그램 작성) 명령에서 리스팅 옵션 및 프로그램이 프로그램 소유자의 사용자 프로파일에 따라 작동해야 하는지 여부를 지정할 수 있습니다.

프로그램은 소유자의 프로파일 또는 사용자의 프로파일을 사용하여 실행될 수 있습니다.

CL 프로시듀어와 프로그램은 프로그래밍 개발 관리자(PDM) 메뉴나 프로그래머 메뉴의 옵션을 사용하여 작성되므로 CRTCLMOD(제어 언어 모듈 작성) 명령 또는 CRTBNDCL을 직접 입력할 필요가 없습니다.

CL 프로시듀어 명령 기록(logging)

프로시듀어를 컴파일할 때 CRTCLMOD(제어 언어 모듈 작성) 명령 또는 CRTBNDCL(바인드 제어 언어 프로그램 작성) 명령의 LOG 매개변수에 다음 값 중 하나를 지정함으로써 CL 프로시듀어로 실행되는 대부분의 CL 명령이 작업 기록부에 기록되도록 지정할 수 있습니다.

***JOB** 이 디폴트 값은 작업의 기록 옵션이 설정되어 있으면 기록됨을 나타냅니다. 이 옵션이 처음에 기록되지 않도록 설정되어 있어도, CHGJOB 명령에서

LOGCLPGM 매개변수로 변경할 수 있습니다. 따라서 이 값을 사용하여 모듈이나 프로그램을 작성한다면, 각각의 작업에 대해서 또는 한 작업 안에서 여러 번 기록 옵션을 변경할 수 있습니다.

***YES** CL 프로시듀어가 수행될 때마다 기록됨을 나타냅니다. 이 값은 CHGJOB 명령으로 변경할 수 없습니다.

***NO** 기록되지 않음을 나타냅니다. 이 값은 CHGJOB 명령으로 변경할 수 없습니다.

이러한 값은 CRTCLMOD(제어 언어 모듈 작성) 및 CRTBNDCL(바인드 제어 언어 프로그램 작성) 명령의 일부이므로 이 값을 변경하려면 모듈 또는 프로그램을 다시 컴파일해야 합니다.

기록을 지정할 때 기록된 명령이 작업 기록부에서 제거되지 않도록 RMVMSG(메세지 제거) 명령을 주의해서 사용해야 합니다. RMVMSG 명령에 CLEAR(*ALL)를 지정하면 RMVMSG 명령이 수행되기 전에 기록된 모든 명령이 작업 기록부에 나오지 않습니다. 이것은 RMVMSG 명령이 들어 있는 CL 프로시듀어에만 영향을 미치며, 이전 또는 다음의 순환 레벨에 대해 기록된 명령에는 영향을 미치지 않습니다.

모든 명령이 작업 기록부에 기록되는 것은 아닙니다. 다음은 기록되지 않는 명령 리스트입니다.

CALLPRC	**CHGVAR**	**DCL**
DCLF	**DO**	**DOFOR**
DOUNTIL	**DOWHILE**	**ELSE**
ENDDO	**ENDPGM**	**ENDSELECT**
GOTO	**IF**	**ITERATE**
LEAVE	**MONMSG**	**OTHERWISE**
PGM	**SELECT**	**WHEN**

기록 옵션이 설정되어 있으면, 기록될 메세지가 CL 프로시듀어의 메세지 대기행렬로 송신됩니다. CL 프로시듀어가 대화식으로 실행 중이고 작업의 LOG 매개변수상의 메세지 레벨이 4로 설정된 경우 F10(상세 메세지 표시) 키를 눌러 모든 명령의 기록을 볼 수 있습니다. 메세지 레벨이 4이고 사인 오프 시 *PRINT를 지정하면 기록부를 인쇄할 수 있습니다.

기록부에는 시간, 프로그램과 프로시듀어명, 메세지 텍스트 및 명령어가 들어 있습니다. 명령어는 원래의 소스문에 들어 있는 대로 규정됩니다. 명령 매개변수 또한 기록되며, 매개변수 정보가 CL 변수인 경우 변수 내용이 인쇄됩니다(RTNVAL 매개변수는 제외).

명령의 기록은 성능에 영향을 미칩니다.

CL 모듈 컴파일러 리스팅

CL 모듈을 작성할 때 CRTCLMOD(제어 언어 모듈 작성) 명령에 OPTION 및 OUTPUT 매개변수를 사용하여 다양한 유형의 리스팅을 작성할 수 있습니다.

OPTION 매개변수 값과 그 의미는 다음과 같습니다.

- *GEN 또는 *NOGEN

  모듈이 작성되는지의 여부(*GEN은 디폴트임).

- *XREF 또는 *NOXREF

  소스 입력에서 변수 및 자료 참조에 대한 상호 참조 리스팅이 생성될 것인지 여부(*XREF가 디폴트임).

OUTPUT 매개변수 값과 그 의미는 다음과 같습니다.

- *PRINT - 리스트 인쇄
- *NONE - 컴파일러 리스트 없음

OUTPUT 매개변수를 지정함으로써 작성되는 리스팅을 *컴파일러 리스팅*이라고 합니다. 다음은 샘플 컴파일러 리스팅입니다. 설명 번호는 리스팅 다음에 오는 설명을 나타냅니다.

```
 1                                                                                                  2                    3
5722SS1 V5R3M0  041231                Control Language        MYLIB/DUMPER          SYSNAME  05/06/00 11:12:55       Page 1
Module  . . . . . . . . . . . . . . . . . . . . :   DUMPERR
  Library . . . . . . . . . . . . . . . . . . . :     MYLIB
Source file . . . . . . . . . . . . . . . . . . :   QCLSRC
  Library . . . . . . . . . . . . . . . . . . . :     MYLIB
Source member name  . . . . . . . . . . . . . . :   DUMPERR   05/06/94 10:42:26  4
Source printing options . . . . . . . . . . . . :   *XREF  *NOSECLVL  *NOEVENTF
Module logging  . . . . . . . . . . . . . . . . :   *JOB
Replace module object . . . . . . . . . . . . . :   *YES
Target release  . . . . . . . . . . . . . . . . :   V5R3M0
Authority . . . . . . . . . . . . . . . . . . . :   *LIBCRTAUT
Sort sequence . . . . . . . . . . . . . . . . . :   *HEX
Language identifier . . . . . . . . . . . . . . :   *JOBRUN
Text  . . . . . . . . . . . . . . . . . . . . . :   Test program
Optimization  . . . . . . . . . . . . . . . . . :   *NONE
Debugging view  . . . . . . . . . . . . . . . . :   *STMT
Enable performance collection . . . . . . . . . :   *PEP
Compiler  . . . . . . . . . . . . . . . . . . . :   IBM iSeries Control Language Compiler  5
 6                          Control Language Source
SEQNBR  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+.  DATE  8
   100- PGM                                                                                                 05/06/94
   200- DCL &ABC *CHAR 10 VALUE('THIS')                                                                     05/06/94
   300- DCL &XYZ *CHAR 10 VALUE('THAT') 7                                                                   05/06/94
   400- DCL &MNO *CHAR 10 VALUE('OTHER')                                                                    05/06/94
   500- CRTLIB LB(LARRY)                                                                                    05/06/94
* CPD0043 30  Keyword LB not valid for this command.  9
   600- DLTLIB LIB(MOE                                                                                      05/06/94
* CPD0013 30  A matching parenthesis not found.
   700- MONMSG CPF0000 EXEC(GOTO ERR)                                                                       05/06/94
   800- ERROR:                                                                                              05/06/94
   900- CHGVAR &ABC 'ONE'                                                                                   05/06/94
  1000- CHGVAR &XYZ 'TWO'                                                                                   05/06/94
  1100- CHGVAR &MNO 'THREE'                                                                                 05/06/94
  1200- DMPCLPGM                                                                                            05/06/94
  1300- ENDPGM                                                                                              05/06/94
                                    * * * * *  E N D  O F  S O U R C E  * * * * *
5722SS1 V5R3M0  040201                Control Language        MYLIB/DUMPER          SYSNAME  05/06/00 11:12:55       Page 2
                                       Cross Reference
Declared Variables
Name             Defined     Type            Length      References
&ABC                 200     *CHAR             10          900
&MNO                 400     *CHAR             10         1100
10
&XYZ                 300     *CHAR             10         1000
Defined Labels
Label            Defined     References  11
ERR              ******          700
* CPD0715 30  Label 'ERR       ' does not exist.
ERROR                800
                             * * * * *  E N D  O F  C R O S S  R E F E R E N C E  * * * * *
5722SS1 V5R3M0  040201                Control Language        MYLIB/DUMPER          SYSNAME  05/06/04 11:12:55       Page 3
                                       Message Summary
              Severity
Total         0-9  10-19  20-29  30-39  40-49  50-59  60-69  70-79  80-89  90-99  12
      3         0      0      0      3      0      0      0      0      0      0
Module DUMPERR not created in library MYLIB. Maximum error severity 30.  13
                               * * * * *  E N D  O F  M E S S A G E  S U M M A R Y  * * * * *
                                  * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

제목:

1 OS/400의 프로그램 번호, 릴리스, 수정 레벨 및 날짜

2 컴파일러 수행 날짜와 시간

3 리스팅의 페이지 번호

프롤로그:

4 CRTCLMOD(제어 언어 모듈 작성) 명령에 지정된 매개변수 값(지정되지 않은 경우에는 디폴트). 소스가 데이터베이스 파일에 없으면 멤버명, 날짜 및 시간은 생략됩니다.

5 컴파일러명

소스:

6 소스에서의 행(레코드) 순번. 순번 뒤의 대시(-)는 소스문이 그 순번에서 시작됨을 나타냅니다. 대시(-)가 없으면 그 명령문이 이전 명령문에서 연속됨을 의미합니다.

소스문 간의 주석은 다른 소스문과 같이 처리되며 순번도 가집니다.

7 소스문

8 소스문이 변경되거나 추가된 최종일. 소스가 데이터베이스 파일에 없거나, RGZPFM을 사용하여 날짜가 재설정된 경우 날짜는 생략됩니다.

9 컴파일하는 동안 오류가 발생하고, 특정 소스문으로 추적되면 오류 메세지가 소스문 바로 다음에 인쇄됩니다. 별표(*)는 그 행에 오류 메세지가 있음을 나타냅니다. 행에는 메세지 ID, 심각도 및 메세지 텍스트가 포함되어 있습니다.

컴파일 오류에 대해 더 자세히 알려면 74 페이지의 『컴파일 시의 오류』 부분을 참조하십시오.

상호 참조:

10 기호 변수 표는 프로그램에 유효하게 선언되어 있는 변수의 상호 참조 리스팅입니다. 이 표에는 변수, 변수가 선언된 명령문의 순번, 변수의 속성, 변수를 참조하는 명령문의 순번 등이 있습니다.

11 레이블 표는 프로그램에 유효하게 정의되어 있는 레이블의 상호 참조 리스팅입니다. 이 표에는 레이블, 레이블이 정의되는 명령문의 순번, 레이블을 참조하는 명령문의 순번 등이 있습니다.

메세지:

샘플 모듈에 대해서는 일반 오류 메세지가 발행되지 않으므로 샘플 리스팅에는 이 섹션이 없습니다. 이 모듈에 대한 일반 오류 메세지가 있으면, 이 섹션에 각 메세지에 대한 메세지 ID, 심각도 및 메세지가 포함됩니다.

메세지 요약:

12 컴파일하는 동안 발행했던 메세지 수의 요약. 총계가 심각도별 합계와 함께 표시됩니다.

13 메세지 요약 다음에 완료 메세지가 인쇄됩니다.

*SOURCE 옵션이 지정되면 제목, 머리말, 소스 및 메세지 요약 섹션이 항상 인쇄됩니다. *XREF 옵션이 지정되면 상호 참조 섹션이 인쇄됩니다. 일반적인 오류가 발견될 때만 메세지 섹션이 인쇄됩니다.

컴파일 시의 오류

모듈의 컴파일러 리스팅에서 특정 명령과 직접 관련된 오류 상태는 그 명령 다음에 나열됩니다. 이러한 인라인 메세지의 예를 보려면 71 페이지의 『CL 모듈 컴파일러 리스팅』 부분을 참조하십시오. 특정 명령에 관련되지 않지만 사실상 보다 일반적인 메세지는 소스 명령문에 인라인되지 않고 리스팅의 메세지 섹션에 나열됩니다.

컴파일 시 발견되는 오류의 유형에는 구문 오류, 정의되지 않은 변수와 레이블에 대한 참조 오류, 명령문 누락 오류 등이 있습니다. 다음과 같은 유형의 오류가 발생하면 프로시듀어 작성이 중단됩니다(심각도 코드는 무시됨).

- 값 오류
- 구문 오류
- 한 명령 안에서 매개변수 간의 종속성에 관계된 오류
- 유효성 검사 시 발견된 오류

프로시듀어의 작성을 중단시킨 오류가 발견된 후에도 컴파일러는 계속해서 소스의 오류를 검사합니다. 이렇게 하여 모듈이나 프로그램 작성을 다시 시도하기 전에 가능한 한 많은 오류를 찾아 정정할 수 있습니다.

프로시듀어 덤프 확보

프로시듀어가 처리되는 동안 CL 프로시듀어 덤프를 구할 수 있습니다. CL 프로시듀어 덤프는 프로시듀어의 메세지 대기행렬에 있는 모든 메세지의 리스팅과 프로시듀어에 사용된 모든 변수의 값들로 구성됩니다. 이러한 정보는 프로시듀어 처리에 영향을 미치는 문제점의 원인을 판별하는 데 유용합니다.

CL 프로시듀어 덤프를 구하려면 다음 중 하나를 수행하십시오.

- DMPCLPGM(CL 프로그램 덤프) 명령을 수행합니다. 이 명령은 CL 프로시듀어에서만 사용할 수 있으며 CL 프로시듀어를 종료시키지 않습니다.
- 조회 메세지 CPA0701 또는 CPA0702에 대한 응답으로 D를 입력합니다. CL 프로시듀어로부터 모니터되지 않은 이탈 메세지를 수신할 때마다 시스템이 이 메세지를 송신합니다. 프로그램이 대화식 작업으로 실행 중이면, 시스템이 메세지를 작업의 외부 메세지 대기행렬로 송신합니다. 프로그램이 일괄처리 작업으로 실행되는 중이면, 시스템이 메세지를 시스템 오퍼레이터 메세지 대기행렬인 QSYSOPR로 송신합니다.

- 작업에 대해 INQMSGRPY(*SYSRPYL)를 지정합니다. 이 작업 속성의 설명에 대해서는 iSeries Information Center에 있는 정보의 **시스템 관리** 범주를 참조하십시오. IBM 제공 시스템 응답 리스트는 메세지 CPA0702와 CPA0701에 대해 D라는 응답을 지정합니다. 조회 메세지 중 하나를 수신할 경우에는 시스템이 덤프를 인쇄합니다.
- 메세지 CPA0701이나 CPA0702에 대한 디폴트 응답(default reply)을 C(프로그램 취소)에서 D(프로시듀어 덤프)로 변경합니다. 이 결과 CL 프로시듀어에서 기능 검사가 발생할 때마다 프로시듀어 덤프가 인쇄됩니다. 디폴트를 변경하려면 아래의 명령을 입력하십시오.

  ```
  CHGMSGD MSGID(CPA0702) MSGF(QCPFMSG) DFT(D)
  ```

  주: CHGMSGD 명령은 보안 담당자 또는 QCPFMSG 파일에 대한 갱신 권한이 있는 다른 사용자가 입력해야 합니다.

다음 조건하에서 메세지 디폴트를 변경하면 덤프가 인쇄됩니다.

- 시스템 오퍼레이터 메세지 대기행렬이 디폴트 모드이고 메세지가 일괄처리 작업으로부터 송신될 때.
- 표시장치 사용자가 응답을 입력하지 않은 채 Enter 키만 눌러 메세지 디폴트가 사용되었을 때.
- INQMSGRPY(*DFT)가 작업에 지정되었을 때.

```
                1
5722SS1 V5R3M0  040201               Control Language        MYLIB/DUMPER         SYSNAME  05/06/00  11:05:03 2 Page 1

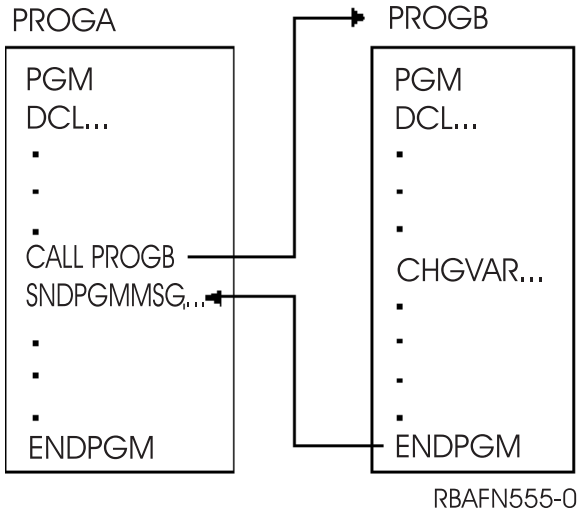
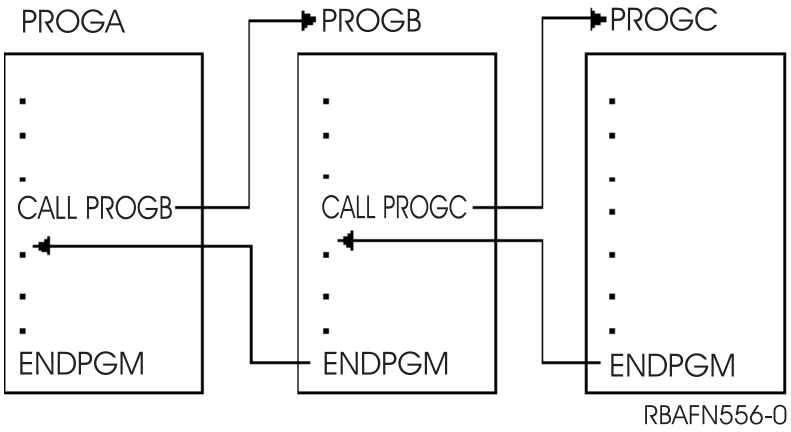
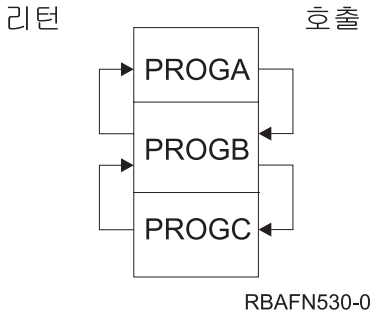
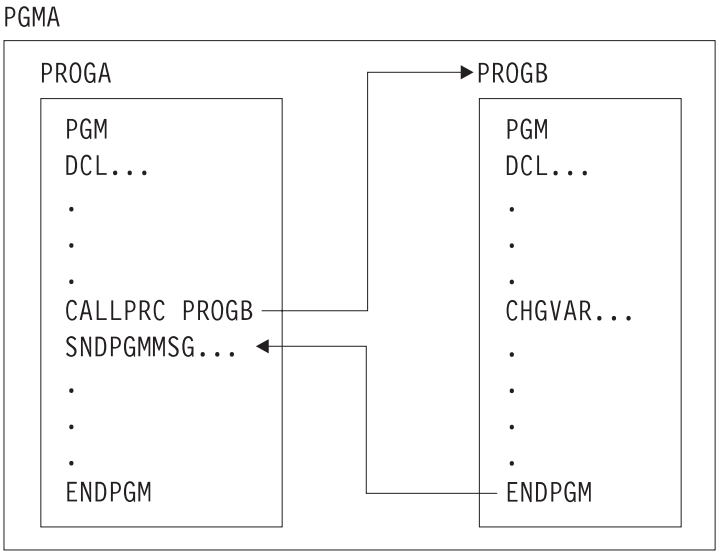
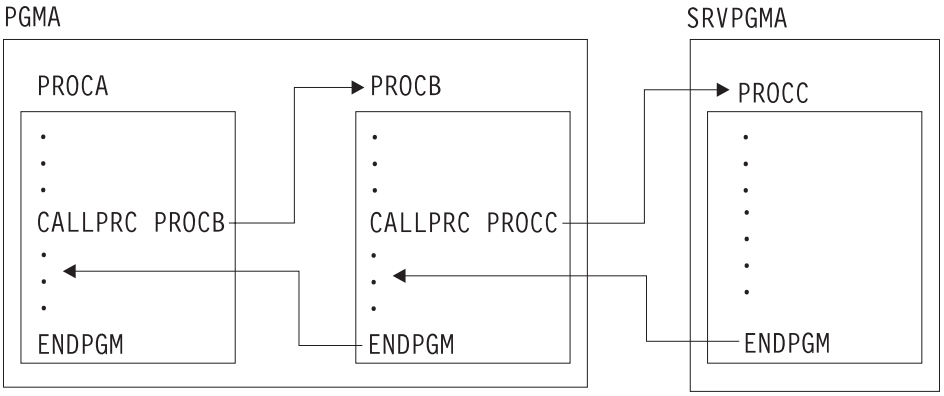
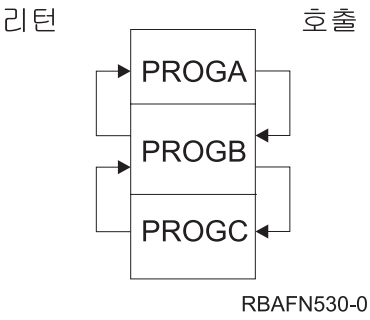
Job name . . . . . :  DSP04 3  User name . . . . . :  SMITH 3  Job number  . . . . . : 01329 3
Program name . . . . . :  DUMP 4  Library . . . . . :  MYLIB 4  Statement . . . . . : 1200 5
Module name  . . . . . :  DUMP    Procedure name  . . . :  DUMP

                                                  Messages

           Message 6                Message                        From                        To
Time       ID              Sev      Type    Text                   Program         Inst        Program         Inst
110503     CPC2102         00       COMP    Library LARRY created. QLICRLIB        *N          DUMP            *N
110503     CPF2110         40       ESC     Library MOE not found. QLICLLIB        *N          DUMP            *N

                                                 Variables 7

Variable          Type       Length          Value                          Value in Hexadecimal
                                             *...+....1....+....2....+      * . . . + . . . . 1 . . . . + . . . . 2 . . . . +
&ABC              *CHAR              10      'ONE       '                   D6D5C540404040404040
&XYZ              *CHAR              10      'TWO       '                   E3E6D640404040404040

                                     * * * * *  E N D  O F  D U M P  * * * * *
```

1 OS/400의 프로그램 번호, 릴리스, 수정 레벨 및 날짜

2 덤프가 인쇄된 날짜와 시간

3 프로시듀어가 수행되었던 작업이 완전히 규정된 이름

4 프로그램의 이름과 라이브러리

5 덤프가 이루어졌을 때 수행 중이던 명령문 번호. 명령이 내포 명령문인 경우 명령문 번호는 바깥쪽 명령문의 번호입니다.

6 메세지가 송신된 시간, 메세지 ID, 심각도, 유형, 텍스트, 송신 프로그램과 명령어 번호, 그리고 수신 프로그램과 명령어 번호가 포함된 호출 메세지 대기행렬 상의 각 메세지.

7 변수명, 유형, 길이, 값, 16진값을 포함하여 프로그램에서 선언된 모든 변수.

10진 변수에 유효하지 않은 10진 자료가 들어 있는 경우 문자 값과 16진값은 *CHAR 변수로 인쇄됩니다.

변수에 대한 값을 찾을 수 없을 경우 *NOT ADDRESSABLE이 인쇄됩니다. 이것은 TYPE(*NULL) 또는 PASSVAL(*NULL)이 지정된 매개변수를 가진 명령에 대해 CL 프로시듀어가 명령 처리 프로그램에서 사용되거나 매개변수에 대해 RTNVAL(*YES)이 지정되고 리턴 변수가 명령에 코드화되지 않은 경우에 발생할 수 있습니다.

변수가 TYPE(*LGL)으로 선언되면 변수는 길이가 1인 *CHAR로 덤프상에 표시됩니다.

모듈 속성 표시

DSPMOD(모듈 표시) 명령을 사용하면 모듈의 속성을 표시할 수 있습니다. 표시되거나 인쇄된 정보는 모듈 작성에 사용된 명령에서 지정 옵션을 판별하는 데 사용될 수 있습니다.

이 명령에 대한 자세한 정보는 **iSeries Information Center**의 *프로그래밍* 범주에서 *CL* 섹션을 참조하십시오.

프로그램 속성 표시

DAPPGM(프로그램 표시) 명령을 사용하면 프로그램의 속성을 표시할 수 있습니다. 화면에 표시되거나 인쇄된 정보를 사용하여 프로그램 작성에 사용된 명령에 지정되어 있는 옵션을 판별할 수 있습니다.

이 명령에 대한 자세한 정보는 **iSeries Information Center**의 *프로그래밍* 범주에서 *CL* 섹션을 참조하십시오.

리턴 코드 요약

RTVJOBA 명령의 리턴 코드(RTNCDE) 매개변수는 소수 자릿수가 없는 5자리의 10진값입니다. 10진값은 호출된 프로그램의 상태를 나타냅니다. CL 프로그램은 리턴 코드를 설정하지 않습니다. 그러나 다른 프로그램이 CL 프로그램에 설정한 대로 리턴 코드의 현재 값을 검색할 수 있습니다. 이것은 RTVJOBA 명령의 RTNCDE 매개변수를 사용할 수 있습니다.

다음 리스트는 OS/400에서 지원되는 언어가 사용하는 리턴 코드에 대한 요약입니다.

- RPG IV 프로그램

  RPG IV 컴파일러가 송신하는 리턴 코드는 다음과 같습니다.

  0 프로그램이 작성되었을 때

  2 프로그램이 작성되지 않았을 때

실행 중인 RPG IV 프로그램이 송신한 리턴 코드는 다음과 같습니다.

0 프로그램 시작 시 또는 프로그램이 호출되기 전 CALL 조작에 의해

1 LR이 설정되어 프로그램이 종료될 때

2 오류로 인해 프로그램이 종료될 때(조회 메세지에 대한 응답 C, D, F 또는 S)

3 정지 인디케이터(halt indicator)(H1-H9)로 인해 프로그램이 종료될 때

RPG IV 리턴 코드는 CALL 이후에만 테스트됩니다.

– 0 또는 1은 오류가 없음을 나타냅니다.

– 3은 RPG IV 상태 코드 231을 나타냅니다.

– 그 외 다른 값은 RPG IV 상태 코드 202를 나타냅니다(오류로 인해 호출 종료).

RPG IV 프로그램에서는 사용자가 리턴 코드를 직접 테스트할 수 없습니다.

- ILE COBOL/400® 프로그램

  실행 중인 COBOL/400 프로그램이 송신한 리턴 코드는 다음과 같습니다.

  0 프로그램이 호출되기 전 각각의 CALL문에 의해

  2 프로그램에 기능 검사(CPF9999)가 수신되었거나, 총칭 I/O 예외 핸들러가 제어를 받았으나 적용 가능한 USE 프로시듀어가 없을 때

  COBOL/400 프로그램은 이 리턴 코드를 검색할 수 없습니다. 리턴 코드값 2는 메세지 CBE9001을 송신하고 *ABNORMAL 옵션으로 RCLRSC(자원 재생) 명령을 수행합니다.

- C/400* 프로그램

  C/400® 프로그램 안의 최종 C/400 리턴 명령문에서 리턴시킨 정수 리턴 코드의 현재 값.

이전 릴리스의 소스 프로그램 컴파일

CRTCLPGM(제어 언어 프로그램 작성) 명령으로 목표 릴리스(TGTRLS) 매개변수를 사용함으로써 이전 릴리스에 사용할 CL 소스 프로그램을 컴파일할 수 있습니다. TGTRLS 매개변수는 작성된 CL 프로그램 오브젝트를 실행시킬 OS/400 사용권 프로그램의 릴리스를 지정합니다. *CURRENT, *PRV 또는 특정 릴리스 레벨을 지정할 수 있습니다.

TGTRLS(*CURRENT)로 컴파일한 CL 프로그램은 현재 릴리스 또는 후속 릴리스의 오퍼레이팅 시스템에서만 수행됩니다. *CURRENT 이외의 지정 TGTRLS 값으로 컴파일한 CL 프로그램은 지정 릴리스 값과 후속 릴리스에서 수행될 수 있습니다.

이전 릴리스(*PRV) 라이브러리

CL 컴파일러는 이전 릴리스 명령과 이전 릴리스(*PRV)의 CL 라이브러리에 있는 파일 정보를 검색합니다. 시스템 라이브러리와 사용자 라이브러리 두 가지 유형의 라이브러리가 이전 릴리스 지원을 포함하고 있습니다. 라이브러리의 이름은 QSYSVxRxMx와 QUSRVxRxMx입니다. (VxRxMx는 지원되는 이전 릴리스의 버전, 릴리스 및 수정 레벨을 나타냅니다.) 예를 들어, QUSRV4R5M0 라이브러리는 OS/400 사용권 프로그램의 버전 4 릴리스 5 수정 레벨 0을 실행하는 시스템을 지원합니다.

CL 컴파일러는 지원되는 이전 릴리스에 대해 컴파일할 때 먼저 이전 릴리스 라이브러리의 명령과 파일을 검사합니다. 이전 릴리스 라이브러리에서 명령이나 파일을 찾지 못하면 시스템이 라이브러리 리스트(*LIBL)나 규정된 라이브러리를 탐색합니다.

QSYSVxRxMx 라이브러리: QSYSVxRxMx 라이브러리는 이전 릴리스의 CL 컴파일러 지원이 설치될 때 동시에 설치됩니다. QSYSVxRxMx 라이브러리는 명령 정의 오브젝트와 해당되는 특정 이전 릴리스의 라이브러리 QSYS에 있는 출력 파일(*OUTFILE)을 포함하고 있습니다.

QUSRVxRxMx 라이브러리: 원하는 명령과 파일이 지원되는 이전 릴리스에 있었을 때 그 명령과 파일 사본을 보유하는 사용자 자신의 QUSRVxRxMx 라이브러리를 작성할 수 있습니다. 이것은 명령이나 파일이 현재 릴리스에서 변경된 경우 특히 중요합니다.

컴파일러는 이전 릴리스의 명령과 파일을 찾을 때 QSYSVxRxMx 라이브러리를 검사하기 전에 QUSRVxRxMx 라이브러리(있는 경우)를 검사합니다.

주: QSYSVxRxMx 라이브러리 대신 이전 릴리스의 사용자 명령과 파일을 보유하려면 QUSRVxRxMx 라이브러리를 사용하십시오. 후속 릴리스의 CL 컴파일러가 설치될 때 이전 릴리스의 지원도 설치됩니다. 일단 이전 릴리스 지원이 설치되었으면, 더 이상 지원되지 않는 릴리스에 대한 QUSRVxRxMx 라이브러리를 삭제할 수 없습니다.

이전 릴리스의 라이브러리는 라이브러리 리스트(*LIBL)에 추가하지 마십시오. 이전 릴리스의 라이브러리에는 현재 시스템에서 실행할 수 없는 이전 릴리스를 지원하는 명령과 파일이 있습니다. CL 컴파일러만이 이전 릴리스 라이브러리의 명령과 파일을 참조하고 사용합니다. 이전 릴리스를 위해 제공되는 시스템 명령은 시스템의 1차 언어로 되어 있습니다. 2차 언어 버전은 제공되지 않습니다.

오브젝트를 다른 릴리스에 저장하는 방법에 대해서는 해당 CL 참조 명령(SAVOBJ(오브젝트 저장), SAVCHGOBJ(변경된 오브젝트 저장) 또는 SAVLIB(라이브러리 저장))을 보십시오.

주: System/38™ 환경에서 컴파일된 CL 프로그램은 이전 릴리스용으로 저장할 수 없습니다.

이전 릴리스용 CL 컴파일러 지원 설치

*PRV CL 컴파일러 지원과 QSYSVxRxMx 라이브러리를 설치하려면 다음과 같이 하십시오.

1. 아래와 같이 입력합니다.

   ```
   GO LICPGM
   ```

   그러면 사용권 프로그램 메뉴가 나옵니다.
2. 옵션 11(사용권 프로그램 설치)을 선택합니다.
3. 이름이 *PRV CL 컴파일러 지원인 옵션을 선택합니다. 그러면 QSYSVxRxMx 라이브러리가 설치됩니다.

   이전 릴리스의 CL 컴파일러 지원을 사용하지 않는 경우에는 아래와 같이 입력하여 이 지원을 제거할 수 있습니다.

   ```
   DLTLICPGM LICPGM(5769SS1) OPTION(9)
   ```

   CL 컴파일러 지원이 제거될 때 QSYSVxRxMx 라이브러리도 시스템에서 제거되지만 QUSRVxRxMx 라이브러리는 제거되지 않습니다. QUSRVxRxMx 라이브러리가 필요없는 경우에는 DLTLIB(라이브러리 삭제) 명령을 사용하여 직접 삭제해야 합니다.

제 3 장 프로그램과 프로시듀어 간의 흐름 제어 및 통신

사용자는 CALL(프로그램 호출), CALLPRC(바인드 프로시듀어 호출) 및 RETURN(리턴) 명령을 사용하여 프로그램과 프로시듀어 간에 제어 권한을 주고 받을 수 있습니다. 각 명령의 특성은 약간씩 다릅니다. 정보는 제어가 전달될 때 매개변수로서 호출된 프로그램과 프로시듀어에 전달됩니다.

CALL이나 CALLPRC 명령을 수행하는 USRPRF(*OWNER)로 작성된 프로그램에 대해서는 특별히 주의해야 합니다. 이 명령의 보안 특성은 이 명령들이 소유자의 사용자 프로파일하에서 수행되는 프로그램 안에서 처리될 때 달라집니다. 사용자 프로파일에 대한 자세한 정보는 보안 - 참조서 책을 참조하십시오.

CALL 명령

CALL(프로그램 호출) 명령은 해당 명령에 명명된 프로그램을 호출하여 제어 권한을 그 프로그램에 전달합니다. CALL 명령의 형식은 다음과 같습니다.

```
CALL PGM(library-name/program-name) PARM(parameter-values)
```

프로그램명이나 라이브러리명은 변수일 수 있습니다. 호출된 프로그램이 라이브러리 리스트에 없는 라이브러리 안에 있다면 PGM 매개변수에 프로그램의 규정된 이름을 지정해야 합니다. PARM 매개변수는 84 페이지의 『프로그램과 프로시듀어 간의 매개변수 전달』 부분에서 설명합니다. 호출된 프로그램이 수행을 종료하면 호출 프로그램의 그 다음 명령으로 제어가 리턴됩니다.



서로를 호출하는 프로그램 세트 안에서 CALL 명령의 순서를 호출 스택이라 합니다. 한 예로 이 시리즈에서는



다음과 같이 처리됩니다. 호출 스택:



PROGC가 처리를 종료하면 PROGB 안에서 PROGC를 호출한 명령 바로 다음의 명령으로 제어가 리턴됩니다. 즉, 제어는 호출 스택에서 위쪽 방향으로 리턴됩니다. 이것은 PROGC가 RETURN 또는 ENDPGM 명령으로 종료되는지의 여부에 관계 없이 발생합니다.

CL 프로그램은 그 자신을 호출할 수 있습니다.

CALLPRC 명령

CALLPRC(바운드 프로시듀어 호출) 명령은 해당 명령에 명명된 프로시듀어를 호출하여 제어를 그 프로시듀어에 전달합니다. CALLPRC 명령의 형식은 다음과 같습니다.

```
CALLPRC PRC(프로시듀어-명) PARM(매개변수-값) RTNVAL(리턴-값-변수)
```

프로시듀어명은 변수일 수 없습니다. PARM 매개변수는 84 페이지의 『프로그램과 프로시듀어 간의 매개변수 전달』 부분에서 설명합니다. 호출된 프로시듀어가 실행을 마치면, 제어가 호출 프로시듀어의 다음 명령으로 리턴됩니다.



RBAFN539-0

서로를 호출하는 프로시듀어 세트 안에서 CALLPRC 명령의 순서를 호출 스택이라 합니다. 예를 들어, 다음을 보십시오.



RBAGN540-0

호출 스택:



RBAFN530-0

PROGC가 처리를 종료하면 PROGB 안에서 PROGC를 호출한 명령 바로 다음의 명령으로 제어가 리턴됩니다. 즉, 제어는 호출 스택에서 위쪽 방향으로 리턴됩니다. 이것은 PROGC가 RETURN 또는 ENDPGM 명령으로 종료되는지의 여부에 관계 없이 발생합니다.

CL 프로시듀어는 그 자신을 호출할 수 있습니다.

RETURN 명령

CL 프로시듀어나 OPM 프로그램의 RETURN(리턴) 명령은 해당 프로시듀어나 OPM 프로그램을 호출 스택에서 제거합니다.

RETURN 명령이 들어 있는 프로시듀어를 CALLPRC 명령이 호출한 경우 호출 프로그램의 해당 CALLPRC 명령 바로 다음 명령문으로 제어가 리턴됩니다.

MONMSG 명령에 RETURN 명령으로 종료되는 조치가 지정되면 MONMSG 명령이 들어 있는 프로시듀어나 프로그램을 호출한 명령문 바로 다음 명령문으로 제어가 리턴됩니다.

RETURN 명령에는 매개변수가 없습니다.

주: 초기 프로그램에 RETURN 명령이 있으면 명령 입력 화면이 나옵니다. 보안상의 이유로 이 화면은 그대로 통과할 수도 있습니다.

프로그램과 프로시듀어 간의 매개변수 전달

제어를 다른 프로그램이나 프로시듀어로 전달할 경우 정보를 수정하기 위해 또는 수신 프로그램이나 프로시듀어 안에서 사용하기 위해 정보를 그 프로그램이나 프로시듀어로 전달할 수도 있습니다. 이에 대한 설명은 88 페이지의 『CALL 명령 사용』 부분을 참조하십시오. 전달될 정보는 CALL 또는 PARM 명령의 매개변수에 지정할 수 있습니다. 이 명령의 특성과 요구사항은 약간씩 다릅니다.

예를 들어, PROGA에 다음 명령이 있는 것으로 가정해 보겠습니다.

```
CALL PROGB PARM(&AREA)
```

이 경우 명령이 PROGB를 호출하여 &AREA 값을 전달합니다. PROGB는 PGM 명령으로 시작해야 하며, 이 명령이 수신할 매개변수를 지정해야 합니다.

```
PGM PARM(&AREA) /* PROGB */
```

CALL 명령이나 CALLPRC 명령의 경우 PARM 매개변수에 전달된 매개변수를 지정해야 하고 수신 프로그램 또는 프로시듀어에 있는 PGM 명령의 PARM 매개변수에서 그것을 지정해야 합니다. 매개변수는 이름이 아닌 위치별로 전달되므로 CALL 명령이

나 CALLPRC 명령에서 전달된 값의 위치는 수신 PGM 명령상의 위치와 같아야 합니다. 예를 들어, PROGA에 다음 명령이 있는 것으로 가정해 보겠습니다.

```
CALL PROGB PARM(&A &B &C ABC)
```

이 경우 PROGB는 3개의 변수와 1개의 문자 스트링을 전달합니다. 그런 후 PROGB가 다음과 같이 시작되는 경우를 가정해 보겠습니다.

```
PGM PARM(&C &B &A &D) /*PROGB*/
```

이 경우 PROGA의 &A 값이 PROGB의 &C에 사용되는 식으로 계속 처리되며, PROGB 안의 &D는 ABC가 됩니다. PROGB에 있는 DCL문의 순서는 중요하지 않습니다. 매개변수가 PGM문에 지정된 순서에 의해서만 전달될 변수가 결정됩니다.

매개변수의 위치 이외에도 매개변수의 길이와 유형에 대해서도 유의해야 합니다. 수신 프로시듀어나 프로그램에 나열된 매개변수는 호출 프로그램의 매개변수와 동일한 길이 및 유형으로 선언(declare)되어야 합니다. 십진 상수는 항상 길이(15 5)로 전달됩니다.

CALLPRC 명령을 사용하여 문자 스트링 상수를 전달하는 경우 정확한 바이트 수를 지정하여 정확히 해당 숫자를 전달해야 합니다. 호출된 프로시듀어는 연산 설명자(operational descriptor)에 있는 정보를 사용하여 전달된 바이트 수를 정확히 판별할 수 있습니다. 연산 설명자의 액세스에는 API CEEDOD를 사용할 수 있습니다. API CEEDOD에 대한 정보는 **iSeries Information Center**의 *프로그래밍* 범주에 있는 *API* 섹션을 참조하십시오.

CALL 명령을 사용할 때 32바이트 이하의 문자 스트링 상수는 항상 32바이트 길이로 전달됩니다. 문자 스트링이 32바이트보다 길면, 정확한 바이트 수를 지정하여 이 수만큼 전달되도록 해야 합니다.

다음은 값 &VAR1을 수신하는 프로시듀어나 프로그램의 예입니다.

```
PGM PARM(&VAR1)  /*PGMA*/
DCL VAR1 *CHAR LEN(36)
.
.
.
    ENDPGM
```

CALL 명령이나 CALLPRC 명령은 36자로 지정해야 합니다.

```
CALLPRC PGMA(ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJ)
```

다음의 예는 디폴트 길이를 지정한 것입니다.

```
PGM PARM(&P1 &P2)
DCL VAR(&P1) TYPE(*CHAR) LEN(32)
DCL VAR(&P2) TYPE(*DEC) LEN(15 5)
IF (&P1 *EQ DATA) THEN(CALL MYPROG &P2)
    ENDPGM
```

이 프로그램을 호출하려는 경우 다음과 같이 지정할 수 있습니다.

```
CALL PROG (DATA 136)
```

문자 스트링 DATA는 &P1로 전달됩니다. 10진값 136은 &P2로 전달됩니다.

로컬로 정의된 변수를 참조하는 것이 전달된 변수를 참조하는 것보다 부수적인 작업이 적습니다. 따라서 호출된 프로시듀어나 프로그램이 전달된 변수를 자주 참조하는 경우 전달된 값을 로컬 변수로 복사하여 전달된 값보다는 로컬로 정의된 값을 참조함으로써 성능을 향상시킬 수 있습니다.

OPM CL 프로그램을 호출할 때 이 프로그램으로 전달되는 매개변수의 수는 이 프로그램이 예상하는 수와 정확히 일치해야 합니다. 예상되는 수는 프로그램이 작성될 때 결정됩니다. (오퍼레이팅 시스템은 프로그램이 예상하는 매개변수의 수보다 더 많거나 더 적은 수로 프로그램을 호출하지 못하도록 합니다.) ILE 프로그램이나 프로시듀어를 호출할 때 오퍼레이팅 시스템은 호출시에 전달되는 매개변수의 수를 검사하지 않습니다. 그리고, 오퍼레이팅 시스템이 매개변수를 저장하는 공간은 프로시듀어 호출 사이에서 다시 초기화되지 않습니다. ″n″ 매개변수를 예상하는 프로시듀어를 ″n-1″ 매개변수로 호출하는 경우 매개변수 공간에 어떤 값이 있든지 그것을 사용하여 ″nth″ 매개변수에 액세스합니다. 이 조치의 결과는 거의 예측이 불가능합니다. 이는 CL 프로시듀어를 호출하거나 CL 프로시듀어에 의해 호출되는 다른 ILE 언어로 작성된 프로시듀어에도 적용됩니다.

프로그래머가 가변 길이 매개변수 리스트를 가진 프로시듀어를 작성할 수 있다는 것은 또한 ILE CL 프로시듀어를 작성할 때 더 큰 융통성을 줍니다. 예를 들면, 나중에 매개변수 리스트에 지정되는 매개변수의 경우 한 매개변수의 값에 기초하고 있으므로 필요하지 않을 수도 있습니다. 제어 매개변수가 옵션인 미지정 매개변수를 지정한 경우 호출된 프로시듀어가 그와 같은 옵션 매개변수를 참조하려고 해서는 안됩니다.

매개변수 리스트에서 생략하고자 하는 매개변수에는 특수한 값 *OMIT를 지정할 수도 있습니다. 매개변수에 *OMIT를 지정하면 호출 프로시듀어가 널(null) 포인터를 전달합니다. 호출된 프로시듀어가 생략되는 매개변수를 참조하는 경우 널 포인터를 처리할 준비가 되어 있어야 합니다. 제어 언어(CL)에서는 생략이 가능한 매개변수를 처음 참조할 때 MCH3601에 대해 모니터함으로써 널 포인터를 검사할 수 있습니다. 프로시듀어가 MCH3601을 수신하는 경우 적절한 조치를 수행해야 합니다.

| 프로시듀어를 호출할 때 참조별 및 값별로 인수를 전달할 수 있습니다. 값별로 전달할
| 때 필요한 정보는 ILE 개념 책의 프로시듀어 및 프로그램 호출 장을 참조하십시
| 오.

다음 예에는 두 개의 CL 프로시듀어가 있습니다. 첫 번째 프로시듀어는 하나의 매개변수를 예상하고 있으나, 그 매개변수가 지정되지 않은 채 남아 있는 경우 예측할 수 없는 결과가 나올 것입니다. 첫 번째 프로시듀어는 또 다른 프로시듀어 PROC1을 호출합니다. PROC1은 하나 이상의 매개변수를 예상합니다. 첫 번째 매개변수의 값이 ‘1’

이면 두 번째 매개변수가 지정될 것으로 예상합니다. 두 번째 매개변수의 값이 '0'이면 두 번째 매개변수가 지정되지 않은 채, 그 대신 디폴트 값이 사용될 것으로 가정합니다. PROC1은 또한 CEEDOD API를 사용하여 두 번째 매개변수에 전달되는 실제 길이를 판별합니다.

```
MAIN: PGM   PARM(&TEXT)/* &TEXT must be specified.  Results will be +
                unpredictable if it is omitted.*/
      DCL   VAR(&TEXT) TYPE(*CHAR) LEN(10)
      CALLPRC   PRC(PROC1) PARM('0')
      CALLPRC   PRC(PROC1) PARM('1' &TEXT)
      CALLPRC   PRC(PROC1) PARM('1' 'Goodbye')
ENDPGM

PROC1: PGM   PARM(&P1 &P2) /* PROC1 - Procedure with optional +
           parameter &P2 */
     DCL  VAR(&P1) TYPE(*LGL) /*Flag which indicates +
         whether or not &P2 will be specified. If +
         value is '1', then &P2 is specified */
     DCL   VAR(&P2) TYPE(*CHAR) LEN(10)
     DCL   VAR(&MSG) TYPE(*CHAR) LEN(10)
     DCL   VAR(&PARMPOS) TYPE(*CHAR) LEN(4) /* +
          Parameter position for CEEDOD*/
     DCL   VAR(&PARMDESC) TYPE(*CHAR) LEN(4) /* +
          Parameter description for CEEDOD*/
     DCL   VAR(&PARMTYPE) TYPE(*CHAR) LEN(4) /* +
          Parameter datatype from CEEDOD*/
     DCL   VAR(&PARMINFO1) TYPE(*CHAR) LEN(4) /* +
          Parameter information from CEEDOD */
     DCL   VAR(&PARMINFO2) TYPE(*CHAR) LEN(4) /* +
          Parameter information from CEEDOD */
     DCL   VAR(&PARMLEN) TYPE(*CHAR) LEN(4) /* +
          Parameter length from CEEDOD*/
     DCL   VAR(&PARMLEND) TYPE(*DEC) LEN(3 0) /* +
          Decimal form of parameter length*/
     IF   COND(&P1) THEN(DO) /* Parm 2 is+
          specified, so use the parm value for the +
          message text*/
     CHGVAR  VAR(%BIN(&PARMPOS 1 4)) VALUE(2) /* Tell +
          CEEDOD that we want the operational +
          descriptor for the second parameter*/
     CALLPRC  PRC(CEEDOD) PARM(&PARMPOS &PARMDESC +
          &PARMTYPE &PARMINFO1 &PARMINFO2 &PARMLEN) +
          /* Call CEEDOD to get the length of data +
          specified for &P2*/
      CHGVAR  VAR(&PARMLEND) VALUE(%BIN(&PARMLEN 1 4)) /* +
              Convert the length returned by CEEDOD to +
              decimal format*/
      CHGVAR  VAR(&MSG) VALUE(%SST(&P2 1 &PARMLEND)) /* +
            Copy the data passed in to a local variable*/
      ENDO
      ELSE   CMD(CHGVAR VAR(%MSG) VALUE('Hello')) /* Use +
             "Hello" for the message text*/
      SNDPGMMSG MSG(&MSG)
ENDPGM
```

CALL 명령 사용

CALL 명령이 CL 프로시듀어에 의해 발행될 때 호출된 프로그램에 전달되는 각 매개변수 값은 문자 스트링 상수, 숫자 상수, 논리 상수 또는 CL 변수일 수 있습니다. 호출된 프로그램으로 최대 255개의 매개변수를 전달할 수 있습니다. 매개변수 값은 CALL 명령에 나오는 순서대로 전달되며, 이 순서는 호출된 프로그램의 매개변수 리스트의 순서와 일치해야 합니다. 전달된 변수명이 수신 매개변수 리스트의 이름과 동일할 필요는 없습니다. 호출된 프로그램에서 값을 수신하는 변수명이 호출된 프로그램에 대해 선언되어야 하며, 선언 명령의 순서는 관계 없습니다.

호출된 프로그램과 이 프로그램이 수신하는 변수의 기억장치 사이에는 상관 관계가 없습니다. 대신에 호출 프로그램이 변수를 전달할 때 변수의 기억치장치는 그것이 원래 선언되었던 프로그램 안에 있습니다. 시스템은 주소를 사용하여 변수를 전달합니다. 상수를 전달할 때 호출 프로그램은 상수를 복사하여 그 사본의 주소를 호출된 프로그램으로 전달합니다.

변수가 전달된 결과 호출된 프로그램은 변수 값을 변경시킬 수 있고, 그 변경이 호출 프로그램에 반영됩니다. 나중에 사용하기 위해 호출 프로그램에 새로운 값을 리턴시킬 필요는 없습니다. 따라서 호출 프로그램에 리턴될 변수에 대해 특별히 코딩할 필요는 없습니다. 상수가 전달되고 그 상수값이 호출된 프로그램에 의해 변경될 때 호출 프로그램으로 변경된 값이 알려지지는 않습니다. 따라서 호출 프로그램이 다시 한번 동일한 프로그램을 호출할 경우 호출 프로그램은 변수가 아닌 상수값을 다시 초기화합니다.

이전 설명에 대한 예외는 CALL 명령이 ILE C 프로그램을 호출할 때입니다. CALL 명령을 사용하여 ILE C 프로그램을 호출하고 문자 또는 논리 상수를 전달할 때 시스템은 마지막 비공백 문자 뒤에 널 문자(x'00')를 추가합니다. 상수가 어포스트로피나 16진 상수로 묶인 문자 스트링이면, 지정된 마지막 문자 뒤에 널 문자가 추가됩니다. 이 경우에 후미 공백은 보존됩니다(x'40' 문자). 숫자값은 널 종료 값이 아닙니다.

컴파일되지 않은 CALL 명령(대화식 CALL 명령 또는 SBMJOB 명령을 통해)을 사용하여 CL 프로그램을 호출하면 10진 매개변수(*DEC)는 `LEN(15 5)`으로 선언되어야 하고, 문자 매개변수(*CHAR)는 수신 프로그램에서 `LEN(32)`이나 그 이하로 선언되어야 합니다.

CL 프로시듀어나 프로그램에 있지 않은 CALL 명령은 변수를 인수로 전달할 수 없습니다. CALL 명령이 TYPE(*CMDSTR)으로 정의된 명령 매개변수로 CALL 명령을 지정할 때에는 주의하십시오. 이로 인해 PARM 매개변수에 지정된 변수의 내용이 상수로 변환됩니다. SBMJOB(작업 제출) 명령, ADDJOBSCDE(작업 스케줄 항목 추가) 명령 또는 CHGJOBSCDE(작업 스케줄 항목 변경) 명령의 CMD(명령) 매개변수가 그 예입니다. CALL(호출) 명령의 온라인 도움말 정보에 CALL 명령을 대화식으로 사용할

때 매개변수를 전달하는 방법이 설명되어 있습니다. 명령 도움말을 참조하거나 **iSeries Information Center**의 **프로그래밍** 범주, *CL* 섹션에 있는 명령 문서를 참조하십시오.

매개변수는 다음과 같이 전달되고 수신될 수 있습니다.

- 32바이트 이하의 문자 스트링 상수는 *항상* 32바이트 길이(오른쪽이 공백으로 채워짐)로 전달됩니다. 문자 상수가 32바이트보다 길면 상수의 전체 길이가 전달됩니다. 매개변수가 32바이트보다 길게 정의된 경우 CALL 명령은 정확히 그 바이트 수가 들어 있는 상수를 전달해야 합니다. 32자보다 긴 상수는 수신 프로그램이 예상한 길이까지 채워지지 ***않습니다***.

  수신 프로그램은 전달된 바이트 수보다 적게 수신할 수 있습니다. 예를 들어, 프로그램이 4자를 수신하도록 지정되어 있는데 ABCDEF가 전달되면(나머지 26자는 공백으로 채워짐), 프로그램은 ABCD만 수신하여 사용합니다.

  수신 프로그램이 전달된 바이트 수보다 많이 수신할 경우 예기치 못한 결과가 발생할 수 있습니다. 문자로 전달되는 숫자값은 어포스트로피로 묶어야 합니다.
- 10진 상수는 팩 형식의 길이 LEN(15 5)으로 전달되는데 여기에서 값의 총 길이는 15자리이며, 그 중 5자리는 소수 자릿수입니다. 따라서 12345라는 매개변수가 전달되면 수신 프로그램이 LEN(15 5)의 길이를 가진 10진 필드를 선언해야 하며, 매개변수는 12345.00000으로 수신됩니다.

  숫자 상수를 프로그램에 전달할 때 프로그램에서 15 5 이외의 길이 및 정밀도(precision)의 값을 원하면 그 상수는 16진 형식으로 코드화될 수 있습니다. 다음의 CALL 명령은 LEN(5 2)으로 선언된 값 25.5가 프로그램 변수에 전달되는 방법을 나타내고 있습니다.

  ```
  CALL PGMA PARM(X'02550F')
  ```
- 논리 상수는 32바이트 길이로 전달됩니다. 첫 번째 바이트가 논리값 0 또는 1이고, 나머지 바이트는 공백입니다. 논리값을 예상하고 있는 프로그램에 0 또는 1 이외의 값이 전달되면 예기치 못한 결과가 발생할 수 있습니다.
- 부동 소수점 리터럴이나 부동 소수점 특수 값(*NAN, *INF 또는 *NEGINF)은 배정밀도의 8바이트 값으로 전달됩니다. CL 프로그램이 부동 소수점 숫자를 처리할 수는 없지만, 부동 소수점값을 문자 변수에 받아들여 부동 소수점값을 처리할 수 있는 고급 언어(HLL) 프로그램으로 이 변수를 전달할 수 있습니다.
- 호출이 CL 프로시듀어나 프로그램에서 이루어진 경우에는 시스템이 변수를 전달할 수 있습니다. 이 경우 수신 프로그램은 호출 CL 프로시듀어나 프로그램에 정의된 변수와 일치하는 필드를 선언해야 합니다. 예를 들어, CL 프로시듀어나 프로그램이 &CHKNUM이라는 이름의 10진 변수를 LEN(5 0)으로 정의하는 경우 수신 프로그램은 필드를 소수 자릿수가 없는 총 5자리의 팩 형식으로 선언해야 합니다. CALL 명령이 CL 프로시듀어나 프로그램에서 SBMJOB 명령을 사용하여 일괄처리 모드로 실행되는 경우 시스템이 인수로서 전달된 변수를 상수로 취급합니다.

- 10진 상수나 프로그램 변수가 호출된 프로그램에 전달될 수 있으면, 매개변수는 LEN(15 5)으로 정의되어야 하고, 모든 호출 프로그램은 이 정의에 따라야만 합니다. 매개변수의 유형, 갯수, 순서 및 길이가 호출 프로그램과 수신 프로그램 간에 일치하지 않으면(이전에 설명한 문자 상수 길이에 대한 예외사항은 제외), 결과를 예측할 수 없습니다.
- 정수 상수를 정수 변수에 전달하려면 해당 상수를 16진 형식으로 지정해야 합니다. 16진 상수와 정수 변수의 *크기*는 같아야 합니다.
- 널값은 다른 프로그램에 전달될 수 없으므로 값 *N은 널값을 지정하는 데 사용할 수 없습니다.

다음 예에서 프로그램 A는 6개의 매개변수(하나의 논리 상수, 세 개의 변수, 하나의 문자 상수 및 하나의 숫자 상수)를 전달합니다.

```
PGM /* PROGRAM A */
DCL VAR(&B) TYPE(*CHAR)
DCL VAR(&C) TYPE(*DEC) LEN(15 5) VALUE(13.529)
DCL VAR(&D) TYPE(*CHAR) VALUE('1234.56')
CHGVAR VAR(&B) VALUE(ABCDEF)
CALL PGM(B) PARM('1' &B &C &D XYZ 2)  /* Note blanks between parms */
.
.
.
    ENDPGM

PGM PARM(&A &B &C &W &V &U) /* PROGRAM B */
DCL VAR(&A) TYPE(*LGL)
DCL VAR(&B) TYPE(*CHAR) LEN(4)
DCL VAR(&C) TYPE(*DEC)
            /* Default length (15 5) matches DCL LEN in program A */
DCL VAR(&W) TYPE(*CHAR)
DCL VAR(&V) TYPE(*CHAR)
DCL VAR(&U) TYPE(*DEC)
.
.
. ENDPGM
```

주: PGMB로 전달된 5번째 매개변수가 XYZ 대신 456이었고 영숫자 자료로 의도된 것이었다면, 값이 매개변수에서 '456'으로 지정되었을 것입니다.

논리 상수 '1'은 호출 프로그램에 선언될 필요가 없습니다. 그것은 논리 유형으로 선언되며, 프로그램 B에서 &A라는 이름을 가집니다.

DCL 명령에서 &B의 길이를 지정하지 않았으므로 디폴트 길이인 32자가 전달됩니다. 단지 &B의 6자만 지정됩니다(ABCDEF). 프로그램 B에서는 &B가 4자로 선언되었으므로 4자만 수신됩니다. 프로그램 B에서 이 값이 변경되면 &B의 4자리는 호출 이후의 나머지 부분에 대해서도 프로그램 A에서 변경됩니다.

프로그램 A에서 &C에는 길이(LEN) 매개변수를 지정해야 합니다. 길이 매개변수를 지정하지 않으면 지정된 값의 길이를 디폴트 길이로 사용하게 되지만, 이 길이는 프로그램 B에서 예상한 디폴트 길이와는 호환되지 않습니다. &C는 값 13.52900을 가집니다.

프로그램 B의 &W(프로그램 A의 &D)는 문자로 선언되었기 때문에 문자로 수신됩니다. TYPE이 *CHAR이면, 스트링을 표시하기 위하여 어포스트로피를 사용하지 않아도 됩니다. 프로그램 A에서 디폴트 값의 길이는 7입니다. (소수점은 문자 스트링에서 한 자리로 간주됩니다.) 프로그램 B는 길이를 32로 예상하고 있습니다. 처음의 7자는 전달되나 나머지 부분의 문자는 예측할 수 없습니다.

변수 &V는 오른쪽이 공백으로 채워지는 문자 스트링 XYZ입니다. 변수 &U는 숫자 자료인 2.00000입니다.

DCL 명령의 디폴트 길이에 대한 정보는 명령 도움말을 참조하거나 **iSeries Information Center**의 **프로그래밍** 범주에 있는 *CL* 섹션을 참조하십시오.

프로그램 및 프로시듀어 호출 시 공통 오류

다음 섹션에서는 CALL 명령이나 CALLPRC 명령에 값을 전달할 때 가장 자주 만나는 오류에 대해 설명합니다. 이 오류 중 일부는 디버그 처리가 매우 어렵고 일부는 프로그램 기능에 심각한 결과를 가져옵니다.

CALL 명령을 사용한 자료 유형 오류

명령 스트링의 총 길이에는 사용된 명령어, 공간, 매개변수명, 괄호, 변수의 내용 및 어포스트로피가 포함됩니다. 대부분의 명령에서 명령 스트링은 예상한 대로 명령 처리 프로그램을 시작합니다. 그러나 일부 명령에서는 예상한 대로 변수가 전달되지 않을 수도 있습니다. 변수 관련 항목에 대해 자세히 알려면 25 페이지의 『변수에 대한 작업』 부분을 참조하십시오.

CALL 명령이 SBMJOB 명령의 CMD 매개변수와 함께 사용되면 예기치 못한 결과가 발생할 수 있습니다. 문법적으로, CALL 명령이 CMD 매개변수와 함께 사용되면 CMD 매개변수가 CALL 명령에 대한 컴파일러 지시문으로 사용될 때처럼 동일한 현상이 나타납니다. CMD 매개변수와 함께 사용되면 CALL 명령은 나중에 일괄처리 서브시스템이 이를 시작할 때 수행될 수 있도록 명령 스트링으로 변환됩니다. CALL 명령이 그 자신에 의해 사용되면 CL 컴파일러는 코드를 생성하여 호출을 수행합니다.

때로는 10진 상수와 문자 변수에 공통되는 문제가 발생하기도 합니다. 다음의 경우에는 명령 스트링이 요구대로 구성되지 않습니다.

- 10진수가 10진 상수로 변환되는 경우.

  명령 스트링이 실행될 때 10진 상수는 길이 LEN(15 5)의 팩 양식으로 전달됩니다. 이는 CL 변수에서 지정한 양식대로 전달되지 않은 것입니다.

- 문자 변수의 길이가 32자보다 길게 선언된 경우.

문자 변수의 내용은 앞에서 설명한 바와 같이, 일반적으로 후미 공백이 제거되고 인용부호로 묶인 문자 상수로서 전달됩니다. 결과적으로 호출된 프로그램은 자료를 충분히 전달받을 수 없습니다.

다음의 방법은 명령 스트링의 구성 시 발생하는 오류를 정정하는 데 사용될 수 있는 것입니다.

- 여러 부분의 명령을 하나의 CL 변수로 연결하여 제출될 CALL 명령 스트링을 작성합니다. SBMJOB 명령의 RQSDTA(자료 요구) 매개변수를 사용하여 명령 스트링을 제출합니다.
- 후미 공백이 유효하고 32자보다 긴 CL 문자 변수의 경우 필요한 문자 수보다 한 자 더 길게 변수를 작성하여 마지막 위치에 비공백 문자를 서브스트링으로 추가 작성합니다. 이것은 유효한 공백이 절단되는 것을 막습니다. 호출된 프로그램은 그 문자가 예상 길이를 초과했으므로 여분의 문자를 무시합니다.
- 호출될 프로그램을 시작할 명령을 작성합니다. CALL 명령을 사용하는 대신에 새로운 명령을 제출합니다. 명령 정의는 매개변수가 예상된 명령 처리 프로그램으로 전달되는지 확인합니다.

자료 유형 오류

값을 전달할 때 자료 유형(TYPE 매개변수)은 호출 프로시듀어나 프로그램과 호출된 프로시듀어나 프로그램에서 동일(*CHAR, *DEC 또는 *LGL)해야 합니다. 숫자 상수를 전달하려고 시도할 때 오류가 자주 발생합니다. 숫자 상수가 어포스트로피로 묶여 있으면 문자 스트링으로서 전달됩니다. 그러나 상수가 어포스트로피로 묶여 있지 않으면 LEN(15 5)으로 지정된 팩 숫자 필드로 전달됩니다.

다음 예에서는 인용부호로 묶은 숫자값이 10진값을 예상하는 프로그램에 전달됩니다. 10진 자료 오류(이탈 메세지 MCH1202)는 아래와 같이 호출된 프로그램(PGMA)에서 변수 &A를 참조할 때 발생합니다.

```
CALL   PGMA PARM('123')  /* CALLING PROGRAM */
PGM PARM(&A)            /* PGMA */
DCL    &A *DEC LEN(15 5)  /* DEFAULT LENGTH */
.
.
.
IF (&A *GT 0) THEN(...) /* MCH1202 OCCURS HERE */
```

다음 예에서는 10진값이 문자 변수를 정의하는 프로그램에 전달됩니다. 일반적으로 이 오류 때문에 수행 시 실패가 발생하지는 않으나 그 결과가 틀릴 수 있습니다.

```
CALL PGMB PARM(12345678)  /* CALLING PROG */

PGM PARM(&A)     /* PGMB */
DCL &A *CHAR 8
```

```
.
.
.
    ENDPGM
```

PGMB의 변수 &A는 16진값 001234567800000F를 가집니다.

일반적으로 값이 '0' 또는 '1'로 표현되면 자료는 오류 없이 논리 변수(*LGL)에서 문자 변수(*CHAR)로 또는 그 반대로 전달됩니다.

10진 길이와 정밀도 오류

10진값이 틀린 소수부 길이와 정밀도를 가진 채 전달되면(너무 길거나 너무 짧으면), 변수가 참조될 때 10진 자료 오류(MCH1202)가 발생합니다. 다음 예에서는 숫자 상수가 LEN(15 5)으로 전달되지만, 호출된 프로시듀어나 프로그램에서는 LEN(5 2)으로 선언되어 있습니다. 숫자 상수는 항상 팩 십진수(15 5)로 전달됩니다.

```
CALL PGMA PARM(123)     /* CALLING PROG */

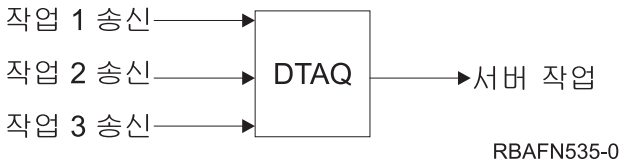
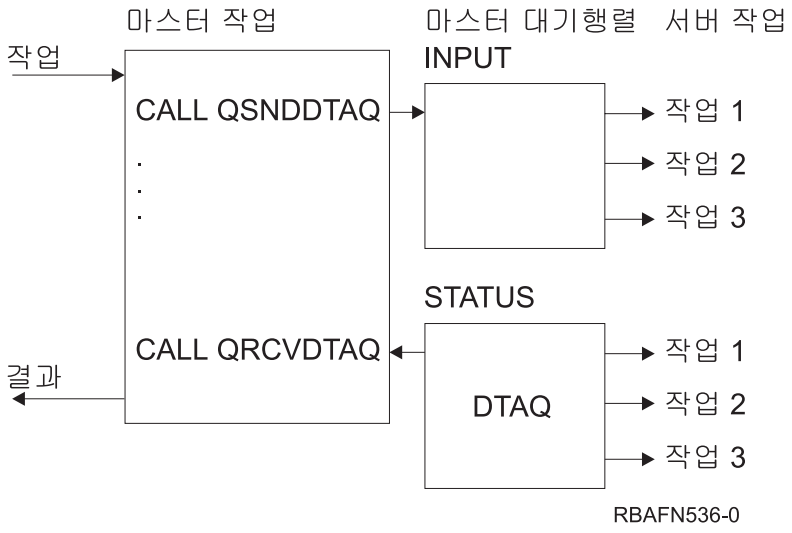
PGM PARM(&A)            /* PGMA */
DCL &A *DEC (5 2)
.
.
.
IF (&A *GT 0) THEN(...) /* MCH1202 OCCURS HERE */
```

10진 변수가 호출 프로그램이나 프로시듀어에서 LEN(5 2)으로 선언되고 값이 상수 대신 변수로 전달되면 오류는 발생하지 않습니다.

숫자 상수를 프로시듀어 또는 프로그램에 전달할 필요가 있고, 프로시듀어나 프로그램이 15 5 이외의 길이와 정밀도를 가진 값을 예상하고 있으면, 상수를 16진 형식으로 코드화할 수 있습니다. 다음의 CALL 명령은 LEN(5 2)으로 선언된 값 25.5가 프로그램 변수에 전달되는 방법을 나타내고 있습니다.

```
CALL PGMA PARM(X'02550F')
```

10진값이 길이는 맞지만 틀린 정밀도(소수 자릿수 숫자)로 전달되는 경우 수신 프로시듀어나 프로그램은 값을 틀리게 해석합니다. 다음 예에서는 프로시듀어로 전달된 숫자 상수값(길이가 (15 5)인)이 25124.00으로 처리됩니다.

```
CALL PGMA PARM(25.124)  /* CALLING PGM */

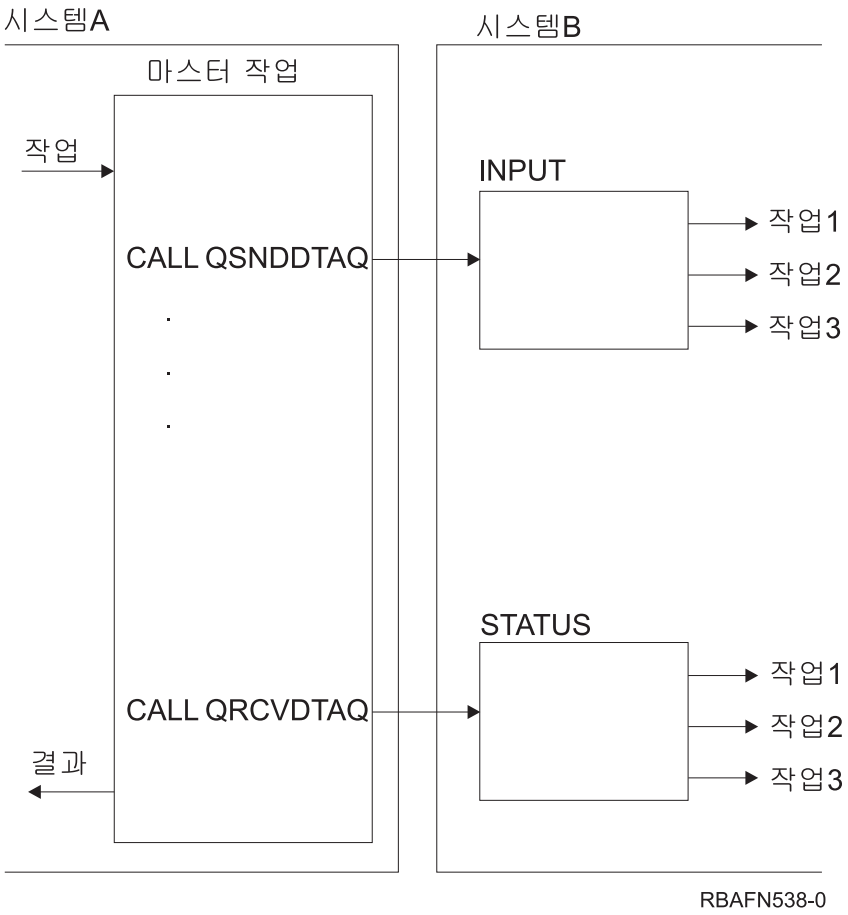
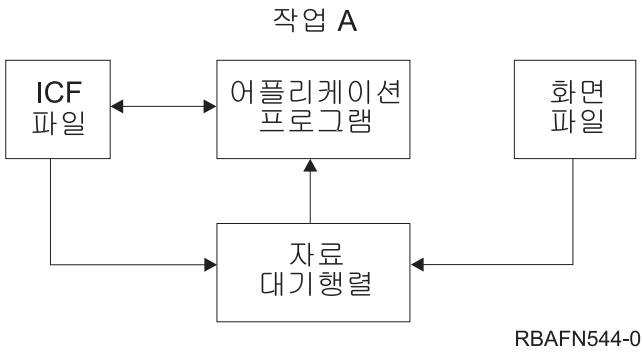
PGM PARM(&A)            /* PGMA */
DCL &A *DEC (15 2)      /* LEN SHOULD BE 15 5*/
.
.
.
    ENDPGM
```

이러한 오류는 변수가 전달되거나 선언될 때가 아니라, 변수가 맨 처음 참조될 때 발생합니다. 다음 예에서는 호출된 프로그램이 변수를 참조하지 않지만, 그 대신 호출 프로

그램으로 리턴되는 변수에 값(발견된 틀린 길이 값)을 넣기만 합니다. 변수가 호출 프로그램에 리턴되어 처음으로 참조될 때까지는 오류가 발견되지 않습니다. 특히 이런 종류의 오류는 발견이 어렵습니다.

```
PGM  /* PGMA */
DCL &A *DEC (7 2)
CALL PGMB PARM(&A)  /* (7 2) PASSED TO PGMB */
IF (&A *NE 0)  THEN(...)  /* *MCH1202 OCCURS HERE */
.
.
.
    ENDPGM

PGM PARM(&A)     /* PGMB */
DCL &A *DEC (5 2)        /* WRONG LENGTH */
.
.
.
 CHGVAR &A (&B-&C)      /* VALUE PLACED in &A */
          RETURN
```

제어가 프로그램 PGMA로 리턴하여 &A를 참조할 때 오류가 발생합니다.

문자 길이 오류

선언된 수신 변수의 문자 길이보다 긴 문자 값을 전달하는 경우 수신 프로시듀어나 프로그램이 초과한 길이에 액세스할 수 없습니다. 다음 예에서 PGMB는 전달되는 변수를 공백으로 변경시킵니다. 변수가 LEN(5)으로 선언되어 있으므로 PGMB에서는 5자만 공백으로 변경되고 나머지 문자는 PGMA에서 참조되었을 때의 그 값으로 계속 남아 있습니다.

```
PGM  /* PGMA */
DCL &A *CHAR 10
CHGVAR &A 'ABCDEFGHIJ'
CALL PGMB PARM(&A)           /* PASS to PGMB */
.
.
.
IF (&A *EQ '  ') THEN(...)  /* THIS TEST FAILS */
ENDPGM

PGM PARM(&A)     /* PGMB */
DCL &A *CHAR 5   /* THIS LEN ERROR*/
CHGVAR &A '  '   /* 5 POSITIONS ONLY; OTHERS UNAFFECTED */
          RETURN
```

이러한 종류의 오류가 이탈 메세지를 발생시키지는 않지만, 이와 같은 방법으로 처리된 변수는 예상과 다른 기능을 할 수도 있습니다.

프로시듀어나 프로그램에 전달된 값이 수신 프로시듀어나 프로그램에서 선언된 길이보다 짧은 경우 매우 심각한 결과가 발생할 수 있습니다. 이러한 경우 호출된 프로시듀어나 프로그램의 변수 값은 원래 전달된 값으로 구성되며, 그 길이는 기억장치에서 해당 값 다음에 무엇이 나오든간에 호출된 프로시듀어나 프로그램에서 선언된 길이가 됩니다. 이렇게 일시적으로 허용된 기억장치의 내용은 원래 예상한 값이 아닐 수 있습니다.

전달된 값이 변수인 경우 프로시듀어나 프로그램에 대해 다른 변수나 내부 제어 구조가 이 값 다음에 따라나올 수 있습니다. 전달된 값이 상수인 경우 CALL 또는 CALLPRC 명령에서 전달된 다른 상수나 내부 제어 구조가 기억장치에서 이 값 다음에 따라나올 수 있습니다.

수신 프로시듀어나 프로그램 값을 변경하는 경우 원래 값과 일시적으로 허용된 기억장치에 영향을 미칩니다. 이로 인한 즉각적인 효과로 인해 다른 변수나 상수가 변경될 수 있고 프로시듀어나 프로그램이 실패하는 것과 같은 식으로 내부 구조가 변경될 수 있습니다. 일시적으로 허용된 기억장치에 대한 변경은 즉시 효력을 나타냅니다.

다음 예에서는 세 자로 된 두 개의 상수가 호출된 프로그램에 전달됩니다. 문자 상수는 CALL 명령에 대해 최소 32자로 전달됩니다. (일반적으로, 값은 후미 공백과 함께 좌측 정렬된 세 자로 전달됩니다.) 수신 프로그램이 수신 변수를 32자리보다 길게 선언하는 경우 초과된 자리에 알 수 없는 값의 일시적으로 허용된 기억장치가 사용됩니다. 다음 예는 두 상수가 기억장치에서 서로 인접해 있는 것으로 가정한 것입니다.

```
CALL PGMA ('ABC' 'DEF')  /* PASSING PROG */

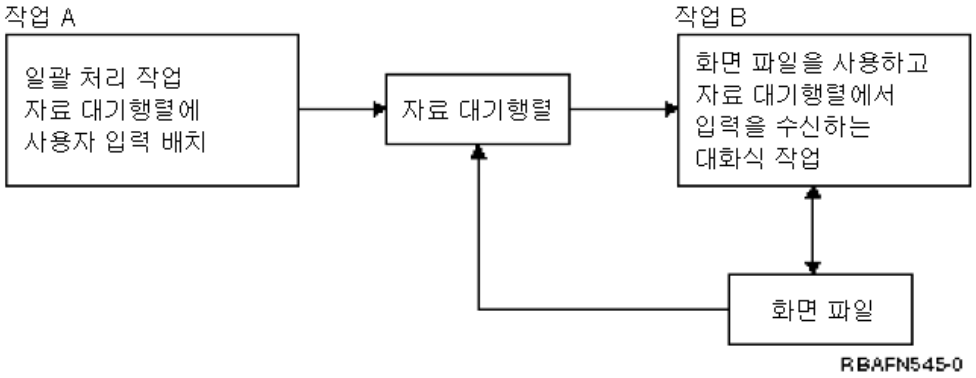
PGM  PARM(&A &B) /* PGMA */
DCL  &A *CHAR 50  /* VALUE:ABC+29' '+DEF+15' ' */
DCL  &B *CHAR 10  /* VALUE:DEF+7' ' */
CHGVAR VAR(&A) (' ')  /* THIS ALSO BLANKS &B */
.
.
.
    ENDPGM
```

변수로 전달된 값은 정확히 같은 방식으로 적용됩니다.

다음 예에서는 세 자로 이루어진 두 개의 상수가 호출된 프로시듀어에 전달됩니다. 지정된 문자 수만 CALLPRC 명령에 대해 전달됩니다. 수신 프로그램이 수신 변수를 전달된 상수 자리보다 길게 선언하면 초과된 자리가 알 수 없는 값의 일시적으로 허용된 기억장치를 사용하게 됩니다.

다음의 예는 두 상수가 기억장치에서 인접해 있다고 가정한 것입니다.

```
CALLPRC PRCA ('ABC' 'DEF')  /* PASSING PROG */

PGM  PARM(&A &B)       /*  *PRCA  */
DCL  &A *CHAR 5        /*   VALUE:'ABC' + 'DE' */
DCL  &B *CHAR 3        /*   VALUE:'DEF' */
CHGVAR &A ' '          /*   This also blanks the first two bytes of &B */
.
.
.
    ENDPGM
```

프로그램과 프로시듀어 간의 통신을 위한 자료 대기행렬 사용

자료 대기행렬은 프로그래머가 작성할 수 있는 시스템 오브젝트 유형으로서, 하나의 HLL 프로시듀어나 프로그램이 이 오브젝트로 자료를 송신할 수 있으며, 또 다른 HLL 프로시듀어나 프로그램이 이 오브젝트로부터 자료를 수신할 수 있습니다. 수신 프로그램은 자료를 이미 대기 중일 수도 있고 나중에 자료를 수신할 수도 있습니다.

자료 대기행렬을 사용하면 다음과 같은 장점이 있습니다.

- 자료 대기행렬을 사용하여 조작을 수행 중이던 작업을 해제시킬 수 있습니다. 작업이 대화식 작업이면 더 빠른 응답 시간을 제공받을 수 있으며, 대화식 프로그램과 그 프로세스 액세스 그룹(PAG)의 크기를 줄일 수 있습니다. 결과적으로, 시스템의 전반적인 성능을 향상시킬 수 있습니다. 예를 들면, 여러 명의 워크스테이션 사용자가 여러 개의 파일 갱신 및 추가와 관련된 트랜잭션을 입력할 경우 대화식 작업에서 트랜잭션에 대한 요구를 하나의 일괄처리 작업으로 제출하면 시스템 성능을 향상시킬 수 있습니다.
- 자료 대기행렬은 두 작업 간의 가장 빠른 비동기 통신 방법입니다. 자료 대기행렬을 사용하여 자료를 송수신하면 데이터베이스 파일, 메세지 대기행렬, 자료 영역을 사용하여 자료를 송수신하는 것보다 부수적인 작업이 훨씬 줄어듭니다.
- HLL 프로시듀어 또는 프로그램을 나가지 않거나 설명을 송수신, 지우기, 검색하기 위한 CL 프로시듀어를 호출하지 않고도 QSNDDTAQ, QRCVDTAQ, QMHRDQM, QCLRDTAQ, QMHQRDQD 프로그램을 호출하여 어떤 HLL 프로시듀어나 프로그램에 있는 자료 대기행렬의 설명도 송수신 및 검색이 가능합니다.
- 자료 대기행렬로부터 자료를 수신할 때 시간종료를 설정하여 항목이 자료 대기행렬 상에 도착될 때까지 작업을 대기시킬 수 있습니다. 이는 OVRDBF 명령의 EOFDLY 매개변수를 사용하여 지연 시간이 종료되면 언제든지 작업이 활성화되도록 하는 것과는 다릅니다.
- 둘 이상의 작업이 동일한 자료 대기행렬로부터 자료를 수신할 수 있습니다. 이러한 점은 처리해야 하는 항목의 수가 한 작업이 적정 상태에서 처리할 수 있는 수보다 많은 어플리케이션의 경우에 유리합니다. 예를 들면, 여러 개의 프린터를 사용하는 인쇄 순서를 정할 경우 여러 개의 대화식 작업이 하나의 자료 대기행렬로 요구를 송신할 수 있습니다. 각 프린터에 대한 개별 작업은 선입선출(FIFO)이나 후입선출(LIFO) 또는 키순 대기행렬 순서로 자료 대기행렬로부터 자료를 수신할 수 있습니다.
- 자료 대기행렬은 대기행렬에 위치한 각 메세지에 송신자 ID를 접속할 수 있습니다. 대기행렬이 작성될 때 설정되는 자료 대기행렬의 속성인 송신자 ID에는 규정화 작업명과 현재 사용자 프로파일이 들어 있습니다.

이러한 장점 외에도, 자료 대기행렬을 저널할 수 있습니다. 이 기능은 비정상적인 초기 프로그램 로드(IPL)나 장애가 발생했을 때 오브젝트가 일부 변경 조치 중에 있더라도 오브젝트를 일관적인 상태로 회복시킬 수 있도록 합니다. 저널링은 리모트 시스템에 자료 대기행렬 저널의 복제를 제공하기도 합니다(예를 들면, 리모트 저널을 사용하여). 이

는 시스템이 어플리케이션 작업을 복제하기 위해 유사 환경에서 조치를 재생성하게 합니다. iSeries 서버에서의 저널링 지원에 대한 자세한 정보는 Information Center의 저널 관리 정보를 참조하십시오.

다음은 자료 대기행렬이 어떻게 작업하는지를 보여주는 예입니다. 여러 작업이 항목을 자료 대기행렬에 넣습니다. 항목은 서버 작업에 의해 처리됩니다. 이것은 처리된 명령을 작업이 인쇄를 실행하는 하나의 작업으로 송신할 때 사용될 수 있습니다. 작업의 수와는 관계 없이 동일한 대기행렬로 송신할 수 있습니다.



자료 대기행렬을 사용하는 다른 예는 다음과 같습니다. 1차 작업이 작업 요구를 받아서, 자료 대기행렬로 그 항목을 송신합니다(QSNDDTAQ 프로그램을 호출하여). 서버 작업은 자료 대기행렬로부터 항목을 수신하여(QRCVDTAQ 프로그램을 호출하여) 자료를 처리합니다. 서버 작업은 다른 자료 대기행렬을 사용하여 1차 작업에 상태를 다시 보고할 수 있습니다.

자료 대기행렬을 사용하면 1차 작업이 서버 작업으로 작업을 라우트하도록 할 수 있습니다. 이렇게 하면 1차 작업이 해제되어 다음 작업 요구를 수신할 수 있습니다. 서버 작업의 수와는 관계 없이 동일한 자료 대기행렬로부터 여러 서버 작업을 수신할 수 있습니다.



자료 대기행렬에 항목이 없으면, 서버 작업에 다음 옵션이 사용됩니다.

- 대기행렬에 항목이 놓일 때까지 대기합니다.
- 일정 시간 동안 대기합니다. 항목이 여전히 도착하지 않으면 처리를 계속합니다.
- 기다리지 않고 즉시 리턴합니다.

자료 대기행렬은 또한 프로그램이 동시에 화면 파일, ICF 파일 및 자료 대기행렬로부터의 입력을 위해 대기해야 할 때 사용됩니다. DTAQ 매개변수를 다음 명령에 대해 지정하는 것으로 가정해보십시오.

- CRTDSPF(화면 파일 작성) 명령
- CHGDSPF(화면 파일 변경) 명령
- OVRDSPF(화면 파일 대체) 명령
- CRTICFF(ICF 파일 작성) 명령
- CHGICFF(ICF 파일 변경) 명령
- OVRICFF(ICF 파일 대체) 명령

이 경우 다음 중 하나가 발생할 때 항목이 놓일 자료 대기행렬을 지정할 수 있습니다.

- 작동 가능한 명령 키나 Enter 키를 초청된 표시장치에서 눌렀을 때
- 자료를 초청된 ICF 세션에서 사용할 수 있을 때

CRTPITQ(출력 대기행렬 작성) 또는 CHGOUTQ(출력 대기행렬 변경) 명령을 사용하면 자료 대기행렬을 출력 대기행렬과 선택적으로 연관시킬 수 있습니다. 스풀 파일이 출력 대기행렬에서 준비완료(RDY) 상태일 때 시스템이 그 항목들을 자료 대기행렬에 기록합니다. 사용자 프로그램은 자료 대기행렬 수신(QRCVDTAQ) API를 사용하여 자료 대기행렬로부터 정보를 수신함으로써 출력 대기행렬에서 스풀 파일을 언제 사용할 수 있는지 알 수 있습니다. CRTOUTQ(출력 대기행렬 작성) 명령에 대한 자세한 정보는 **iSeries Information Center**의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오. 출력 대기행렬의 대기 행렬에 대한 자세한 정보는 Printer Device Programming  책을 참조하십시오.

또한 시스템에서 수행되는 작업은 QSNDDTAQ 프로그램을 사용함으로써 DTAQ 매개변수에 지정된 것과 동일한 자료 대기행렬에 항목을 놓을 수 있습니다.

어플리케이션은 자료 대기행렬에 놓인 각 항목을 수신한 후 화면 파일, ICF 파일 또는 QSNDDTAQ 프로그램이 항목을 대기행렬에 놓는지 여부에 따라 항목을 처리하도록 QRCADATAQ 프로그램을 호출합니다. 더 자세한 내용은 103 페이지의 『예 2: 화면 파일과 ICF 파일로부터의 입력 대기』 부분과 106 페이지의 『예 3: 화면 파일과 자료 대기행렬로부터의 입력 대기』 부분을 참조하십시오.

리모트 자료 대기행렬

사용자는 분산 자료 관리(DDM) 파일을 사용하여 리모트 자료 대기행렬에 액세스할 수 있습니다. DDM 파일은 다음 기능 중 하나를 수행하기 위해 한 서버에서 상주하고 있는 프로그램이 리모트 서버에 있는 자료 대기행렬을 액세스할 수 있도록 해줍니다.

- 자료 대기행렬로 자료를 송신합니다.
- 자료 대기행렬로부터 자료를 수신합니다.

- 자료 대기행렬로부터 자료를 지웁니다.

현재 표준 자료 대기행렬이 사용하는 어플리케이션 프로그램은 어플리케이션을 변경하거나 다시 컴파일하지 않고서도 리모트 DDM 자료 대기행렬에 액세스하도록 할 수 있습니다. 올바른 자료 대기행렬이 액세스되었는지 확인하기 위해서는 다음 중 하나를 수행할 경우도 있습니다.

- 표준 자료 대기행렬을 삭제하고 원래 표준 자료 대기행렬과 동일한 이름을 가진 DDM 자료 대기행렬을 작성합니다.
- 표준 자료 대기행렬의 이름을 변경합니다.

사용자는 다음 명령을 사용하여 DDM 자료 대기행렬을 작성할 수 있습니다.

```
 CRTDTAQ DTAQ(LOCALLIB/DDMDTAQ) TYPE(*DDM)
RMTDTAQ(REMOTELIB/REMOTEDTAQ) RMTLOCNAME(SYSTEMB)
TEXT('DDM data queue to access data queue on SYSTEMB')
```

또한 리모트 자료 대기행렬을 사용하여 DDM 자료 대기행렬을 작성하기 위해서는 이전 예(″Master Job/Server Job″)의 확장 부분을 사용할 수도 있습니다. 마스터 작업은 SystemA에 있으며, 자료 대기행렬과 서버 작업은 SystemB로 이동합니다. 두 개의 DDM 자료 대기행렬(INPUT 및 STATUS)이 작성되고 나면, 마스터 작업이 SystemB에 있는 서버 작업과 계속하여 비동기식으로 통신합니다. 다음 예는 리모트 자료 대기행렬을 사용하여 DDM 자료 대기행렬을 작성하는 방법을 보여줍니다.

```
 CRTDTAQ DTAQ(LOCALLIB/INPUT) TYPE(*DDM)
RMTDTAQ(REMOTELIB/INPUT) RMTLOCNAME(SystemB)
TEXT('DDM data queue to access INPUT on SYSTEMB')

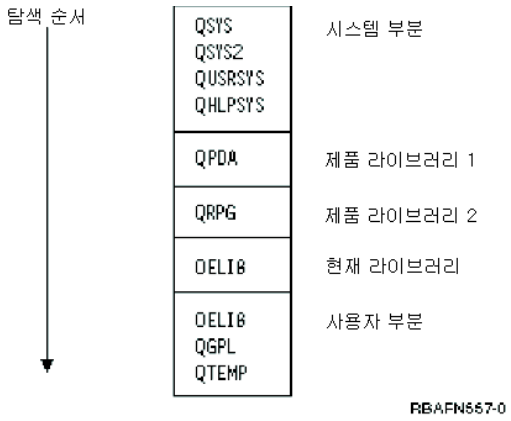
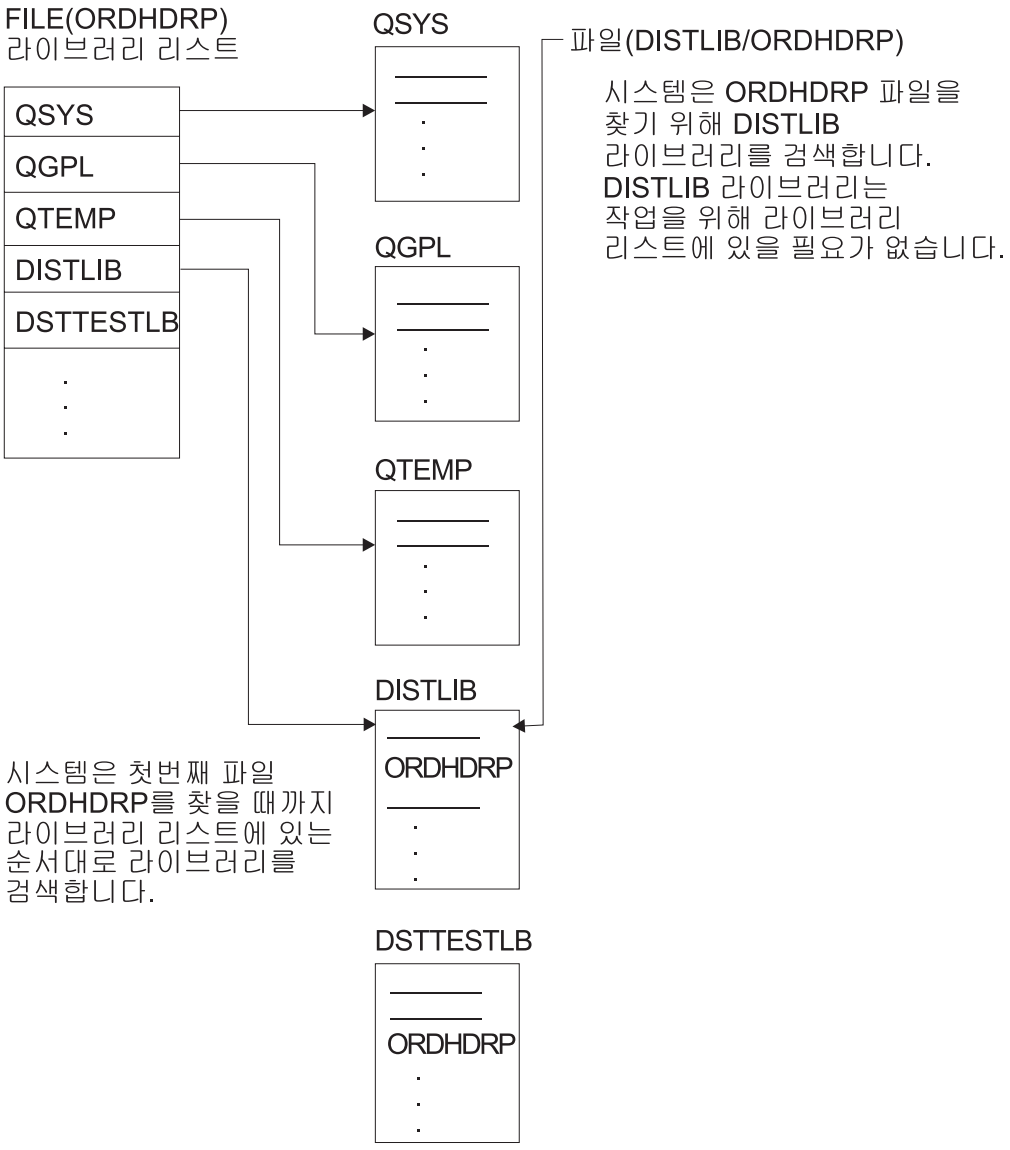
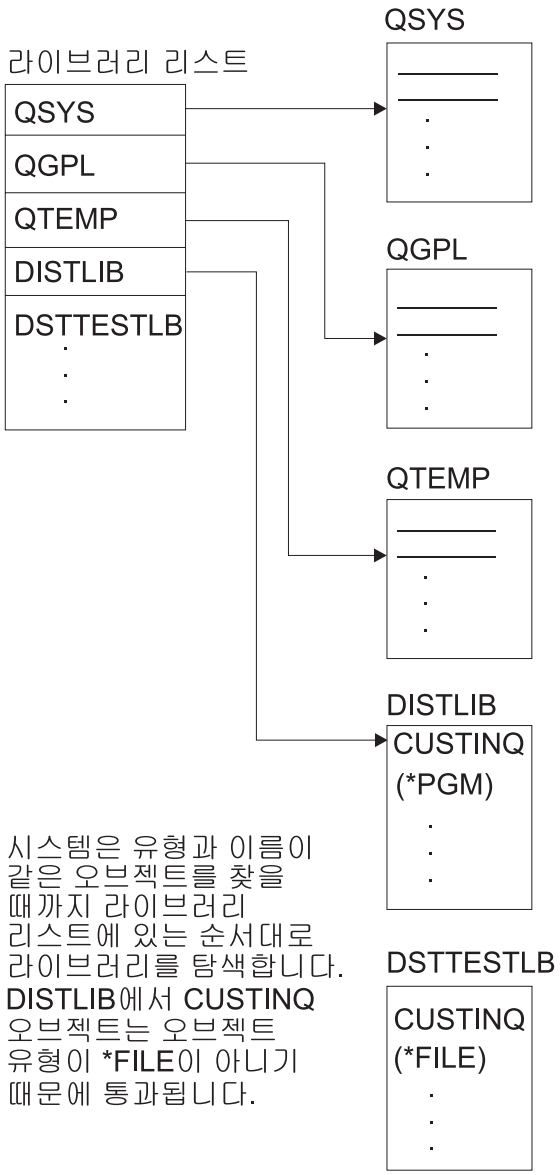
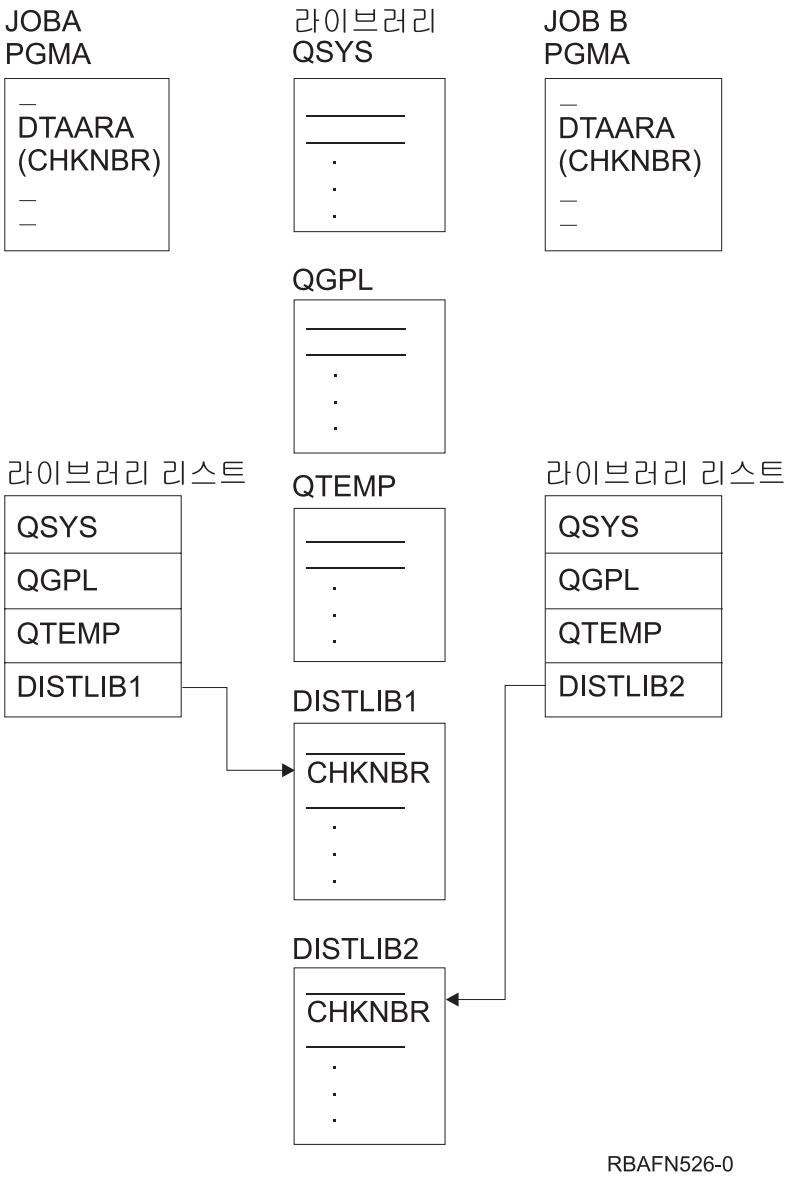
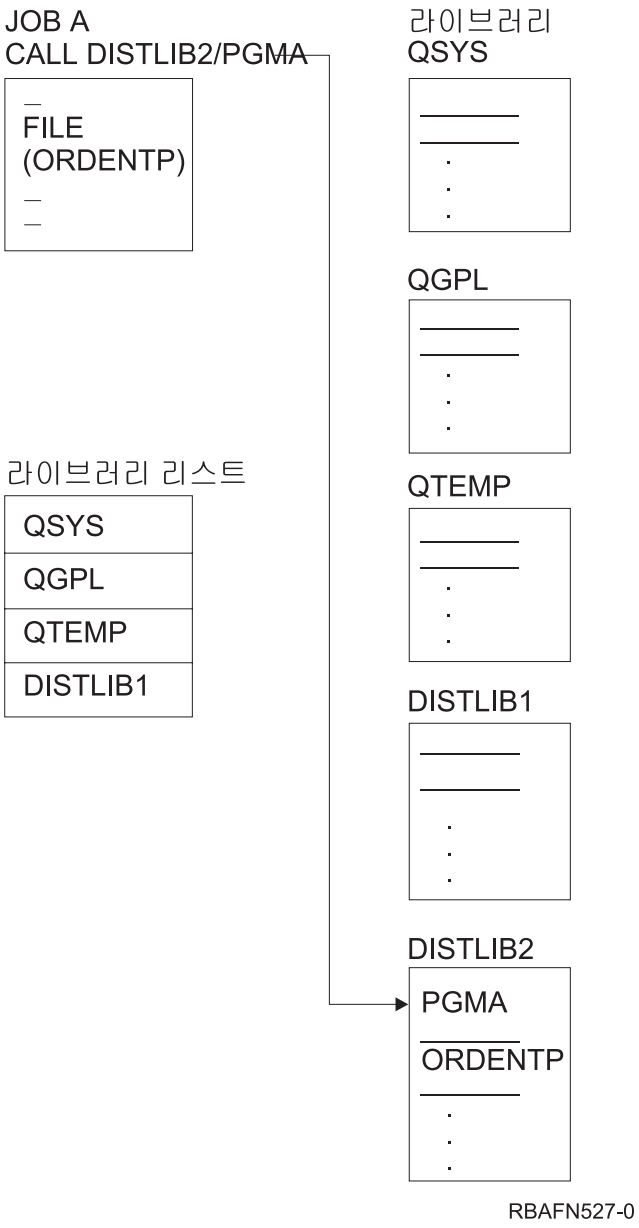
CRTDTAQ DTAQ(LOCALLIB/STATUS) TYPE(*DDM)
   RMTDTAQ(REMOTELIB/STATUS) RMTLOCNAME(SystemB)
   TEXT('DDM data queue to access STATUS on SYSTEMB')
```

QSNDDTAQ 마스터 작업은 LOCALLIB/INPUT이라는 이름의 자료 대기행렬을 전달하고 SystemB에 있는 리모트 자료 대기행렬(REMOTELIB/INPUT)로 자료를 송신합니다. 리모트 자료 대기행렬(REMOTELIB/STATUS)로부터 자료를 수신하기 위해 마스터 작업은 그 호출에 대해 LOCALLIB/STATUS라는 이름의 자료 대기행렬을 QRCVDTAQ로 전달합니다.

DDM 자료 대기행렬에 대한 자세한 정보는 **iSeries Information Center**의 *프로그래밍* 범주에서 *CL* 섹션을 참조하십시오.



그림 1. 리모트 자료 대기행렬 액세스 예

데이터베이스 파일과 대기행렬의 사용 비교

다음은 자료 대기행렬과 데이터베이스 파일을 사용할 때의 차이점에 관한 설명입니다.

- 대량의 자료나 많은 수의 항목을 저장할 목적이 아닌 활동 프로시듀어와 프로그램 간 통신을 위해 자료 대기행렬이 개선되었습니다. 대량의 자료나 많은 수의 항목을 저장할 때는 데이터베이스 파일을 대기행렬로 사용해야 합니다.
- 자료를 장기간 저장할 목적으로 자료 대기행렬을 사용해서는 안됩니다. 자료를 장기간 보관하려면 데이터베이스 파일을 사용해야 합니다.
- 자료 대기행렬을 사용할 때 시스템이 종료되기 전 완전히 처리되지 않은 항목을 회복시키려는 경우에는 프로그램에 비정상 종료 루틴을 포함시켜야 합니다.
- 정기적으로(예: 하루에 한 번씩) 안전한 시점에 자료 대기행렬을 삭제했다가 다시 작성하는 것이 좋습니다. 지나치게 많은 수의 항목을 제거하지 않고 그대로 두면 성능에 영향을 미칠 수 있습니다. 자료 대기행렬을 정기적으로 다시 작성하여 자료 대기

행렬을 최적의 크기로 유지할 수 있습니다. 『자료 대기행렬이 사용하는 기억장치 관리』에 설명된 자동 재생 피쳐를 사용하는 것이 더욱 효과적일 수 있습니다.

메세지 대기행렬과의 유사성

자료 대기행렬은 프로시듀어와 프로그램이 나중에 다른 프로시듀어나 프로그램이 수신할 자료를 대기행렬로 송신할 수 있다는 점에서 메세지 대기행렬과 유사합니다. 그러나 자료 대기행렬에서는 두 개 이상의 프로그램이 동시에 수신을 지연할 수 있는 반면, 메세지 대기행렬에서는 한번에 하나의 프로그램만이 수신을 지연할 수 있습니다(둘 이상의 프로그램이 대기하고 있어도 하나의 프로그램만이 자료 대기행렬로부터 항목을 수신함). 자료 대기행렬상의 항목은 선입선출, 후입선출 또는 키순 대기행렬 순서 중 하나로 처리됩니다. 항목이 수신되면 이 항목은 자료 대기행렬에서 제거됩니다.

자료 대기행렬 사용 시 요구사항

자료 대기행렬을 사용하기 위해서는 먼저 CRTDTAQ(자료 대기행렬 작성) 명령을 사용하여 자료 대기행렬을 작성해야 합니다. 예를 들면, 다음과 같습니다.

```
CRTDTAQ DTAQ(MYLIB/INPUT)  MAXLEN(128)
        TEXT('Sample data queue')
```

필수 매개변수 MAXLEN이 자료 대기행렬로 송신되는 항목의 최대 길이(1에서 64,512자)를 지정합니다.

자료 대기행렬이 사용하는 기억장치 관리

각 항목은 자료 대기행렬로 송신될 때 기억장치를 할당 받습니다. 할당 받은 기억장치가 CRTDTAQ(자료 대기행렬 작성) 명령에 지정된 자료 대기행렬의 최대 입력 길이 값입니다. 자료 대기행렬로부터 하나의 항목을 수신할 때 자료 대기행렬이 항목을 제거하지만 보조 기억장치를 비우지는 않습니다. 자료 대기행렬에 신규 항목을 송신할 때 시스템이 보조 기억장치를 다시 사용합니다. 대기행렬로 송신된 항목들을 수신하지 않을 때에는 대기행렬이 더 커집니다. 초기 항목 수를 초과하지 않는 대기행렬이 적을수록 성능이 좋습니다. 자료 대기행렬이 너무 커진 경우에는 DLTDTAQ(자료 대기행렬 삭제) 명령을 사용하여 자료 대기행렬을 삭제하십시오. 자료 대기행렬을 삭제했으면, CRTDTAQ(자료 대기행렬 작성) 명령을 사용하여 대기행렬을 다시 작성하십시오.

릴리스 V4R5M0 이상의 경우 자료 대기행렬의 크기를 관리하는 또 다른 방법이 있습니다. 이것은 CRTDTAQ 명령에 SIZE 및 AUTORCL 키워드를 사용하는 것입니다. SIZE 키워드를 사용하여 자료 대기행렬의 최대 항목 수와 초기 항목 수를 지정할 수 있습니다. 확장된 대기행렬에 대해 AUTORCL 키워드를 사용하면 자료 대기행렬이 비어 있을 때 자료 대기행렬이 자동으로 기억장치를 재생하도록 할 것인지 여부를 지정할 수 있습니다. 대기행렬에 할당된 상태로 유지되는 기억장치의 용량은 대기행렬 작성 시 대기행렬에 지정된 초기 항목 수와 같습니다. AUTORCL에 디폴트 값인 *NO가 있으면 사용되지 않은 공간으로부터 시스템이 자동으로 기억장치를 재생시키지 않습니다. 자

료 대기행렬이 사용하는 기억장치를 재생시키려면 앞에서 설명한 대로 기억장치를 삭제한 다음 재작성해야 합니다. 대기행렬의 크기에 따라서는 자동 재생에 많은 비용이 들 수도 있으므로 CRTDTAQ 명령에 지정하는 초기 항목 수는 자료 대기행렬에 예상되는 가장 큰 항목 수로 설정해야 합니다. 초기 항목 수를 너무 작게 설정하면 시스템이 재생 기능을 더 자주 실행해야 합니다.

자료 대기행렬 할당

어플리케이션에서 한번에 둘 이상의 작업이 자료 대기행렬에 액세스하지 못하도록 요구하면 자료 대기행렬을 사용하기 전에 ALCOBJ(오브젝트 할당) 명령을 포함시키도록 어플리케이션을 코드화해야 합니다. ALCOBJ 명령을 사용하는 어플리케이션을 끝냈을 때는 DLCOBJ(오브젝트 할당 해제) 명령을 사용하여 자료 대기행렬의 할당을 해제해야 합니다.

ALCOBJ 명령은 어떤 다른 작업이 자료 대기행렬로부터 자료를 송수신하거나 자료 대기행렬을 지우지 못하도록 작업을 제한하지 *않습니다*. 그러나 자료 대기행렬을 사용하기 전에 모든 어플리케이션에 ALCOBJ 명령이 포함되도록 코드화되었으면, 이미 다른 작업에 할당된 자료 대기행렬을 다시 할당할 수 없으므로 두 개 이상의 작업이 동시에 자료 대기행렬을 사용할 수 없습니다.

자료 대기행렬이 다른 작업에 이미 할당되어 있어 할당될 수 없으면, 시스템이 오류 메세지 CPF1002를 발행합니다. MONMSG(메세지 모니터) 명령은 메세지를 모니터하고 오류 메세지에 응답할 수 있도록 어플리케이션 프로시듀어에서 사용할 수 있습니다. 사용이 가능한 응답으로는 사용자에게 메세지를 송신하는 것과 자료 대기행렬을 다시 할당하는 것이 있습니다. 보다 자세히 알려면 280 페이지의 『CL 프로그램이나 프로시듀어에서의 메세지 모니터』 부분을 참조하십시오.

자료 대기행렬 사용의 예

다음 예에서는 자료 대기행렬 파일을 처리하기 위한 세 가지 방법에 대해 설명합니다.

예 1: 자료 대기행렬에서 자료를 수신하기 위해 2시간까지 대기

다음 예에서 프로그램 B는 자료 대기행렬로부터 항목을 수신하기 위해 2시간(7200초) 동안 대기하도록 지정합니다. 프로그램 A는 라이브러리 QGPL 안의 자료 대기행렬 DTAQ1로 항목을 송신합니다. 프로그램 A가 2시간 이내에 항목을 송신하면 프로그램 B는 이 자료 대기행렬로부터 항목을 수신하여 즉시 처리를 시작합니다. 프로시듀어 A가 항목을 송신하지 않고 2시간이 경과되면 프로그램 B는 리턴된 필드 길이가 0이 되므로 시간종료 상태를 처리합니다. 프로그램 B는 이 시간종료 상태가 발생할 때까지 계속해서 항목을 수신합니다. 여기서는 프로그램이 CL로 작성되어 있지만 고급 언어로도 작성할 수 있습니다.

자료 대기행렬은 다음 명령으로 작성됩니다.

```
CRTDTAQ DTAQ(QGPL/DTAQ1) MAXLEN(80)
```

이 예에서 모든 자료 대기행렬 항목의 길이는 80바이트입니다.

프로그램 A에서는 다음 명령문이 자료 대기행렬과 관련된 부분입니다.

```
   PGM
        DCL  &FLDLEN  *DEC  LEN(5 0)  VALUE(80)
        DCL  &FIELD  *CHAR  LEN(80)
.
.(대기행렬로 송신될 자료를 판별함)
.
CALL  QSNDDTAQ  PARM(DTAQ1 QGPL &FLDLEN &FIELD)
.
.
.
```

프로그램 B에서는 다음 명령문이 자료 대기행렬과 관련된 부분입니다.

```
   PGM
        DCL  &FLDLEN  *DEC  LEN(5 0)  VALUE(80)
        DCL  &FIELD  *CHAR  LEN(80)
        DCL  &WAIT  *DEC  LEN(5 0)  VALUE(7200)   /* 2 hours */
        .
        .
        .
LOOP:   CALL QRCVDTAQ  PARM(DTAQ1 QGPL &FLDLEN &FIELD &WAIT)
        IF    (&FLDLEN *NE 0) DO   /* Entry received */
               .
               . (process data from data queue)
               .
               GOTO LOOP   /* Get next entry from data queue */
 ENDDO
        .
        . (no entries received for 2 hours; process time-out condition)
        .
```

예 2: 화면 파일과 ICF 파일로부터의 입력 대기

다음 예에는 작업이 하나만 있으므로 자료 대기행렬의 일반적인 사용과는 다릅니다. 여기에서 자료 대기행렬은 두 작업 사이에서가 아닌 한 작업 안에서 통신 오브젝트를 처리합니다.



이 예에서 프로그램은 화면 파일과 ICF 파일로부터 입력을 기다리고 있습니다. 프로그램은 자료를 하나씩 기다리는 대신에 자료 대기행렬을 사용하여 하나의 오브젝트(자료 대기행렬)에서 자료를 기다릴 수 있습니다. 프로그램은 QRCVDTAQ를 호출한 다음,

화면 파일과 ICF 파일에서 지정된 자료 대기행렬에 놓일 항목을 기다립니다. 두 파일이 동일한 자료 대기행렬을 지정합니다. 두 파일 가운데 어느 한 파일이든지 자료를 사용할 수 있을 때 화면 자료 관리와 ICF 자료 관리 지원에 의해 두 가지 유형의 항목이 대기행렬에 기록됩니다. ICF 파일 항목은 *ICFF로 시작하고, 화면 파일 항목은 *DSPF로 시작합니다.

자료 대기행렬상에 놓인 화면 파일이나 ICF 파일 항목의 길이는 80자이고, 다음 리스트에 서술된 필드 속성을 포함하고 있습니다. 따라서 CRTDSPF, CHGDSPF, OVRDSPF, CRTICFF, CHGICFF, OVRICFF 명령을 사용하여 지정된 자료 대기행렬은 적어도 길이가 80자여야 합니다.

위치(및 자료 유형)
설명

1-10(문자)
자료 대기행렬에 항목을 넣은 파일의 유형. 이 필드는 다음 두 값 중 하나가 됩니다.

*ICFF: ICF 파일의 경우
*DSPF: 화면 파일의 경우

자료 대기행렬로부터 자료를 수신하는 작업이 단지 하나의 화면 파일이나 ICF 파일을 여는 경우 이 필드만이 자료 대기행렬로부터 수신한 항목의 유형을 결정합니다.

11-12(2진)
파일의 고유 ID. ID 값은 파일의 개방 피드백 영역에 있는 값과 같습니다. 이 필드는 자료 대기행렬에 항목을 넣는 동일한 이름의 파일이 둘 이상일 경우에만 자료 대기행렬로부터 항목을 수신하는 프로그램에 의해 사용됩니다.

13-22(문자)
화면 파일명 또는 ICF 파일명. 이것은 대체(override)가 모두 처리된 후에 실제로 열리는 파일명이며, 파일의 개방 피드백 영역에 있는 파일명과 같습니다. 이 필드는 둘 이상의 화면 파일이나 ICF 파일이 자료 대기행렬에 항목을 넣는 경우에만 자료 대기행렬로부터 항목을 수신하는 프로그램에 의해 사용됩니다.

23-32(문자)
파일이 놓일 라이브러리. 이것은 대체가 모두 처리된 후의 라이브러리명이며, 파일의 개방 피드백 영역에 있는 라이브러리명과 같습니다. 이 필드는 둘 이상의 화면 파일이나 ICF 파일이 자료 대기행렬에 항목을 넣는 경우에만 자료 대기행렬로부터 항목을 수신하는 프로그램에 의해 사용됩니다.

33-42(문자)
대체가 모두 처리된 후의 프로그램 장치명. 이것은 개방 피드백 영역의 프로그램 장치 정의 리스트에 있는 것과 같습니다. 파일 유형이 *DSPF이면 이는 명령 또는 Enter 키가 입력된 표시장치명이고, 파일 유형이 *ICFF이면 이는 사

용이 가능한 자료가 들어 있는 프로그램 장치명입니다. 이 필드는 자료 대기행렬에 항목을 넣은 파일이 자료 대기행렬 항목을 수신하기 이전에 둘 이상의 초청 장치나 세션을 가진 경우에만 자료 대기행렬로부터 항목을 수신하는 프로그램에 의해 사용됩니다.

43-80(문자)
예약.

다음 예는 앞에서 설명한 프로그램이 사용할 수 있는 코딩 논리를 보여줍니다.

```
 .
 .
 .
 .
 OPEN DSPFILE ... /* Open the Display file. DTAQ parameter specified on*/
                  /* CRTDSPF, CHGDSPF, or OVRDSPF for the file.        */

 OPEN ICFFILE ... /* Open the ICF file. DTAQ parameter specified on    */
                  /* CRTICFF, CHGICFF, or OVRICFF for the file.        */

.
.
        DO
  WRITE DSPFILE   /* Write with Invite for the Display file            */
  WRITE ICFFILE   /* Write with Invite for the ICF file                */

  CALL QRCVDTAQ   /* Receive an entry from the data queue specified    */
                  /* on the DTAQ parameters for the files. Entries     */
                  /* are placed on the data queue when the data is     */
                  /* available from any invited device or session     */
                  /* on either file.                                   */
                  /* After the entry is received, determine which file */
                  /* has data available, read the data, process it,    */
                  /* invite the file again and return to process the   */
                  /* next entry on the data queue.                     */
  IF 'ENTRY TYPE' FIELD = '*DSPF     ' THEN   /* Entry is from display */
    DO                                       /*  file. Since this entry*/
                                             /* does not contain the   */
                                             /* data received, the data*/
                                             /* must be read from the  */
                                             /* file before it can be  */
      READ DATA FROM DISPLAY FILE            /* processed.             */
      PROCESS INPUT DATA FROM DISPLAY FILE
      WRITE TO DISPLAY FILE                   /* Write with Invite    */
         END
  ELSE                                        /* Entry is from ICF     */
                                             /*  file. Since this entry*/
                                             /* does not contain the   */
                                             /* data received, the data*/
                                             /* must be read from the  */
                                             /* file before it can be  */
                                             /* processed.             */
      READ DATA FROM ICF  FILE
      PROCESS INPUT DATA FROM ICF FILE
      WRITE TO ICF FILE                       /* Write with Invite     */
  LOOP BACK TO RECEIVE ENTRY FROM DATA QUEUE
.
.
.
        END
```

예 3: 화면 파일과 자료 대기행렬로부터의 입력 대기

다음 예에서 작업 B의 프로그램은 사용 중인 화면 파일로부터의 입력과 작업 A로부터 자료 대기행렬에 도착하는 입력을 기다리고 있습니다. 화면 파일과 자료 대기행렬을 교대로 기다리는 대신, 프로그램은 하나의 오브젝트인 자료 대기행렬을 기다립니다.



프로그램이 QRCVDTAQ를 호출하고, 화면 파일에 지정된 자료 대기행렬에 들어갈 항목을 기다립니다. 작업 A도 동일한 자료 대기행렬에 항목을 넣습니다. 이 대기행렬에 기록되는 항목의 유형에는 화면 파일 항목과 사용자 정의 항목의 두 가지 유형이 있습니다. 화면 파일 항목은 자료가 화면 파일로부터 사용이 가능해질 때 화면 자료 관리에 의해 자료 대기행렬에 들어갑니다. 작업 A가 사용자 정의 항목을 자료 대기행렬에 넣습니다.

화면 파일 항목의 구조는 이전 예에 설명되어 있습니다.

작업 A에 의해 대기행렬에 들어가는 항목의 구조는 어플리케이션 프로그래머에 의해 정의됩니다.

다음 예는 작업 B의 어플리케이션 프로그램 논리를 코딩한 것입니다.

```
  .
  .
  .
  .
 OPEN DSPFILE ... /* Open the Display file. DTAQ parameter specified on*/
                  /* CRTDSPF, CHGDSPF, or OVRDSPF for the file.        */


.
.
         DO
  WRITE DSPFILE   /* Write with Invite for the Display file            */

  CALL QRCVDTAQ   /* Receive an entry from the data queue specified    */
                  /* on the DTAQ parameter for the file. Entries       */
                  /* are placed on the data queue either by Job A or   */
                  /* by display data management when data is           */
                  /* available from any invited device on the display  */
                  /* file.                                             */
                  /* After the entry is received, determine what type  */
```

```
                /* of entry it is, process it, and return to receive */
                /* the next entry on the data queue.                 */
  IF 'ENTRY TYPE' FIELD = '*DSPF     ' THEN   /* Entry is from display */
    DO                                       /*  file. Since this entry*/
                                             /* does not contain the   */
                                             /* data received, the data*/
                                             /* must be read from the  */
                                             /* file before it can be  */
      READ DATA FROM DISPLAY FILE            /* processed.             */
      PROCESS INPUT DATA FROM DISPLAY FILE
      WRITE TO DISPLAY FILE                   /* Write with Invite    */
        END
  ELSE                                       /* Entry is from Job A.  */
                                             /* This entry contains   */
                                             /* the data from Job A,  */
                                             /* so no read is required*/
                                             /* before processing the */
                                             /* data.                 */
      PROCESS DATA QUEUE ENTRY FROM JOB A
  LOOP BACK TO RECEIVE ENTRY FROM DATA QUEUE
.
.
.
        END
```

출력 대기행렬과 연관된 자료 대기행렬 작성

자료 대기행렬은 출력 대기행렬과 연관시킬 수 있습니다. 출력 대기행렬의 스풀 파일이 READY 상태가 되면 자료 대기행렬 항목이 자료 대기행렬로 송신됩니다. 자료 대기행렬을 작성할 때에는 CRTDTAQ(자료 대기행렬 작성) 명령을 사용하십시오. 최소한 128로 최대 메세지 길이(MAXLEN) 매개변수 값을 지정하십시오. 순서(SEQ) 매개변수 값은 *FIFO나 *LIFO여야 합니다.

출력 대기행렬과 연관된 자료 대기행렬에 대한 정보는 iSeries Information Center(http://www.iseries.ibm.com/infocenter)의 인쇄 범주에 있는 스풀 파일에 대한 자료 대기행렬 지원을 참조하십시오. 이 정보에는 샘플 자료 대기행렬 항목인 『레코드 유형 01 자료 대기행렬 입력 형식』도 포함되어 있습니다.

프로그램과 프로시듀어 간의 통신을 위한 자료 영역 사용

자료 영역은 시스템에서 수행되고 있는 작업이 액세스할 자료를 보유하는 데 사용하는 오브젝트입니다. 자료 영역은 프로시듀어나 파일의 존재 여부와 관계 없이 제한된 크기의 정보를 저장할 필요가 있을 때마다 언제든지 사용할 수 있습니다. 자료 영역의 일반적인 용도는 다음과 같습니다.

- 한 작업 안에서 정보를 전달하기 위한 영역을(각 작업의 QTEMP 라이브러리 안에) 제공할 경우.
- 한 작업 안에서 참조를 제어하기 위해 다음과 같이 내용이 자주 변경되는 필드를 제공할 경우.

  - 할당할 다음 순번을 제공할 경우.
  - 다음 검사 번호를 제공할 경우.
  - 사용할 다음 저장/복원 매체 볼륨을 제공할 경우.
- 세율이나 분배 리스트와 같이 여러 작업에서 사용될 상수 필드를 제공할 경우.
- 자료 영역을 필요로 하는 대규모 프로세스에 대한 액세스를 제한할 경우. 자료 영역을 한 사용자만 사용할 수 있도록 제한시킴으로써 다른 사용자가 동시에 처리하지 못하도록 할 수 있습니다.

로컬 또는 그룹 자료 영역 이외의 자료 영역을 작성하려면 CRTDTAARA(자료 영역 작성) 명령을 사용하십시오. 이렇게 하면 특정 라이브러리에 별도의 오브젝트를 작성해서 그것을 원하는 값으로 초기화할 수 있습니다. CL 프로시듀어나 프로그램에서 그 값을 사용하려면 RTVDTAARA(자료 영역 검색) 명령을 사용하여 현재 값을 프로시듀어나 프로그램의 변수로 가져가십시오. CL 프로시듀어나 프로그램에서 이 값을 변경하여 새로운 값을 자료 영역에 리턴시키려는 경우 CHGDTAARA(자료 영역 변경) 명령을 사용하십시오.

DSPDTAARA(자료 영역 표시) 명령을 사용하면 현재 값을 표시할 수 있습니다. DLTDTAARA(자료 영역 삭제) 명령을 사용하면 자료 영역을 삭제할 수 있습니다.

자료 영역을 저널할 수 있습니다. 이 기능은 IPL 또는 장애가 발생했을 때 오브젝트가 일부 변경 조치 중에 있더라도 오브젝트를 일관적인 상태로 회복시킬 수 있도록 합니다. 저널링은 리모트 시스템에 대해 자료 영역 저널의 복제를 제공하기도 합니다(예를 들면, 리모트 저널을 사용하여). 이는 시스템이 어플리케이션 작업을 복제하기 위해 유사 환경에서 조치를 재생성하게 합니다. iSeries 서버에서 저널링 지원에 대한 자세한 정보는 백업 및 회복  책을 참조하십시오.

로컬 자료 영역

로컬 자료 영역은 자동 시작 작업, 판독기가 시스템에서 시작한 작업, 그리고 서브시스템 모니터 작업을 포함한 시스템 안의 각 작업에 대해 작성됩니다.

시스템은 공백으로 초기화되고, 그 길이가 1024이며, 유형이 *CHAR인 로컬 자료 영역을 작성합니다. SBMJOB 명령을 사용하여 작업을 제출하면 제출하는 작업의 로컬 자료 영역 값이 제출된 작업의 로컬 자료 영역에 복사됩니다. CHGDTAARA, RTVDTAARA 및 DSPDTAARA 명령의 DTAARA 키워드에 *LDA를 지정하거나 서브스트링 내장 기능(%SST)에 *LDA를 지정하여 사용자 작업의 로컬 자료 영역을 참조할 수 있습니다.

다음은 로컬 자료 영역에 대한 사항입니다.

- 로컬 자료 영역은 다른 작업으로부터 참조될 수 없습니다.
- 로컬 자료 영역은 사용자가 작성, 삭제 또는 할당할 수 없습니다.

- 로컬 자료 영역과 관련된 라이브러리는 없습니다.
- 2차 스레드에서는 로컬 자료 영역(LDA)을 변경할 수 없습니다.
- ILE CL 컴파일러는 2차 스레드에서 실행 중인 프로시듀어가 초기(1차) 스레드에서 로컬 자료 영역(LDA)을 변경하는 동안 그것에 액세스하지 못하도록 보장하기 위해 코드를 생성합니다.

로컬 자료 영역의 내용은 라우팅 단계의 경계에 존재합니다. 따라서 TFRJOB(작업 전송), TFRBCHJOB(일괄처리 작업 전송), RRTJOB(작업 라우트) 또는 RETURN 명령을 사용해도 로컬 자료 영역의 내용에는 영향을 미치지 않습니다.

다음과 같은 경우에는 자료 영역을 사용할 수 있습니다.

- 매개변수 리스트를 사용하지 않고 프로시듀어나 프로그램에 정보를 전달할 경우.
- 정보를 로컬 자료 영역에 로드하고 작업을 제출함으로써 제출된 작업으로 정보를 전달할 경우. 그러면 사용자는 제출된 작업 안에서 자료에 액세스할 수 있습니다.
- CL 프로시듀어나 프로그램에서 다른 유형의 자료 영역으로 액세스하는 것보다 성능이 향상될 경우.
- 자료 영역을 작성하고 삭제하는 데 따른 부수적인 작업을 하지 않고 정보를 저장할 경우.

대부분의 고급 언어(HLL)도 로컬 자료 영역을 사용할 수 있습니다. SBMxxxJOB와 STRxxxRDR 명령의 사용으로 인해 공백으로 초기화된 로컬 자료 영역으로 작업이 시작됩니다. SBMJOB 명령을 사용해야만, 제출하는 작업의 로컬 자료 영역 내용을 새로운 작업에 전달할 수 있습니다.

그룹 자료 영역

시스템은 대화식 작업이 그룹 작업이 될 때(그룹 속성 변경 [CHGGRPA] 명령을 사용하여) 그룹 자료 영역을 작성합니다. 하나의 그룹에 대해서는 그룹 자료 영역이 하나만 존재할 수 있습니다. 그룹 안의 최종 작업이 종료되거나(ENDJOB, SIGNOFF, ENDGRPJOB 명령이나 비정상 종료로), 작업이 그룹 작업에 더 이상 속하지 않으면 (GRPJOB(*NONE)이 지정된 CHGGRPA 명령을 사용하여), 그룹 자료 영역이 삭제됩니다.

그룹 자료 영역은 공백으로 초기화되고, 길이는 512이며, 유형은 *CHAR입니다. CHGDTAARA, RTVDTAARA, DSPDTAARA 명령의 DTAARA 매개변수에 *GDA를 지정하면 그룹 작업 안에서 그룹 자료 영역을 사용할 수 있습니다. 그룹 자료 영역은 그룹 안의 모든 작업에 액세스할 수 있습니다.

다음은 그룹 자료 영역에 대한 사항입니다.

- 그룹 자료 영역은 서브스트링 내장 기능(%SUBSTRING 또는 %SST)의 문자 변수에 대한 대체로 사용할 수 없습니다. (그러나 서브스트링 기능이 사용하는 512바이트 문자 변수는 그룹 자료 영역 안이나 밖으로 이동시킬 수 있습니다.)
- 그룹 자료 영역은 그룹 외부의 작업에 의해서는 참조될 수 없습니다.
- 그룹 자료 영역은 사용자가 작성, 삭제 또는 할당할 수 없습니다.
- 그룹 자료 영역과 관련된 라이브러리는 없습니다.

그룹 자료 영역의 내용은 TFRGRPJOB(그룹 작업 전송) 명령에 의해 변경되지 않습니다.

다른 자료 영역을 사용하는 것처럼 그룹 자료 영역을 사용할 수 있는 점 이외에도 같은 그룹 안에 있는 그룹 작업 간의 정보 통신을 위해 그룹 자료 영역을 사용할 수 있습니다. 예를 들면, CHGGRPA(그룹 작업 속성 변경) 명령을 발행한 후 아래의 명령을 사용하여 그룹 자료 영역의 값을 설정할 수 있습니다.

```
CHGDTAARA  DTAARA(*GDA)  VALUE('January1988')
```

이 명령은 프로그램에서 실행하거나 워크스테이션 사용자가 직접 발행할 수 있습니다.

그룹에 있는 다른 CL 프로시듀어나 프로그램은 다음 CL 명령을 사용하여 그룹 자료 영역의 값을 검색할 수 있습니다.

```
RTVDTAARA  DTAARA(*GDA)  RTNVAR(&GRPARA)
```

이 명령은 그룹 자료 영역의 값(January1988)을 CL 변수 &GRPARA에 넣습니다.

프로그램 초기화 매개변수(PIP) 자료 영역

작업이 시작될 때 각각의 사전 시작 작업에 대해 PIP 자료 영역(PDA)이 작성됩니다. PDA의 오브젝트 부속 유형은 정규 자료 영역과는 다릅니다. PDA는 특수 값 이름 *PDA에 의해서만 참조될 수 있습니다. PDA의 *크기*는 2000바이트이나 그 안에 들어갈 매개변수의 수에는 제한이 없습니다.

RTVDTAARA, CHGDTAARA, DSPDTAARA CL 명령과 RTVDTAARA, CHGDTAARA 매크로 명령은 자료 영역명 매개변수에 대해 특수 값 *PDA를 지원합니다.

리모트 자료 영역

분산 자료 관리(DDM) 파일을 사용할 경우 사용자가 리모트 자료 영역에 액세스할 수 있습니다. 리모트 서버에 상주하는 자료를 검색할 때 한 서버에 상주하는 어플리케이션 프로그램을 변경하거나 다시 컴파일할 필요가 없습니다. 올바른 자료 영역에 액세스하고 있는지를 확인하기 위해서는 다음 중 하나를 수행해야 하는 경우도 있습니다.

- 표준 자료 영역을 삭제하고 원래 표준 자료 영역과 동일한 이름을 가진 DDM 자료 영역을 작성합니다.

- 표준 자료 영역의 이름을 변경합니다.

사용자는 다음 명령을 사용하여 DDM 자료 영역을 작성할 수 있습니다.

```
CRTDTAARA DTAARA(LOCALLIB/DDMDTAARA) TYPE(*DDM)
   RMTDTAARA(REMOTELIB/RMTDTAARA) RMTLOCNAME(SYSTEMB)
   TEXT('DDM data area to access data area on SYSTEMB')
```

CL 프로그램에서 리모트 서버에 있는 자료 영역의 값을 사용하려면 RTVDTAARA(자료 영역 검색) 명령을 사용하십시오. 현재 값을 사용자 프로그램의 변수로 가져오려면 DDM 자료 영역명을 지정하십시오. CL 프로그램에서 이 값을 변경하여 새로운 값을 리모트 자료 영역에 리턴시키려면 CHGDTAARA(자료 영역 변경) 명령 및 동일한 DDM 자료 영역을 지정하여 사용하십시오.

DSPDTAARA(자료 영역 표시) 명령을 사용할 때 DDM 자료 영역 이름을 지정한 경우에는 리모트 자료 영역의 값이 아닌 DDM 자료 영역의 값이 표시됩니다.
DLTDTAARA(자료 영역 삭제) 명령을 사용하면 자료 영역을 삭제할 수 있습니다.

DDM 자료 영역에 대한 자세한 정보는 **iSeries Information Center**의 *프로그래밍* 범주에서 *CL* 섹션을 참조하십시오.

자료 영역 작성

변수와 달리, 자료 영역은 오브젝트이므로 사용하기 전에 작성해야 합니다. 자료 영역은 다음과 같이 작성될 수 있습니다.

- 최대 2000자 길이의 문자 스트링.
- CL 프로그램이나 프로시듀어에서만 사용되는지 아니면 다른 고급 언어 프로그램이나 프로시듀어와도 함께 사용될 것인지의 여부에 따라 서로 다른 속성의 10진값. CL 프로시듀어와 프로그램의 경우 자료 영역은 소수점 왼쪽으로 15자릿수까지, 오른쪽으로 9자릿수까지 가질 수 있지만, 총 15자릿수까지만 가능합니다. 다른 언어의 경우 소수점 왼쪽으로 15자릿수, 오른쪽으로는 9자릿수를 가질 수 있으며, 전체 24자릿수까지 가능합니다.
- '0' 또는 '1'의 논리값, 여기서 '0'은 OFF, 거짓, 아니오를 의미하고, '1'은 ON, 참, 예를 의미합니다.

자료 영역 작성 시, 자료 영역의 초기값을 지정할 수도 있습니다. 초기값을 지정하지 않으면 다음 값으로 가정됩니다.

- 10진수는 0
- 문자는 공백
- 논리 값은 '0'

자료 영역을 작성하려면 자료 영역 작성(CRTDTAARA) 명령을 사용하십시오. 다음 예에서는 한 프로그램에서 다른 프로그램으로 고객 번호를 전달하기 위해 자료 영역이 작성됩니다.

```
CRTDTAARA  DTAARA(CUST) TYPE(*DEC) +
           LEN(5 0) TEXT('Next customer number')
```

자료 영역 잠금 및 할당

CHGDTAARA 명령은 명령 처리 중 자료 영역에서 *SHRUPD(갱신 공유) 잠금을 사용합니다. RTVDTAARA와 DSPDTAARA 명령은 명령 처리 중 자료 영역에서 *SHRRD(읽기 공유) 잠금을 사용합니다. 한 자료 영역에서 둘 이상의 조작을 수행 중이면, ALCOBJ(오브젝트 할당) 명령을 사용하여 조작이 완료될 때까지 다른 사용자가 자료 영역에 액세스하지 못하도록 할 수 있습니다. 예를 들면, 동시에 수행 중인 작업에 의해 읽혀지고 증가되는 값이 자료 영역에 들어 있는 경우 ALCOBJ 명령을 사용한 읽기 및 갱신 조작에서 모두 이 값을 보호할 수 있습니다. 오브젝트 할당에 대해서는 제 4 장을 참조하십시오.

다른 언어(CL 이외의 언어)에서 자료 영역을 처리하는 방법에 대해 알려면 해당 고급 언어 안내서를 참조하십시오.

자료 영역 표시

자료 영역의 속성(이름, 라이브러리, 유형, 길이, 자료 영역 텍스트 설명)과 값은 표시가 가능합니다. DSPDTAARA(자료 영역 표시) 명령에 관한 자세한 설명은 **iSeries Information Center**의 *프로그래밍* 범주에서 *CL* 섹션을 참조하십시오.

표시장치에서는 선행 0을 제거한 24자리 형식을 사용합니다.

자료 영역 변경

CHGDTAARA(자료 영역 변경) 명령은 지정된 자료 영역의 값을 모두 또는 일부 변경합니다. 이 명령이 자료 영역의 다른 속성을 변경하지는 않습니다. 새로운 값으로는 상수 또는 CL 변수가 가능합니다. 이 명령이 CL 프로시듀어에 있으면, 프로그램 작성 시 자료 영역이 존재할 필요가 없습니다.

자료 영역 검색

RTVDTAARA(자료 영역 검색) 명령은 지정된 자료 영역을 모두 또는 일부 검색하여 CL 변수로 복사합니다. CL 프로그램이 작성될 때 자료 영역이 존재할 필요는 없으며, CL 변수가 자료 영역의 이름과 같을 필요도 없습니다. 이 명령은 지정된 자료 영역의 내용을 검색만 할 뿐 변경하지는 않습니다.

자료 영역 검색의 예

예 1

ORDINFO라고 명명된 자료 영역을 사용하여 주문 파일의 상태를 추적하고 있다고 가정하겠습니다. 이 자료 영역은 다음과 같이 설계되어 있습니다.

- 위치 1은 O(열림), P(처리) 또는 C(완료).

- 위치 2는 I(반입) 또는 O(반출).
- 위치 3부터 5까지는 주문 담당자명의 첫 글자.

프로시듀어에서 이 필드는 다음과 같이 선언될 수 있습니다.

```
DCL VAR(&ORDSTAT) TYPE(*CHAR) LEN(1)
DCL VAR(&STOCKC) TYPE(*CHAR) LEN(1)
DCL VAR(&CLERK) TYPE(*CHAR) LEN(3)
```

주문 상태를 검색하여 &ORDSTAT에 넣으려면 다음과 같이 입력합니다.

```
RTVDTAARA DTAARA(ORDINFO (1  1))  RTNVAR(&ORDSTAT)
```

재고 상태를 검색하여 &STOCK에 넣으려면 다음과 같이 입력합니다.

```
RTVDTAARA DTAARA(ORDINFO (2  1)) RTNVAR(&STOCKC)
```

담당자의 이름 첫 글자를 검색하여 &CLERK에 넣으려면 다음과 같이 입력합니다.

```
RTVDTAARA DTAARA(ORDINFO (3  3)) RTNVAR(&CLERK)
```

RTVDTAARA 명령을 사용할 때는 액세스할 자료 영역이 필요합니다. 여러 개의 서브필드를 검색하는 경우 변수로 자료 영역 전체를 검색한 후 서브스트링 내장 기능을 사용하여 서브필드를 추출하는 것이 효율적입니다.

예 2

RTVDTAARA 명령의 다음 예에서는 지정된 5자의 자료 영역의 내용을 3자의 변수에 넣습니다. 이 예에 대해서는 다음을 보십시오.

- 이름이 DA1이고(라이브러리 MYLIB에 있음) 초기값이 ABCDE인 5자의 자료 영역을 작성합니다.
- &CLVAR1이라는 이름의 3자 변수를 선언합니다.
- DA1의 최종 세 번째 위치의 내용을 &CLVAR1에 복사합니다.

이렇게 하려면 다음의 명령을 입력합니다.

```
CRTDTAARA DTAARA(MYLIB/DA1) TYPE(*CHAR) LEN(5) VALUE(ABCDE)
.
.
.
DCL VAR(&CLVAR1) TYPE(*CHAR) LEN(3)
RTVDTAARA DTAARA(MYLIB/DA1 (3 3)) RTNVAR(&CLVAR1)
```

&CLVAR1에는 이제 'CDE'가 들어 있습니다.

예 3

RTVDTAARA 명령의 다음 예에서는 5자리 10진 자료 영역의 내용을 5자리 10진 변수에 넣습니다. 이 예에 대해서는 다음을 보십시오.

- 이름이 DA2이고(라이브러리 MYLIB에 있음) 소수 자릿수가 2자리이며, 초기값이 12.39인 5자리 자료 영역을 작성합니다.

- &CLVAR2라는 이름의 1자리 소수 자릿수를 가진 5자리 변수를 선언합니다.
- &CLVAR2에 DA2의 내용을 복사합니다.

이렇게 하려면 다음의 명령을 입력합니다.

```
CRTDTAARA DTAARA(MYLIB/DA2) TYPE(*DEC) LEN(5 2) VALUE(12.39)
.
.
.
DCL  VAR(&CLVAR2) TYPE(*DEC) LEN(5 1)
RTVDTAARA DTAARA(MYLIB/DA2) RTNVAR(&CLVAR2)
```

&CLVAR2에는 이제 0012.3이 들어 있습니다(소수부에서 잘림).

자료 영역 변경 및 검색의 예

다음은 문자 서브스트링 연산에 CHGDTAARA와 RTVDTAARA 명령을 사용한 예입니다.

이 예에 대해서는 다음을 보십시오.

- 이름이 DA1이고(라이브러리 MYLIB에 있음) 초기값이 ABCD5678IJ인 10자의 자료 영역을 작성합니다.
- &CLVAR1이라고 명명된 5자 변수를 선언합니다.
- 자료 영역 DA1(위치 5에서 시작하여 길이가 4인)의 내용을 값 EFG(G 다음에 하나의 공백이 채워짐)로 변경합니다.
- 자료 영역 DA1의 내용(위치 5에서 시작하여 길이가 5인)을 검색하여 CL 변수 &CLVAR1에 넣습니다.

이렇게 하려면 다음의 명령을 입력합니다.

```
DCL VAR(&CLVAR1) TYPE(*CHAR) LEN(5)
.
CRTDTAARA DTAARA(MYLIB/DA1) TYPE(*CHAR) LEN(10) +
     VALUE('ABCD5678IJ')
.
.
.
CHGDTAARA DTAARA((MYLIB/DA1) (5 4)) VALUE('EFG')
RTVDTAARA DTAARA((MYLIB/DA1) (5 5)) RTNVAR(&CLVAR1)
```

변수 &CLVAR1에는 이제 'EFG I'가 들어 있습니다.

제 4 장 오브젝트와 라이브러리

오브젝트는 명령이 작업을 수행하는 기본 단위입니다. 예를 들면, 프로그램과 파일이 오브젝트가 될 수 있습니다. 오브젝트를 통해 iSeries 서버에서 자료를 찾고, 유지보수 및 처리할 수 있습니다. 이 경우 오브젝트나 기능을 사용하기 위해서는 사용하려는 오브젝트와 기능(명령)이 무엇인지만 알면 되고, 기억장치 주소까지는 알 필요가 없습니다.

주: 오브젝트는 라이브러리와 디렉토리에 모두 놓일 수 있습니다. (이전에는 오브젝트를 라이브러리에만 놓을 수 있었습니다.) 여기에서는 라이브러리에 놓인 오브젝트에 관해서만 설명합니다. 디렉토리에 대한 정보는 iSeries Information Center의 **데이터베이스 및 파일 시스템** 범주에 있는 *통합 파일 시스템* 주제를 참조하십시오.

오브젝트 유형과 공통 속성

서버에 있는 각 오브젝트 유형은 시스템 내에서 고유한 목적을 가지며, 해당 오브젝트 유형을 처리하는 관련 명령 세트를 가집니다. 오브젝트 유형, 오브젝트 유형 매개변수에 대해 매개변수 값으로 사용되는 약어 및 해당 유형에 속하는 오브젝트 정의에 대한 전체 리스트는 **iSeries Information Center**의 **프로그래밍** 범주에 있는 *CL* 섹션을 참조하십시오.

각 오브젝트 유형은 오브젝트를 설명하는 공통 속성 세트를 가지고 있습니다. 이 공통 속성은 140 페이지의 표 3에 나와 있습니다. DSPOBJD(오브젝트 설명 표시) 명령에 대한 온라인 도움말 정보에서 이러한 속성을 설명합니다. **iSeries Information Center** *프로그래밍* 범주의 **CL** 섹션에 나와 있는 명령 문서를 참조하십시오.

오브젝트에서 수행되는 기능

많은 기능이 오브젝트에서 수행될 수 있습니다. 일부 기능은 시스템이 자동으로 수행하며 그 나머지 기능은 사용자가 명령을 통해 요구합니다.

시스템이 자동으로 수행하는 기능

자동으로 수행되는 기능이 있어서 오브젝트를 일관되고 안전하며 올바른 방법으로 처리할 수 있습니다. 이러한 기능으로는 다음과 같은 것이 있습니다.

- 오브젝트 유형 검증. 시스템은 오브젝트의 유형과 오브젝트에 대해 수행될 기능의 유형을 검사하여 해당 유형의 오브젝트에 대해 그 기능을 수행할 수 있는지를 검증합니다. 예를 들면, CALL 명령에 지정된 오브젝트가 프로그램이 아닌 경우 호출 기능이 수행될 수 없습니다.

- 오브젝트 권한 검증. 시스템은 오브젝트와 기능을 검사하고 사용자가 그 오브젝트에 대해 해당 기능을 사용할 권한이 있는지를 검증합니다. 예를 들어, USERA에게 OBJB를 사용할 권한이 부여되지 않으면 USERA는 OBJB에 대해 어떤 기능도 수행할 수 없습니다.
- 오브젝트 잠금 강제. 두 명 이상의 사용자가 동시에 같은 오브젝트를 사용하려고 시도할 때 시스템이 오브젝트 무결성을 보장합니다. 오브젝트에 대한 동시 변경은 금지되어 있으므로 오브젝트가 변경되는 동안에는 그것을 사용할 수 없습니다.
- 오브젝트 손상 검출 및 통지. 시스템은 오브젝트를 처리하는 동안 오류를 모니터하여 알 수 없는 오브젝트 내용으로 인해 발생한 예기치 못한 장애를 사용자에게 알립니다. 이러한 장애는 오브젝트 손상을 표시하는 표준 메세지를 통하여 사용자에게 알려집니다. 시스템은 이러한 장애가 거의 없도록 설계되어 있으며 이러한 오류를 모니터하여 통지함으로써 무결성을 제공합니다.

명령을 사용하여 수행할 수 있는 기능

명령을 통하여 요구할 수 있는 기능에는 다음 두 가지 유형이 있습니다.

- 각 오브젝트 유형에 대한 특정 기능. 예를 들어, 작성, 변경 및 표시는 특정 기능입니다. 특정 기능들은 본문 가운데 오브젝트 유형을 설명하는 섹션에 나옵니다.
- 이 책에는 일반적으로 오브젝트에 적용되는 몇몇 공통 기능이 설명되어 있습니다.

표 2. 오브젝트에 대한 공통 기능

기능	페이지
라이브러리에서 여러 오브젝트 또는 하나의 오브젝트를 탐색합니다.	128
라이브러리 안의 오브젝트에 대해 권한을 지정합니다.	130
라이브러리에 오브젝트를 놓습니다.	134
오브젝트를 설명합니다.	139
오브젝트 설명을 표시합니다.	139
오브젝트 설명을 검색합니다.	144
시스템에서 사용하지 않는 오브젝트를 검출합니다.	147
라이브러리 간 오브젝트를 이동합니다.	153
오브젝트 사본을 작성합니다.	156
오브젝트의 이름을 변경합니다.	158
오브젝트를 삭제합니다.	163
오브젝트를 할당하고 할당을 해제합니다.	164
오브젝트의 잠금 상태를 표시합니다.	167
오브젝트 존재를 검사합니다.	172

라이브러리

iSeries 서버에서 오브젝트는 *라이브러리*라고 하는 특수 오브젝트로 그룹화됩니다. 오브젝트는 라이브러리를 사용하여 찾을 수 있습니다. 라이브러리에 있는 오브젝트에 액세스하려면 라이브러리와 오브젝트에 대한 권한이 있어야 합니다. 자세한 내용을 알려면 131 페이지의 『오브젝트에 대한 보안 고려사항』 부분과 130 페이지의 『라이브러리에 대한 권한 지정』 부분을 참조하십시오.

오브젝트명과 동일한 매개변수에 라이브러리명을 지정하는 경우 그러한 오브젝트명을 *규정된 이름*이라고 합니다.

라이브러리를 작성할 때 사용자는 라이브러리 작성 후 라이브러리를 넣을 사용자 보조 기억장치 풀(ASP: Auxiliary Storage Pool)을 지정할 수 있습니다. 라이브러리는 기본적인 사용자 ASP나 독립 ASP에 작성할 수 있습니다. 독립 ASP에 관한 자세한 정보는 독립 ASP 주제를 참조하십시오. 라이브러리 내에 작성된 모든 오브젝트는 그 라이브러리와 ASP에 작성됩니다.

예를 들어, 규정된 이름을 지정해야만 하는 명령을 입력할 경우 오브젝트명은 다음과 같을 수 있습니다.

```
DISTLIB/ORD040C
```

주문 입력 프로그램 ORD040C는 라이브러리 DISTLIB에 들어 있습니다.

프롬트 환경에서 명령을 입력할 때 규정된 이름을 입력하라고 프롬트되는 경우 오브젝트명과 라이브러리명에 대해 모두 프롬트가 나옵니다. 대부분의 명령에서 특정 라이브러리명 또는 *CURLIB(작업에 대한 현재 라이브러리)를 지정하거나 라이브러리 리스트를 사용할 수 있습니다. 라이브러리 리스트는 다음 섹션에서 설명됩니다.

라이브러리 리스트

규정된 이름을 지정할 수 있는 명령에 대해서는 라이브러리명을 지정하지 않아도 됩니다. 이렇게 하면 다음 중 하나의 결과가 나옵니다.

- 작성 명령의 경우 오브젝트가 작성되어 사용자의 현재 라이브러리인 *CURLIB나 오브젝트 유형에 따라 시스템 라이브러리에 들어갑니다. 예를 들어, 프로그램이 작성된 후 *CURLIB에 놓이고, 권한 부여 리스트는 작성된 후 QSYS에 놓입니다.
- 작성 명령 이외의 명령인 경우 시스템이 일반적으로 라이브러리 리스트를 사용하여 오브젝트를 찾습니다.

OS/400에서 사용되는 라이브러리 리스트는 다음 네 부분으로 구성됩니다.

시스템 부분

라이브러리 리스트의 시스템 부분에는 시스템에 필요한 오브젝트가 들어 있습니다.

제품 라이브러리

두 개의 제품 라이브러리가 라이브러리 리스트에 포함될 수 있습니다. 시스템은 QSYS 이외의 라이브러리 종속 언어와 유틸리티가 자신의 명령을 처리하도록 지원하기 위해 제품 라이브러리를 사용합니다.

사용자 명령과 메뉴도 종속 오브젝트가 반드시 발견될 수 있도록 CRTCMD(명령 작성)과 CRTMNU(메뉴 작성) 명령의 PRDLIB 매개변수에 제품 라이브러리를 지정할 수 있습니다.

제품 라이브러리는 시스템이 관리하며, 필요할 경우 제품 라이브러리(예: QRPG)를 라이브러리 리스트의 예약된 제품 라이브러리 부분에 자동으로 위치시킵니다. 제품 라이브러리는 현재 라이브러리의 사본이거나 라이브러리 리스트의 사용자 부분에 있는 라이브러리의 사본입니다.

예를 들어, 제품 라이브러리를 가진 명령이나 메뉴가 시작할 때 라이브러리 리스트에 제품 라이브러리가 있다고 가정하십시오. 이 경우 시스템은 신규 명령이 종료하거나 사용자가 신규 메뉴에서 빠져 나오기까지 라이브러리 안의 제품 라이브러리를 신규 제품 라이브러리로 대체합니다.

현재 라이브러리

현재 라이브러리는 라이브러리 리스트에 있는 라이브러리의 사본일 수 있으나 반드시 그래야 하는 것은 아닙니다. 값 *CURLIB(현재 라이브러리)는 대부분의 명령에서 라이브러리가 작업의 현재 라이브러리로 지정된 것은 무엇이든지 라이브러리명으로 사용될 수 있습니다. 라이브러리 리스트에 현재 라이브러리가 없고 *CURLIB가 라이브러리로 지정된 경우 QGPL이 사용됩니다.
CHGCURLIB(현재 라이브러리 변경)이나 CHGLIBL(라이브러리 리스트 변경) 명령을 사용하면 작업의 현재 라이브러리를 변경할 수 있습니다.

사용자 부분

라이브러리 리스트의 사용자 부분에는 시스템의 사용자 및 어플리케이션이 참조하는 라이브러리가 들어 있습니다. 사용자 부분, 제품 및 현재 라이브러리는 시스템의 각 작업마다 다를 수 있습니다. 이들은 250개의 라이브러리로 제한됩니다.

시스템과 함께 제공되거나 선택적으로 시스템에 설치할 수 있는 라이브러리 리스트를 보려면 483 페이지의 부록 C『사용권 프로그램(LP) 안의 IBM 제공 라이브러리』를 참조하십시오.

다음은 라이브러리 리스트 구조의 다이어그램입니다.



주: 소스 입력 유틸리티(SEU)가 사용되면 시스템은 라이브러리 QPDA를 제품 라이브러리 1에 놓습니다. SEU가 소스 코드 구문 검사에 사용되면 두 번째 제품 라이브러리가 제품 라이브러리 2에 추가될 수 있습니다. 예를 들어, RPG 소스 구문 검사가 진행되는 경우 QPDA는 제품 라이브러리 1이고 QRPG는 제품 라이브러리 2입니다. 대부분의 다른 시스템 기능에서는 제품 라이브러리 2가 사용되지 않습니다.

라이브러리 리스트를 사용하면 시스템에서 오브젝트를 찾는 것이 간단합니다. 각 작업은 관련된 라이브러리 리스트를 가지고 있습니다. 라이브러리 리스트를 사용하여 오브젝트를 찾을 때 리스트의 각 라이브러리는 지정된 이름과 유형의 오브젝트를 찾을 때까지 리스트에 있는 순서대로 탐색됩니다. 리스트에 유형과 이름이 같은 오브젝트가 둘 이상 존재하면 라이브러리 리스트에 먼저 나오는 라이브러리로부터 오브젝트가 사용됩니다. 다음 다이어그램은 라이브러리 리스트(*LIBL)가 사용된 경우와 라이브러리명이 지정된 경우의 오브젝트 탐색을 보여줍니다.

주: 다른 방법으로, *LIBL 대신 *NLVLIBL을 사용하여 명령을 규정할 수 있습니다. CL 프로그램에서 나온 명령을 명령 행이나 사용자가 일반적으로 명령을 입력하는 곳에 입력하십시오. 시스템이 *NLVLIBL을 사용하여 *CMD 오브젝트를 탐색할 라이브러리를 판별합니다. *NLVLIBL을 지정하여 라이브러리에 있는 자국어 지원 라이브러리만 탐색합니다.

*CMD 오브젝트 서명 및 확인에 관한 자세한 정보는 iSeries Information Center의 **보안** 범주에서 오브젝트 서명 확인 주제를 참조하십시오.



다음 다이어그램은 이름은 같으나 유형이 다른 두 개의 오브젝트가 동일한 라이브러리 리스트에 들어 있을 경우에 발생하는 현상을 보여줍니다. 시스템은 다음을 지정하여 라이브러리 리스트에서 CUSTINQ *FILE을 탐색합니다.

```
DSPOBJD OBJ(*LIBL/CUSTINQ) OBJTYPE(*FILE)
```



RBAFN541-0

일반적으로 라이브러리 리스트는 규정된 이름보다 더 큰 융통성이 있으며 사용하기 쉽습니다. 라이브러리명을 입력하지 않는다는 장점보다 더 중요한 것은, 어플리케이션을 변경하지 않고도 다른 라이브러리 리스트를 사용함으로써, 그 어플리케이션에서 다른 자료에 대한 기능을 수행할 수 있다는 점입니다. 예를 들어, CL 프로그램 PGMA가 자료 영역 CHKNBR을 갱신합니다. 이 경우 라이브러리명을 지정하지 않았으면, 프로그램은 라이브러리 리스트를 사용하여 다른 라이브러리 안에 있는 CHKNBR이라는 자료 영역을 갱신할 수 있습니다. 예를 들어, 다음 그림과 같이 JOBA와 JOBB가 모두 PGMA를 호출한다고 가정해 보겠습니다.



그러나 다음과 같은 경우에는 규정된 이름을 사용하는 것이 좋습니다.

- 사용 중인 오브젝트가 작업의 라이브러리 리스트에 없을 때
- 라이브러리 리스트에 동일한 이름의 오브젝트가 둘 이상 있으나 사용자가 특정 라이브러리에 있는 오브젝트를 원할 때
- 보안상의 이유로 특정 라이브러리가 반드시 사용되기를 원할 때

그러나 규정된 이름을 사용하여 프로그램을 호출하였으나 프로그램에서 이름이 규정되지 않은 파일을 열려고 할 경우에는 다음 예와 같이 파일이 라이브러리 리스트에 없으면 파일이 열리지 않습니다.



RBAFN527-0

PGMA에 대한 호출은 CALL 명령에 프로그램명이 규정되어 있기 때문에 성공적으로 이루어집니다. 그러나 프로그램이 파일 ORDENTP를 열려고 했으나 파일이 라이브러리 리스트의 어떤 라이브러리에도 없고, 파일명이 규정되지 않았기 때문에 파일을 열 수 없습니다. 라이브러리 DISTLIB2를 라이브러리 리스트에 추가하거나 규정된 파일명이 사용된 경우에는 프로그램이 파일을 열 수 있습니다. 일부 고급 언어(HLL)에서는 규정된 파일명을 지정할 수 없습니다. 그러한 경우 OVRxxx(대체) 명령을 사용하여 규정된 이름을 지정할 수 있습니다.

작업의 라이브러리 리스트

각 작업의 라이브러리 리스트는 시스템 부분, 사용자 부분, 현재 라이브러리 및 제품 라이브러리의 네 부분으로 구성됩니다. 시스템 부분만 항상 라이브러리 리스트에 포함됩니다.

시스템이 공급될 때 시스템 값 QSYSLIBL에는 라이브러리 리스트의 시스템 부분이 될 라이브러리명이 포함되어 있습니다. 제공되는 값은 QSYS, QSYS2, QHLPSYS, QUSRSYS입니다. 시스템 값 QUSRLIBL에는 라이브러리 리스트의 사용자 부분이 될 라이브러리명이 들어 있습니다.

QSYSLIBL은 15개의 라이브러리명을 포함할 수 있으며 QUSRLIBL은 25개의 라이브러리명을 포함할 수 있습니다. 작업 라이브러리 리스트의 시스템 부분을 변경하려는 경우 CHGSYSLIBL(시스템 라이브러리 리스트 변경) 명령을 사용할 수 있습니다. QSYSLIBL이나 QUSRLIBL의 값을 변경하려면 CHGSYSVAL(시스템 값 변경) 명령을 사용하십시오. 이렇게 시스템 값을 변경하면 시스템 값이 변경된 후 시작되는 새로운 작업부터 영향을 받습니다.

라이브러리 리스트 변경

수행되고 있는 작업의 경우 ADDLIBLE(라이브러리 리스트 항목 추가) 명령을 사용하여 라이브러리 리스트에 항목을 추가하거나 RMVLIBLE(라이브러리 리스트 항목 제거) 명령을 사용하여 라이브러리 리스트에서 항목을 제거할 수 있으며, 또한 CHGLIBL 명령이나 EDTLIBL 명령을 사용하여 라이브러리 리스트에 있는 라이브러리를 변경할 수 있습니다. 이 명령들은 라이브러리 리스트의 시스템 부분이 아닌 사용자 부분을 변경합니다.

CHGCURLIB(현재 라이브러리 변경) 또는 CHGLIBL 명령을 사용하면 현재 라이브러리를 추가하거나 변경할 수 있습니다. 현재 라이브러리는 또한 사용자의 사용자 프로파일에서 사인 온 시 또는 SBMJOB(작업 제출) 명령에서도 변경될 수 있습니다. CL 명령을 사용해서는 제품 라이브러리를 추가할 수 없으며, 사용 중인 명령이나 메뉴가 수행될 때 시스템이 이들 라이브러리를 추가합니다. 제품 라이브러리는 CL 명령으로 변경시킬 수 없지만 라이브러리 리스트 변경(QLICHGLL) API를 사용하면 변경시킬 수 있습니다.

이 명령들을 사용할 때 라이브러리 리스트의 변경 내용은 명령이 수행되는 작업에만 영향을 미치며, 이 변경 내용은 작업의 라이브러리 리스트를 다시 변경할 때까지 또는 작업이 수행 중일 동안만 유효합니다. 이 명령을 사용하여 라이브러리 리스트를 변경할 경우 명령이 수행될 때 라이브러리가 존재해야 합니다. 사용자 작업의 라이브러리 리스트에 존재하는 라이브러리는 삭제할 수 없습니다. 라이브러리가 다른 작업의 라이브러리 리스트에 존재하는 경우에는 라이브러리 검색 리스트에서 시스템 값 QLIBLCKLVL이 라이브러리 잠금으로 설정된 경우에만 라이브러리 삭제가 제한됩니다.

작업이 시작될 때 라이브러리 리스트의 사용자 영역은 작업 설명에 들어 있는 값 또는 SBMJOB 명령에서 지정한 값에 의해 결정됩니다. *SYSVAL 값이 지정되면 시스템 값 QUSRLIBL에 의해 지정된 라이브러리가 라이브러리 리스트의 사용자 영역이 됩니다. 라이브러리명을 작업 설명과 일괄처리 작업(BCHJOB)이나 SBMJOB 명령 양쪽에 모두 지정하면 BCHJOB나 SBMJOB 명령에 지정한 라이브러리명이 작업 설명과 시스템 값 QUSRLIBL에 지정된 라이브러리를 모두 대체합니다.

다음은 QUSRLIBL에 지정된 라이브러리 리스트의 사용자 부분 순서가 각 작업의 명령에 의해 대체되는 것을 보여줍니다.

- 라이브러리 리스트는 작업이 수행될 때 QUSRLIBL에 지정된 라이브러리 리스트를 대체하는 작업 설명에서 지정됩니다. (작업 설명에 대한 자세한 정보는 iSeries Information Center의 **시스템 관리** 범주에서 *작업 관리* 주제를 참조하십시오.)
- BCHJOB 명령이나 SBMJOB 명령을 통해 작업이 제출되면 명령에서 라이브러리 리스트가 지정될 수 있습니다. 이 리스트는 작업 설명이나 시스템 값 QUSRLIBL에 지정된 라이브러리 리스트를 대체합니다.
- SBMJOB 명령을 사용하여 작업이 제출되면 라이브러리 리스트에 *CURRENT(디폴트 값)을 지정할 수 있습니다. *CURRENT는 SBMJOB 명령을 발행한 작업의 라이브러리 리스트가 사용됨을 나타냅니다.
- 작업에 있는 ADDLIBLE, RMVLIBLE 또는 CHGLIBL 명령을 사용할 수 있습니다. 이 명령들은 이전의 라이브러리 리스트 스펙을 대체합니다.
- 작업의 현재 라이브러리는 CHGCURLIB나 CHGLIBL 명령을 사용하여 변경할 수 있습니다.

라이브러리 리스트를 변경할 때마다 CHGLIBL 명령을 입력하는 대신 CL 프로그램에 이 명령을 넣어 사용할 수 있습니다.

```
PGM  /* SETLIBL - Set library list */
CHGLIBL LIBL(APPDEVLIB QGPL QTEMP)
    ENDPGM
```

이 라이브러리 리스트에 대해 정상적으로 작업할 경우 매번 프로그램을 호출하는 대신 초기 프로그램을 설정하여 라이브러리 리스트를 작성할 수 있습니다.

```
PGM  /* Initial program for QPGMR */
CHGLIBL LIBL(APPDEVLIB QGPL QTEMP)
TFRCTL PGM(QPGMMENU)
    ENDPGM
```

이 프로그램은 반드시 작성해야 하며, 이 프로그램이 적용될 사용자 프로파일은 새로운 초기 프로그램을 지정하도록 변경시켜야 합니다. 그러고 나면 이 프로그램으로부터 프로그래머 메뉴를 표시하는 QPGMMENU 프로그램으로 제어가 전송됩니다.

초기 프로그램에 지정된 라이브러리 리스트에 라이브러리를 추가해야 할 경우에는 ADDLIBLE 명령을 사용할 수 있습니다. 예를 들어, 다음의 명령은 라이브러리 JONES를 라이브러리 리스트의 끝에 추가시킵니다.

```
ADDLIBLE LIB(JONES) POSITION(*LAST)
```

작업의 일부에서 다른 라이브러리 리스트를 필요로 하는 경우 다음 프로그램과 같이 현재 라이브러리 리스트를 저장했다가 나중에 복원하는 CL 프로그램을 작성할 수 있습니다.

```
            PGM
       DCL &LIBL *CHAR 2750
       DCL &CMD *CHAR 2760
(1)    RTVJOBA USRLIBL(&LIBL)
(2)    CHGLIBL (QGPL QTEMP)
       .
       .
       .
(3)    CHGVAR &CMD ('CHGLIBL (' *CAT &LIBL *TCAT ')')
(4)    CALL QCMDEXC (&CMD 2760)
       .
       .
       .
    ENDPGM
```

(1) 라이브러리 리스트를 저장하는 명령. 라이브러리 리스트가 변수 &LIBL에 저장됩니다. 각 라이브러리명은 10바이트(필요한 경우 오른쪽이 공백으로 채워짐)를 차지하며 각 라이브러리명 사이에 하나의 공백이 들어갑니다.

(2) 이 명령은 다음 기능이 요구하는 대로 라이브러리 리스트를 변경합니다.

(3) CHGVAR(변수 변경) 명령은 변수 &CMD에 CHGLIBL 명령을 빌드합니다.

(4) QCMDEXC는 변수 &CMD에 있는 명령 스트링을 처리하기 위해 호출됩니다. CALL 명령에는 다른 명령이 연결될 수 없으므로 QCMDEXC를 호출하기 전에 CHGVAR 명령이 필요합니다.

라이브러리 리스트 설정 시 고려사항

라이브러리 리스트를 설정하여 사용할 때는 다음 사항을 고려해야 합니다.

- 라이브러리 리스트 안의 라이브러리는 시스템상에 반드시 존재해야 합니다. OS/400이 시작될 때 시스템 값 QSYSLIBL과 QUSRLIBL이 액세스됩니다. 이 값에 대한 라이브러리가 시스템상에 전혀 존재하지 않으면 메세지가 시스템 오퍼레이터의 메세지 대기행렬(QSYSOPR)로 송신되고, 라이브러리는 무시되며, OS/400은 라이브러리 없이 시작됩니다. 일단 OS/400이 시작되면 활동 작업의 라이브러리 리스트 중 어떠한 라이브러리도 삭제할 수 없습니다. 사용자의 작업 라이브러리 리스트에 있는 라이브러리는 삭제할 수 없습니다. 다른 작업의 경우 라이브러리 검색 리스트에서 시스템 값 QLIBLCKLVL이 라이브러리 잠금으로 설정되어 있으면 라이브러리 리스트에서 라이브러리를 삭제할 수 없습니다. 작업 설명이나 BCHJOB(일괄처리 작업) 또는 SBMJOB(작업 제출) 명령으로 지정된 라이브러리 리스트에 라이브러리가 없거나 사용할 수 없는 경우 작업이 시작되지 않습니다.
- 라이브러리 리스트 안의 라이브러리를 사용하려는 모든 사용자에게는 권한이 부여되어야 합니다. 라이브러리 리스트를 초기화하려면(예: SBMJOB[작업 제출], JOB[작업] 또는 CRTJOBD[작업 설명 작성] 명령에서), 라이브러리에 대한 오브젝트 조작 권한이 사용자에게 있어야 하며, 없는 경우에는 작업이 시작되지 않습니다. 사용자는 ADDLIBLE(라이브러리 리스트 항목 추가)나 CHGLIBL(라이브러리 리스트 변경) 명령을 반드시 사용하여 라이브러리 리스트에 추가된 라이브러리에 대해서도 *USE 권한이 있어야 합니다.

- 일시적으로 허용된 사용자 프로파일하에서 수행되고 있는 프로그램이 현재 사용자에게 권한이 없는 라이브러리 리스트에 라이브러리를 추가한 후 프로그램이 종료되기에 앞서 라이브러리 리스트에서 이 라이브러리를 제거하지 않으면 프로그램에서 나간 후에야 이 라이브러리에 액세스할 수 있습니다(*USE 권한). 이는 *LIBL이 오브젝트에 액세스하도록 지정될 때만 발생합니다.
- 라이브러리 리스트가 짧을수록 시스템 성능은 향상됩니다.

라이브러리 리스트 표시

DSPLIBL(라이브러리 리스트 표시) 명령을 사용하면 현재 수행 중인 작업의 라이브러리 리스트를 표시할 수 있습니다. 이 화면에는 라이브러리 리스트에 있는 순서대로 라이브러리 리스트 안의 라이브러리가 나열됩니다.

DSPJOB(작업 표시) 명령을 사용하고 작업 표시 메뉴에서 옵션 13을 선택하여 활동 중인 작업에 대해서도 라이브러리 리스트를 표시할 수 있습니다.

총칭 오브젝트명 사용

때로는 오브젝트명이 같은 문자로 시작할 때 여러 오브젝트를 탐색하려는 경우가 있습니다(하나만 발견될지라도). 이러한 유형의 탐색을 *총칭 탐색*이라고 하며, 이것을 여러 명령에 사용할 수 있습니다.

총칭 탐색을 사용하려면 명령에서 오브젝트명의 위치에 총칭명을 지정하십시오. 총칭명은 오브젝트 그룹을 식별하는 모든 오브젝트명에 공통으로 들어 있는 문자들로 이루어지고 별표(*)로 끝납니다. 지정된 문자로 이름이 시작되고 사용자에게 권한이 주어진 모든 오브젝트에 대해 요구된 기능이 수행됩니다. 예를 들면, 총칭명 ORD*를 사용하여 DSPOBJD(오브젝트 설명 표시) 명령을 입력하면 ORD로 시작하는 오브젝트에 대한 오브젝트 설명이 나옵니다.

총칭 탐색은 총칭명의 다음과 같은 라이브러리 규정자에 의해 제한될 수 있습니다. (가능하다면, 라이브러리명 매개변수 값은 괄호로 묶습니다.)

- 지정된 라이브러리. 사용자가 요구한 조작은 지정된 라이브러리 안에서 총칭적으로 명명된 오브젝트에 대해서만 수행됩니다.
- 작업에 대한 라이브러리 리스트(*LIBL). 라이브러리는 라이브러리 리스트에 나열하는 순서에 따라 탐색됩니다. 사용자가 요구한 조작은 작업의 라이브러리 리스트에 지정된 라이브러리 안에서 총칭적으로 명명된 오브젝트에 대해 수행됩니다.
- 작업의 현재 라이브러리(*CURLIB). 작업의 현재 라이브러리가 탐색됩니다. 현재 라이브러리가 없으면 QGPL이 사용됩니다.
- 작업의 라이브러리 리스트 가운데 사용자 부분(*USRLIBL) 안에 있는 모든 라이브러리. 라이브러리는 현재 라이브러리(*CURLIB)를 포함하여 라이브러리 리스트에 나

열된 순서에 따라 탐색됩니다. 사용자가 요구한 조작은 작업의 라이브러리 리스트 가운데 사용자 영역에 지정된 라이브러리에서 총칭적으로 명명된 오브젝트에 대해 수행됩니다.

- 권한이 부여된 모든 사용자 라이브러리(*ALLUSR)와 **iSeries Information Center**의 **프로그래밍** 범주에 있는 *API* 섹션의 일반 라이브러리명에 나열된 것처럼 Q자로 시작되는 라이브러리.
- 라이브러리는 영숫자순으로 탐색됩니다. #으로 시작되는 #CGULIB, #COBLIB, #DFULIB, #DSULIB, #RPGLIB, #SDALIB, #SEULIB 등의 S/36 환경 라이브러리는 *ALLUSR을 지정해도 탐색되지 않습니다. 사용자가 요구한 조작은 사용자에게 권한이 부여된 모든 사용자 라이브러리에서 총칭적으로 명명된 오브젝트에 대해 수행됩니다.
- 권한 부여된 시스템상의 모든 라이브러리(*ALL). 라이브러리는 영숫자순으로 탐색됩니다. 사용자가 요구한 조작은 사용자에게 권한이 부여된 시스템상의 모든 라이브러리에서 총칭적으로 명명된 오브젝트에 대해 수행됩니다.

IBM에서는 총칭 기능을 사용하는 연산에 관한 정보를 제공합니다. **iSeries Information Center**의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

복수 오브젝트 또는 단일 오브젝트의 탐색

총칭명을 지정할 수 있는 모든 명령 안에 오브젝트명(별표는 지정하지 못함)을 지정할 수 있으며 여러 개의 오브젝트를 탐색할 수도 있습니다. 오브젝트명을 지정하고 라이브러리명으로 *ALL이나 *ALLUSR을 지정하면 시스템이 복수 오브젝트를 탐색하여 사용자에게 권한이 부여된 지정 이름과 유형의 오브젝트를 리턴시킵니다. 총칭명을 지정하거나 *ALL, *ALLUSR 또는 오브젝트명과 함께 라이브러리를 지정하는 경우 지원되는 모든 오브젝트 유형(또는 *ALL 오브젝트 유형)을 지정할 수 있습니다.

라이브러리 사용

라이브러리는 관련 오브젝트들을 그룹으로 묶고 이 오브젝트들을 이름으로 찾을 때 사용되는 오브젝트입니다. 따라서 라이브러리는 오브젝트 그룹의 디렉토리가 될 수 있습니다.

라이브러리는 다음 목적으로 사용될 수 있습니다.

- 각 사용자의 특정 오브젝트를 그룹화하기 위해. 이것은 시스템에서 오브젝트를 관리하는 데 도움이 되는 것입니다. 예를 들면, 사용자 JOE가 사용하는 모든 파일을 라이브러리 JOELIB에 넣을 수 있습니다.
- 각 어플리케이션에 사용되는 모든 오브젝트를 그룹화하기 위해. 예를 들면, 모든 주문 입력 파일과 프로그램을 주문 입력 라이브러리 DISTLIB에 넣을 수 있습니다. 모든 주문 입력 파일과 프로그램이 리스트 안에 있는지 확인하려면 라이브러리 리스트

에 하나의 라이브러리를 추가하기만 하면 됩니다. 이는 주문 입력 파일이나 프로그램을 사용할 때마다 라이브러리명을 지정하고 싶지 않을 때 유용합니다.

- 보안을 보장하기 위해. 예를 들면, 라이브러리를 사용할 권한이 있는 사용자를 지정할 수 있고, 사용자가 라이브러리에 수행할 수 있는 작업을 지정할 수 있습니다.
- 라이브러리의 CRTAUT 매개변수 값에 근거하여 새로 작성된 오브젝트에 대해, 자동으로 권한 부여 리스트를 부여하고 공용 권한을 할당하여 보안을 단순화하기 위해. 새로 작성된 오브젝트에 대한 감사 속성은 CRTOBJAUD(오브젝트 감사 작성) 매개변수 값에 기초하여 설정될 수 있습니다.
- 동일한 라이브러리에서 동시에 저장 및 복원되는 오브젝트를 그룹화하여 저장/복원 작업을 단순화하기 위해. SAVOBJ(오브젝트 저장) 명령을 사용하여 오브젝트를 개별적으로 저장하는 대신 SAVLIB(라이브러리 저장) 명령을 사용할 수 있습니다.
- 테스트용으로 복수 라이브러리를 사용하기 위해. 보다 자세히 알려면 444 페이지의 『OPM 프로그램 디버깅』을 참조하십시오.
- 복수 제품 라이브러리를 사용하기 위해. 예를 들면, 소스 파일과 오브젝트 작성, 어플리케이션 프로그램과 파일, 가끔 저장되는 오브젝트 및 자주 저장되는 오브젝트에 대한 각각의 제품 라이브러리를 사용할 수 있습니다.

복수 라이브러리는 오브젝트를 보다 쉽게 사용할 수 있도록 만듭니다. 예를 들면, 이름은 동일하나 다른 라이브러리에 들어 있는 두 개의 파일을, 하나는 테스트용으로 다른 하나는 정상적인 처리에 사용할 수 있습니다. 라이브러리명을 프로그램에 지정하지 않는 한, 프로그램상의 파일명은 테스트나 정상적인 처리를 위해 따로 변경할 필요가 없습니다. 사용자는 라이브러리 리스트를 통해 어느 라이브러리를 사용할 것인지 제어합니다. (유형이 같은 오브젝트들은 서로 다른 라이브러리에 들어 있을 때만 같은 이름을 가질 수 있습니다.)

라이브러리에는 테스트 라이브러리와 제품 라이브러리의 두 가지 유형이 있습니다. 제품 라이브러리는 정상적인 처리를 위한 것입니다. 디버그 모드에서는 제품 라이브러리에 들어 있는 데이터베이스 파일이 갱신되는 것을 막을 수 있습니다. 디버그 모드에서 테스트 라이브러리의 파일은 고유 스펙 없이 갱신될 수 있습니다. (테스트 라이브러리 사용에 대해 자세히 알려면 444 페이지의 『OPM 프로그램 디버깅』을 참조하십시오.)

라이브러리 작성

라이브러리를 작성하려면 CRTLIB(라이브러리 작성) 명령을 사용하십시오. 한 예로, 다음의 CRTLIB 명령은 주문 입력 파일과 프로그램이 들어 있는 라이브러리를 작성합니다. 라이브러리의 이름은 DISTLIB이고 유형은 제품 라이브러리(production library)입니다. 공용으로 주어진 디폴트 권한은 사용자가 라이브러리에 액세스하는 것을 방지합니다. 라이브러리 안에 작성된 오브젝트는 CRTAUT 값에 근거하여 *CHANGE라는 디폴트 공용 권한을 부여받습니다.

```
CRTLIB   LIB(DISTLIB) TYPE(*PROD) CRTAUT(*CHANGE) CRTOBJAUD(*USRPRF) +
         ASP(1) ASPDEV(*ASP) AUT(*EXCLUDE) TEXT('Distribution library')
```

이름이 Q로 시작하는 라이브러리를 작성해서는 안됩니다. 총칭 탐색 시, 시스템은 문자 Q로 시작하는 이름의 라이브러리(QRPG 또는 QPDA) 가운데 거의 대부분을 시스템 라이브러리라고 가정합니다. 자세히 알려면 127 페이지의 『총칭 오브젝트명 사용』 부분을 참조하십시오.

라이브러리에 대한 권한 지정

다음은 라이브러리에 대해서 사용자에게 부여될 수 있는 각각의 권한을 설명한 것입니다. 자세한 정보는 보안 - 참조서  책을 참조하십시오.

오브젝트 권한

라이브러리의 **오브젝트 조작 권한**은 사용자에게 라이브러리의 설명을 표시할 수 있는 권한을 부여합니다.

라이브러리의 **오브젝트 관리 권한**은 다음 작업을 할 수 있는 권한을 포함하고 있습니다.

- 권한의 허용(grant)과 취소(revoke). 사용자는 자신이 가지고 있는 권한만 부여 및 취소할 수 있습니다. 오브젝트 소유자나 *ALLOBJ 권한을 가진 사용자만 라이브러리에 대해 오브젝트 관리 권한을 부여할 수 있습니다.
- 라이브러리의 이름 변경

오브젝트 존재 권한과 **사용 권한**은 사용자에게 라이브러리를 삭제할 수 있는 권한을 부여합니다.

오브젝트 존재 권한과 **오브젝트 조작 권한**은 사용자에게 라이브러리의 소유권을 전송할 수 있는 권한을 부여합니다.

자료 권한

라이브러리의 **추가 권한**과 **읽기 권한**은 사용자로 하여금 라이브러리에 신규 오브젝트를 작성하거나 오브젝트를 라이브러리 안으로 이동시킬 수 있게 합니다.

라이브러리의 **갱신 권한**과 **실행 권한**은 사용자가 오브젝트에 대한 권한을 부여받은 경우 사용자로 하여금 라이브러리에 있는 오브젝트명을 변경할 수 있게 합니다.

삭제 권한은 사용자로 하여금 오브젝트에서 항목을 제거할 수 있게 합니다. 라이브러리에 대한 삭제 권한으로는 라이브러리에서 오브젝트를 삭제할 수 없습니다. 라이브러리 안의 오브젝트에 대한 권한은 오브젝트가 삭제될 수 있는지 판별하는 데 사용됩니다.

실행 권한은 사용자로 하여금 오브젝트의 라이브러리를 탐색할 수 있게 합니다.

조합 권한

라이브러리의 ***USE 권한**(오브젝트 조작 권한, 읽기 권한 및 실행 권한으로 구성됨)은 다음 작업을 할 수 있는 권한을 포함하고 있습니다.

- 오브젝트를 찾기 위해 라이브러리를 사용합니다.
- 라이브러리 내용을 표시합니다.
- 라이브러리를 라이브러리 리스트에 놓습니다.
- 라이브러리를 저장합니다(오브젝트에 대해 충분한 권한이 있을 경우).
- 라이브러리에서 오브젝트를 삭제합니다(사용자에게 라이브러리 안의 오브젝트에 대해 권한이 부여된 경우).

라이브러리의 ***CHANGE 권한**(오브젝트 조작 권한과 라이브러리에 대한 모든 자료 권한으로 구성됨)은 다음 작업을 할 수 있는 권한을 포함하고 있습니다.

- 오브젝트를 찾기 위해 라이브러리를 사용합니다.
- 라이브러리 내용을 표시합니다.
- 라이브러리를 라이브러리 리스트에 놓습니다.
- 라이브러리를 저장합니다(오브젝트에 대해 충분한 권한이 있을 경우).
- 라이브러리에서 오브젝트를 삭제합니다(사용자에게 라이브러리 안의 오브젝트에 대해 권한이 부여된 경우).
- 라이브러리에 오브젝트를 추가합니다.

***ALL 권한**은 모든 오브젝트 권한과 자료 권한을 제공합니다. 사용자는 라이브러리를 삭제할 수 있고, 라이브러리에 대한 보안을 지정할 수 있으며, 라이브러리를 변경할 수 있고, 라이브러리의 설명과 내용을 표시할 수 있습니다.

***EXCLUDE 권한**은 사용자가 오브젝트에 액세스할 수 없게 합니다.

라이브러리와 연관된 권한을 표시하기 위해 DSPOBJAUT(오브젝트 권한 표시) 명령을 사용할 수도 있습니다.

오브젝트에 대한 보안 고려사항

시스템은 사용자가 참조하는 오브젝트에 액세스할 때 그 사용자가 오브젝트를 사용할 수 있는 권한과 사용자가 요구한 방식으로 오브젝트를 사용할 수 있는 권한을 부여받았는지를 검사하여 판별합니다. 일반적으로, 사용자는 두 가지 레벨에서 권한이 있어야 합니다.

- 사용자가 요구한 기능이 수행될 오브젝트에 대해 권한이 있어야 합니다.
- 오브젝트가 들어 있는 라이브러리에 대해 권한이 있어야 합니다. 라이브러리 리스트가 사용되면 리스트 안의 라이브러리에 대해 반드시 권한이 있어야 합니다.

오브젝트 권한은 다음과 같이 시스템의 보안 기능에 의해 제어됩니다.

- 오브젝트 소유자와 *ALLOBJ 특수 권한을 가진 오브젝트 사용자는 오브젝트에 대한 모든 권한을 가지며 다른 사용자에게 권한을 부여하거나 다른 사용자로부터 권한을 취소시킬 수 있습니다.

- 사용자에게 오브젝트에 대한 개인 권한이 부여되지 않을 경우 사용자는 공용 권한을 가집니다.

보안 - 참조서  책에는 오브젝트에 부여받을 수 있는 권한 유형과 그 오브젝트에서 기능을 수행하는 데 필요한 권한이 자세히 설명되어 있습니다. 라이브러리에 부여할 수 있는 권한은 130 페이지의 『라이브러리에 대한 권한 지정』 부분에 그 설명이 나옵니다.

보안이 필요한 프로그램(예: 보안 담당자의 사용자 프로파일을 채택하는 프로그램)을 작성할 때에는 고려해야 할 점이 있습니다. 보안 - 참조서  책을 참조하십시오.

DSPAUDJRNE(감사 저널 항목 표시) 명령

DSPAUDJRNE(감사 저널 항목 표시) 명령은 보안 저널 감사 보고서를 생성할 수 있도록 해줍니다. 보고서는 명령에 지정된 감사 항목 유형과 사용자 프로파일에 기초하고 있습니다. 특정 시간대로 보고서를 제한하고 접속이 해제된 저널 리시버를 탐색할 수 있습니다. 그리고 이 보고서를 활동 중인 화면이나 출력 대기행렬로 보낼 수 있습니다.

제한사항: 이 명령을 사용하려면 *ALLOBJ 및 *AUDIT 특수 권한이 있어야 합니다.

이 명령에 대한 매개변수와 값의 설명을 보려면 온라인 도움말을 참조하십시오.

새로 작성된 오브젝트에 대한 디폴트 공용 권한

라이브러리에서 오브젝트가 작성될 때 오브젝트의 공용 권한은 라이브러리의 CRTAUT 값을 사용하여 디폴트로 설정됩니다.

다음 예를 보십시오.

```
CRTLIB LIB(TESTLIB) CRTAUT(*USE) AUT(*LIBCRTAUT)
```

이 경우 라이브러리 TESTLIB가 작성됩니다. 라이브러리 TESTLIB로 작성되는 모든 오브젝트에는 디폴트로 *USE라는 공용 권한이 있습니다. 라이브러리 TESTLIB에 대한 공용 권한이 라이브러리 QSYS의 CRTAUT 값으로 결정됩니다.

다음 예를 보십시오.

```
CRTDTAARA DTAARA(TESTLIB/DTA1) TYPE(*CHAR) +
      AUT(*LIBCRTAUT)

CRTDTAARA DTAARA(TESTLIB/DTA2) TYPE(*CHAR) +
        AUT(*EXCLUDE)
```

이 경우 자료 영역 DTA1이 라이브러리 TESTLIB에 작성됩니다. DTA1의 공용 권한은 라이브러리 TESTLIB의 CRTAUT 값에 기초한 *USE입니다.

자료 영역 DTA2가 라이브러리 TESTLIB에 작성됩니다. DTA2의 공용 권한은 *EXCLUDE입니다. *EXCLUDE는 CRTDTAARA(자료 영역 작성) 명령의 AUT 매개변수에 지정됩니다.

권한 부여 리스트는 오브젝트가 라이브러리에 작성되는 경우 오브젝트 보안을 위해서도 사용할 수 있습니다.

다음 예를 보십시오.

```
CRTAUTL AUTL(PAYROLL)
CRTLIB  LIB(PAYLIB) CRTAUT(PAYROLL) +
        AUT(*EXCLUDE)
```

이 경우 이름이 PAYROLL인 권한 부여 리스트가 작성됩니다. 라이브러리 PAYLIB는 *EXCLUDE 공용 권한으로 작성됩니다. 디폴트로, 라이브러리 PAYLIB 안에 작성되는 오브젝트는 권한 부여 리스트 PAYROLL에 의해 보안됩니다.

다음 예를 보십시오.

```
CRTPF FILE(PAYLIB/PAYFILE) +
      AUT(*LIBCRTAUT)

CRTPF FILE(PAYLIB/PAYACC) +
      AUT(*CHANGE)
```

이 경우 파일 PAYFILE이 라이브러리 PAYLIB에 작성됩니다. 파일 PAYFILE은 권한 부여 리스트 PAYROLL에 의해 보안됩니다. 파일 PAYFILE의 공용 권한이 CRTPF(실제 파일 작성) 명령의 일부로 *AUTL에 설정됩니다. *AUTL은 파일 PAYFILE에 대한 공용 권한이 파일 PAYFILE을 보안하는 권한 부여 리스트 PAYROLL로부터 가져온 것임을 나타냅니다.

파일 PAYACC는 라이브러리 PAYLIB에 작성됩니다. 파일 PAYACC에 대한 공용 권한은 CRTPF 명령의 AUT 매개변수에 지정된 *CHANGE입니다.

주: 대부분의 CRT 명령에 존재하는 AUT 매개변수의 *LIBCRTAUT 값은 오브젝트에 대한 공용 권한이 오브젝트가 작성되고 있는 라이브러리의 CRTAUT 값으로 설정됨을 나타냅니다.

라이브러리의 CRTAUT 값은 라이브러리에 작성된 오브젝트의 공용 사용에 대한 디폴트 권한을 지정합니다. 가능한 값은 아래와 같습니다.

***SYSVAL**
작성 중인 오브젝트에 대한 공용 권한은 시스템 값 QCRTAUT에 지정된 값입니다.

***ALL** 모든 공용 권한

***CHANGE**
변경 권한

***USE** 사용 권한

***EXCLUDE**
제외 권한

권한 부여 리스트명
권한 부여 리스트가 오브젝트를 보안합니다.

새로 작성된 오브젝트에 대한 디폴트 감사 속성

라이브러리에서 오브젝트가 작성될 때 오브젝트의 감사 속성은 라이브러리의 CRTOBJAUD 값을 사용하여 디폴트로 설정됩니다.

다음 예를 보십시오.

```
CRTLIB LIB(PAYROLL) AUT(*EXCLUDE) CRTAUT(*EXCLUDE) CRTOBJAUD(*ALL)
```

이 경우 급여 라이브러리(payroll library) 안에 작성된 모든 오브젝트가 읽기 및 변경 액세스에 대해 감사됩니다. 감사에 관한 자세한 정보는 보안 - 참조서  책을 참조하십시오.

라이브러리에 오브젝트 놓기

오브젝트를 작성하면 오브젝트가 라이브러리 안에 놓입니다. 라이브러리를 지정하지 않으면 오브젝트는 작업의 현재 라이브러리(*CURLIB)에 놓이거나, 작업의 현재 라이브러리가 없는 경우에는 QGPL에 놓입니다. 라이브러리가 작성될 때 사용자는 CRTLIB(라이브러리 작성) 명령에 CRTAUT 매개변수를 사용하여 라이브러리에서 작성되는 오브젝트에 대해 공용 권한을 지정할 수 있습니다. 해당 라이브러리에 놓인 모든 오브젝트는 라이브러리의 CRTAUT 값에 지정된 공용 권한을 가정합니다. 라이브러리를 지정하려면 규정된 이름(라이브러리명과 오브젝트명)을 지정하십시오. 한 예로, 다음의 CRTPF(실제 파일 작성) 명령은 DISTLIB에 위치하는 주문 입력 실제 파일 ORDHDRP를 작성합니다.

```
CRTPF FILE(DISTLIB/ORDHDRP)
```

오브젝트를 라이브러리에 놓으려면 라이브러리에 대한 읽기 및 추가 권한이 있어야 합니다.

같은 유형을 가진 둘 이상의 오브젝트가 같은 라이브러리에서 이름이 같을 수는 없습니다. 예를 들어, 이름이 ORDHDRP인 두 개의 파일은 라이브러리 DISTLIB에 함께 들어 있을 수 없습니다. 라이브러리 안의 기존 오브젝트와 이름과 유형이 같은 오브젝트를 라이브러리에 넣으려고 시도하면 시스템은 요구를 거절하고 사용자에게 그 이유 메세지를 송신합니다.

주: QSYS 라이브러리는 시스템 오브젝트에 대해서만 사용하십시오. OS/400의 새로운 릴리스를 설치할 때 변경 내용이 유실되므로 기타 사용권 프로그램을 QSYS 라이브러리에 복원하지 마십시오.

라이브러리 삭제 및 지우기

DLTLIB(라이브러리 삭제) 명령으로 라이브러리를 삭제하면 라이브러리 자체뿐만 아니라 라이브러리 안에 있는 오브젝트도 삭제됩니다. CLRLIB(라이브러리 지우기) 명령으로 라이브러리를 지우면, 라이브러리가 삭제되지 않고 라이브러리 안의 오브젝트만 삭제됩니다. 라이브러리를 삭제하거나 지울 때는 라이브러리명을 지정해야 합니다. 예를 들어, 다음과 같은 경우가 있습니다.

```
DLTLIB LIB(DISTLIB)
```

또는

```
CLRLIB LIB(DISTLIB)
```

라이브러리를 삭제하려면 라이브러리와 라이브러리 안의 오브젝트에 대한 오브젝트 존재 권한과 라이브러리에 대한 사용 권한이 있어야 합니다. 라이브러리 안의 모든 오브젝트에 대해 오브젝트 존재 권한이 없으면서 라이브러리를 삭제하려 하면 권한이 없는 라이브러리와 모든 오브젝트는 삭제되지 않습니다. 권한이 있는 오브젝트는 모두 삭제됩니다. 라이브러리에 대한 오브젝트 존재 권한이 없으면서 라이브러리를 삭제하려 하면 라이브러리뿐만 아니라 라이브러리 안의 어떤 오브젝트도 삭제되지 않습니다. 오브젝트 존재 권한을 가진 특정 오브젝트를 삭제하려면 DLTPGM(프로그램 삭제) 명령과 같이 그 오브젝트 유형에 대한 삭제 명령을 사용하면 됩니다.

활동 중인 작업의 라이브러리 리스트에서는 라이브러리를 삭제할 수 없습니다. 작업이 종료되어 라이브러리의 삭제가 허용될 때까지 기다려야 합니다. 그렇기 때문에 다음 라우팅 단계가 시작되기 전에 라이브러리를 삭제해야 합니다. 라이브러리를 삭제할 때는 다른 사용자가 라이브러리나 라이브러리 안의 어떤 오브젝트도 필요로 하지 않는지 확인하십시오.

라이브러리가 시스템 값 QSYSLIBL과 QUSRLIBL에 의해 정의된 초기 라이브러리 리스트의 일부인 경우 라이브러리를 삭제할 때는 다음 단계를 따르도록 하십시오.

1. CHGSYSVAL(시스템 값 변경) 명령을 사용하여 라이브러리가 들어 있는 시스템 값에서 라이브러리를 제거하십시오. (변경된 시스템 값은 수행 중인 작업의 라이브러리 리스트에 영향을 주지 않습니다.)
2. CHGLIBL(라이브러리 리스트 변경) 명령을 사용하여 작업의 라이브러리 리스트를 변경하십시오.

   CHGSYSLIBL(시스템 라이브러리 리스트 변경), ADDLIBLE(라이브러리 리스트 항목 추가), EDTLIBL(라이브러리 리스트 편집) 및 RMVLIBLE(라이브러리 리스트 항목 제거) 명령도 라이브러리 리스트를 변경하는 데 사용됩니다.

3. DLTLIB 명령을 사용하여 라이브러리와 라이브러리 안의 오브젝트를 삭제하십시오.

주: 라이브러리 QSYS는 삭제할 수 없으며, 그 안의 오브젝트도 삭제해서는 안됩니다. 시스템이 적절히 작동하기 위해서는 QSYS 안의 오브젝트가 필요하므로 삭제한다면 시스템을 종료시키는 결과를 초래할 수 있습니다. 라이브러리 QGPL도 삭제해서는 안 되며, 그 이유는 QGPL에 시스템을 효율적으로 수행하는 데 필요한 오브젝트가 들어 있기 때문입니다. 라이브러리 QRECOVERY는 시스템만이 사용하도록 작성되어 있기 때문에 사용자는 사용할 수 없습니다. 라이브러리 QRECOVERY에는 시스템이 제대로 작동하는 데 필요한 오브젝트가 들어 있습니다.

라이브러리 이외의 오브젝트 삭제에 관한 사항은 163 페이지의 『오브젝트 삭제』 부분을 참조하십시오.

라이브러리를 지우려면 라이브러리 안의 오브젝트에 대한 오브젝트 존재 권한과 라이브러리에 대한 사용 권한이 있어야 합니다. 라이브러리 안의 모든 오브젝트에 대해 오브젝트 존재 권한이 없으면서 라이브러리를 지우려 하면 권한이 없는 오브젝트는 라이브러리에서 삭제되지 않습니다. 오브젝트가 다른 사용자에게 할당된 경우에도 삭제되지 않습니다.

라이브러리명과 내용 표시

DSPLIB(라이브러리 표시) 또는 WRKLIB(라이브러리에 대한 작업) 명령을 사용하면 권한을 가진 모든 라이브러리를 표시하거나 인쇄하고 라이브러리 안의 각 오브젝트에 대한 기본 정보를 찾을 수 있습니다.

오브젝트 정보는 다음과 같습니다.

- 오브젝트 이름과 유형
- 오브젝트 속성
- 오브젝트 크기
- 오브젝트가 작성되었을 때 오브젝트에 대해 입력된 설명

DSPLIB 명령에서 특정 라이브러리명을 지정할 수 있으며 이 경우 라이브러리 선택 화면이 바이패스됩니다. 이 리스트에서 오브젝트는 라이브러리별로 그룹화되고, 각 라이브러리에서는 오브젝트 유형별로 그룹화되며, 각 유형에서는 영숫자순으로 나열됩니다. 라이브러리의 순서는 다음 중 하나로 결정됩니다.

- 라이브러리가 DSPLIB 명령에 지정된 경우 라이브러리는 표시 명령에 지정된 순서대로 나옵니다.
- *LIBL 또는 *USRLIBL이 DSPLIB 명령에 지정된 경우 라이브러리의 순서는 작업의 라이브러리 리스트에 있는 라이브러리의 순서와 일치합니다.

- *ALL 또는 *ALLUSR이 DSPLIB 명령에 지정된 경우 라이브러리의 순서는 영숫자순입니다. 사용자에게는 표시될 라이브러리에 대한 읽기 권한이 반드시 있어야 합니다.

예를 들면, 다음의 DSPLIB 명령은 DISTLIB에 포함된 오브젝트 리스트를 표시합니다.

```
DSPLIB  LIB(DISTLIB)  OUTPUT(*)
```

OUTPUT 매개변수의 별표(*)는 라이브러리가 대화식 처리의 경우 표시장치에 표시됨을 의미하고, 일괄처리의 경우 인쇄됨을 의미합니다. 대화식 처리인 경우 리스트를 인쇄하려면 디폴트 * 대신 *PRINT를 지정해야 합니다.

DSPLIB 명령에 대한 자세한 정보 및 샘플 화면에 대해서는 **iSeries Information Center**의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

라이브러리 설명 표시 및 검색

DSPLIBD(라이브러리 설명 표시)와 RTVLIBD(라이브러리 설명 검색) 명령을 사용하면 라이브러리에 대한 설명을 표시하고 검색할 수 있습니다.

라이브러리 설명 정보는 다음과 같습니다.

- 라이브러리 유형(PROD 또는 TEST)
- 라이브러리의 보조 기억장치 풀 번호
- 라이브러리의 보조 기억장치 풀 이름
- 라이브러리 작성 권한
- 라이브러리의 오브젝트 감사 작성
- 라이브러리 텍스트 설명

OS/400 국제화

OS/400 사용권 프로그램은 동일한 시스템에서 다른 자국어를 지원합니다. 따라서 한 국가의 언어로 된 정보가 한 사용자에게 표시되는 동시에 다른 국가의 언어로 된 정보가 또 다른 사용자에게 표시될 수 있습니다.

사용자가 읽을 수 있는 정보(화면, 메세지, 인쇄 출력 및 온라인 도움말 정보)에 사용되는 언어는 작업에 대한 라이브러리 리스트에 의해서 제어됩니다. 자국어 라이브러리를 라이브러리 리스트의 시스템 부분에 추가시키면 다른 자국어 버전의 정보도 표시할 수 있습니다. 1차 언어의 경우 **자국어 버전**은 각 사용권 프로그램에 입력된 실행 코드와 텍스트 자료입니다. 2차 언어의 경우에 자국어 버전은 모든 사용권 프로그램에 대한 텍스트 자료입니다.

시스템의 1차 언어에 대한 언어 정보는 IBM 사용권 프로그램용 프로그램과 같은 라이브러리에 저장됩니다. 예를 들면, 시스템의 1차 자국어가 영어인 경우 QSYS,

QHLPSYS, QSSP 라이브러리 등에는 영어로 된 정보가 들어 있습니다. 라이브러리 QSYS와 QHLPSYS는 라이브러리 리스트의 시스템 부분에 있습니다. 기타 사용권 프로그램(예: QRPGLE for ILE RPG for OS/400*)에 대한 라이브러리는 필요할 때 시스템이 라이브러리 리스트에 추가합니다.

시스템의 1차 언어를 제외한 다른 자국어 버전은 2차 자국어 라이브러리에 설치됩니다. 각 2차 언어 라이브러리에는 모든 IBM 사용권 프로그램을 위한 화면, 메세지, 명령 프롬트, 도움말의 단일 자국어 버전이 들어 있습니다. 2차 언어 라이브러리명의 양식은 QSYSnnnn이며, 여기서 nnnn은 언어 피처 코드입니다. 예를 들면, 불어의 피처 코드가 2928이므로 불어에 대한 2차 자국어 라이브러리명은 QSYS2928입니다.

시스템의 1차 자국어로 표시된 정보를 원할 경우 특별한 조치가 필요없습니다. 시스템의 1차 자국어와 다른 자국어로 정보를 표시하려면 라이브러리 리스트를 변경하여 원하는 자국어 라이브러리를 자국어 정보가 들어 있는 라이브러리 리스트의 다른 모든 라이브러리 앞에 위치시켜야 합니다. 다음 옵션 중 어느 것이나 사용하여 원하는 자국어 라이브러리를 맨 앞에 위치시킬 수 있습니다.

- CRTSBSD 또는 CHGSBSD에서 SYSLIBLE 매개변수를 사용하여 특정 언어에 대한 화면, 메세지 등을 표시할 수 있습니다. 예를 들어, 다음과 같은 경우가 있습니다.

  ```
  CRTSBSD SBSD(QSBSD 2928) POOLS((1 *NOTSG)) SYSLIBLE(QSYS2928)
  ```

- CHGSYSLIBL 명령상의 LIB 매개변수를 사용하여 라이브러리 리스트의 맨 위에 원하는 자국어 라이브러리를 지정할 수 있습니다. 예를 들어, 다음과 같은 경우가 있습니다.

  ```
  CHGSYSLIBL  LIB(QSYS2928)
  ```

- 사용자 프로파일에 초기 프로그램을 설정하여 대화식 작업에 대한 라이브러리 리스트의 맨 위에 원하는 자국어 라이브러리를 지정할 수 있습니다. 사용자가 시스템을 사인 온할 때마다 CHGSYSLIBL 명령이 실행되지 않도록 할 경우에 이 옵션을 사용하면 좋습니다. 초기 프로그램은 CHGSYSLIBL(시스템 라이브러리 리스트 변경) 명령을 사용하여 원하는 자국어 라이브러리를 라이브러리 리스트의 맨 위에 추가시킵니다.

  주: CHGSYSLIBL 명령과 함께 제공된 권한으로 모든 사용자가 이 명령을 수행할 수 있는 것은 아닙니다.

CHGSYSLIBL 명령에 대한 사용자 권한을 부여하지 않고 이 명령을 수행하려면 CHGSYSLIBL 명령이 들어 있는 CL 프로그램을 작성하면 됩니다. 이 프로그램은 보안 담당자(security officer)가 소유하며 프로그램 작성 시 보안 담당자의 권한을 채택합니다. 프로그램 수행 권한을 가진 모든 사용자는 이 명령을 사용하여 사용자 작업의 라이브러리 리스트에서 시스템 부분을 변경할 수 있습니다. 다음은 불어 사용자의 라이브러리 리스트를 설정하는 프로그램의 예입니다.

```
      PGM
CHGSYSLIBL  LIB(QSYS2928)  /* Use French information */
      ENDPGM
```

오브젝트 설명

작성 명령을 사용하여 오브젝트를 작성할 때마다 작성 명령 가운데 TEXT 매개변수의 50자 필드에 오브젝트를 설명할 수 있습니다. 어떤 명령들은 작성될 오브젝트에 대한 텍스트를 나타내는 디폴트 *SRCMBRTXT를 오브젝트가 작성될 소스 멤버의 텍스트로부터 나오게 만듭니다. 이 디폴트는 데이터베이스 소스 파일 안의 소스로부터 작성되는 오브젝트에 대해서만 유효합니다.

작성 명령에 대한 소스 입력이 장치 또는 인라인 파일이거나 소스가 사용되지 않으면 디폴트 값은 공백입니다. 이 텍스트가 오브젝트 설명의 일부가 되며, DSPOBJD(오브젝트 설명 표시) 또는 DSPLIB(라이브러리 표시) 명령을 사용하여 표시될 수 있습니다. 텍스트는 CHGOBJD(오브젝트 설명 변경) 명령 또는 각각의 오브젝트 유형별로 다양한 CHGxxx(변경) 명령을 사용하여 변경될 수 있습니다.

오브젝트 설명 표시

DSPOBJD(오브젝트 설명 표시) 또는 WRKOBJ(오브젝트에 대한 작업) 명령을 사용하면 오브젝트에 대한 설명을 표시할 수 있습니다. 이 설명은 오브젝트가 사용되고 있지는 않으나 시스템에 오브젝트가 존재하는지를 판별하는 데 도움을 줍니다. 사용자가 일괄처리를 사용하고 있으면, 설명이 인쇄되거나 데이터베이스 파일에 기록될 수 있습니다. 대화식 처리를 사용하고 있으면, 설명이 화면에 표시되거나 인쇄 또는 데이터베이스 파일에 기록될 수 있습니다.

사용자는 오브젝트 설명에 대한 기본 속성이나 모든 속성 또는 서비스 속성을 표시할 수 있습니다. 오브젝트 설명은 다음과 같습니다.

표 3. 오브젝트 설명에 대해 표시되는 속성

기본 속성	모든 속성	서비스 속성(주 참조)
• 오브젝트명	• 오브젝트명	• 오브젝트명
• 라이브러리명	• 라이브러리명	• 라이브러리명
• 라이브러리 ASP 장치	• 라이브러리 ASP 장치	• 라이브러리 ASP 장치
• 오브젝트 유형	• 오브젝트 유형	• 오브젝트 유형
• 확장 속성	• 소유자	• 소스 파일과 라이브러리
• 오브젝트 크기	• 1차 그룹	• 멤버명
• 텍스트 설명(부분)	• 확장 속성	• 확장 속성
	• 사용자 정의 속성	• 사용자 정의 속성
	• 텍스트 설명	• 해제 상태(freed status)
	• 작성 날짜와 시간	• 오브젝트 크기
	• 오브젝트를 작성한 사용자	• 작성 날짜와 시간
	• 작성된 시스템 오브젝트	• 소스 파일 멤버가 최종 갱신된 날짜와 시간
	• 오브젝트 정의역	• 시스템 레벨
	• 변경 날짜와 시간	• 컴파일러
	• 사용 자료의 수집 여부	• 오브젝트 제어 레벨
	• 최종 사용 날짜	• 프로그램에 의한 변경
	• 사용일 날짜 계수	• 사용자에 의한 변경 여부
	• 사용일 날짜 계수 재설정 날짜	• 사용권 프로그램
	• 프로그램에 의한 변경 허용	• PTF 번호
	• 오브젝트 감사 값	

표 3. 오브젝트 설명에 대해 표시되는 속성 (계속)

기본 속성	모든 속성	서비스 속성(주 참조)
	• 디지털 서명 • 디지털로 서명된 시스템 신뢰 소스 • 디지털로 서명된 복수 서명 • 오브젝트 *크기* • 오프라인 *크기* • 연관된 공간 *크기* • 최적 공간 정렬 • 해제 상태 • 압축 상태 • 오브젝트 ASP 번호 • 오브젝트 넘침 • 오브젝트 ASP 장치 • 저널링 상태 • 현재 또는 최종 저널 • 저널 이미지 • 저널 항목 생략 • 저널 시작 날짜와 시간 • 저장 작업 날짜와 시간 • 복원 작업 날짜와 시간 • 저장 명령 • 장치 유형	• APAR ID • 오브젝트의 텍스트 설명 또는 오브젝트 상태 조건

주:

1. 서비스 정보는 프로그래밍 지원 담당자가 사용하는 것이며, 오브젝트가 작성된 시스템 레벨 및 오브젝트가 제공된 이후의 오브젝트 변경 여부를 알려줍니다. 이 정보의 일부는 오브젝트를 작성하는 데 사용된 소스 멤버를 나타내고, 오브젝트가 작성된 그 소스의 최종 변경 날짜를 나타내기 때문에 사용자에게 유용합니다.
2. 라이브러리 오브젝트에는 라이브러리에 포함된 오브젝트*명*만 들어 있습니다. 오브젝트 유형 *LIB에 대해 DSPOBJD가 사용되었을 경우 오브젝트 *크기* 정보는 라이브러리 오브젝트의 *크기*만 나타내며, 라이브러리에 포함된 오브젝트의 전체 *크기*는 나타내지 않습니다.

   QLIRLIBD(라이브러리 설명 검색) API나 DSPLIB OUTPUT(*PRINT) 명령을 사용하면 라이브러리의 전체 *크기*를 찾을 수 있습니다.

DSPOBJD나 WRKOBJ 명령을 사용하여 권한 부여된 라이브러리 안의 오브젝트는 다음과 같은 기준으로 분류하여 나열할 수 있습니다.

- 이름
- 총칭명
- 유형
- 오브젝트 유형 안에서의 이름 또는 총칭명

오브젝트는 라이브러리별로 나열되고, 라이브러리에서는 유형별로 나열됩니다. 오브젝트 유형에서는 영숫자순으로 나열됩니다.

*FULL 또는 *SERVICE 옵션을 가진 여러 오브젝트를 표시할 경우 DSPOBJD 명령을 일괄처리 작업으로 사용할 수 있습니다. 출력은 표시장치에 표시되지 않고, 스풀 프린터 파일로 가서 인쇄되거나, 데이터베이스 파일로 갈 수 있습니다. 출력을 데이터베이스 파일로 보낼 경우 오브젝트의 모든 속성이 파일에 기록됩니다. 파일 QADSPOBJ에 대한 레코드 형식을 보려면 라이브러리 QSYS의 파일 QADSPOBJ에 대해 DSPFFD(파일 필드 설명 표시) 명령을 사용하십시오.

다음 명령은 이름이 ORD로 시작되는 주문 입력 파일(즉, DISTLIB 안의 파일)의 설명을 표시합니다. ORD*는 총칭명입니다.

```
DSPOBJD   OBJ(DISTLIB/ORD*)  OBJTYPE(*FILE) +
          DETAIL(*BASIC)  OUTPUT(*)
```

그 결과 기본 화면은 다음과 같습니다.

```
                    오브젝트 설명 표시 - 기본
                                                          라이브러리 1의 1
라이브러리 . . . . . :    DISTLIB        라이브러리 ASP 장치 . :    *SYSBAS

옵션을 입력한 후 Enter 키를 누르십시오.
  5=모든 속성 표시           8=서비스 속성 표시


Opt  오브젝트     유형       속성                 크기   텍스트
 _   ORDDTLP     *FILE       PF                 8192   주문 정보
 _   ORDHDRP     *FILE       PF                 8192   주문 헤더

                                                                  맨 아래
F3=나감   F12=취소     F17=맨 위  F18=맨 아래
```

*BASIC 대신 *FULL을 지정하거나 기본 화면에서 ORDDTLP 앞에 5를 입력하는 경우 전체 화면은 다음과 같습니다.

```
                    오브젝트 설명 표시 - 전체
                                                          라이브러리 1의 1
오브젝트 . . . . . :   ORDDTLP        속성 . . . . . :   PF
  라이브러리 . . . . :     DISTLIB     소유자 . . . . . :   QSECOFR
라이브러리 ASP 장치 . :   *SYSBAS       1차 그룹  . . . :   *NONE
유형 . . . . . :   *FILE

사용자 정의 정보:
   속성. . . . . :
  텍스트 . . . . . :

작성 정보:
  작성 날짜/시간 . . . . . :   06/08/89  10:17:03
  작성한 사용자  . . . . . :   QSECOFR
  작성된 시스템  . . . . . :   SYSTEM01
  오브젝트 정의역  . . . . . :   *SYSTEM

        계속...
계속하려면 Enter 키를 누르십시오.

F3=나감    F12=취소
```

```
                    오브젝트 설명 표시 - 전체
                                                          라이브러리 1의 1
오브젝트 . . . . . :   ORDDTLP        속성 . . . . . :   PF
  라이브러리 . . . . :     DISTLIB     소유자 . . . . . :   QSECOFR
라이브러리 ASP 장치. :   *SYSBAS       1차 그룹       . . . :   *NONE
유형 . . . . . :   *FILE

변경/사용 정보:
  변경 날짜/시간 . . . . . :   05/11/90  10:03:02
  수집된 자료 사용 . . . . . :   YES
  최종 사용 날짜 . . . . . :   05/11/90
  사용일 날짜 계수 . . . . . :   20
    재설정 날짜. . . . . :   03/10/90
  프로그램에 의한 변경 허용. . . . . :   YES
  감사/무결성 정보                :
  오브젝트 감사 값 . . . . . :   *NONE
  디지털 서명. . . . . :   NO

        계속...
계속하려면 Enter 키를 누르십시오.

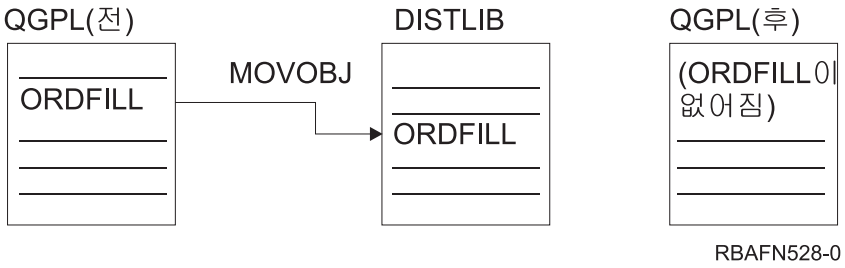
F3=나감    F12=취소
```

```
                  오브젝트 설명 표시 - 전체
                                                        라이브러리 1의 1
오브젝트 . . . . . :    ORDDTLP       속성 . . . . . :   PF
  라이브러리 . . . . :     DISTLIB     소유자 . . . . . :   QSECOFR
라이브러리 ASP 장치. :   *SYSBAS       1차 그룹      . . . :   *NONE
유형 . . . . . :    *FILE

기억장치 정보:
  크기 . . . . . :   32768
  오프라인 크기  . . . . . :   0
  연관된 공간 크기      . . . . . :   3840
  최적 공간 정렬          . . . . . :   NO
  해제 . . . . . :   NO
  압축       . . . . . :   INELIGIBLE
  오브젝트 ASP 번호  . . . . . :   1
    오브젝트 넘침. . . . . :   NO
  오브젝트 ASP 장치  . . . . . :   *SYSBAS
저널링 정보:
  현재 저널됨          . . . . . :   NO

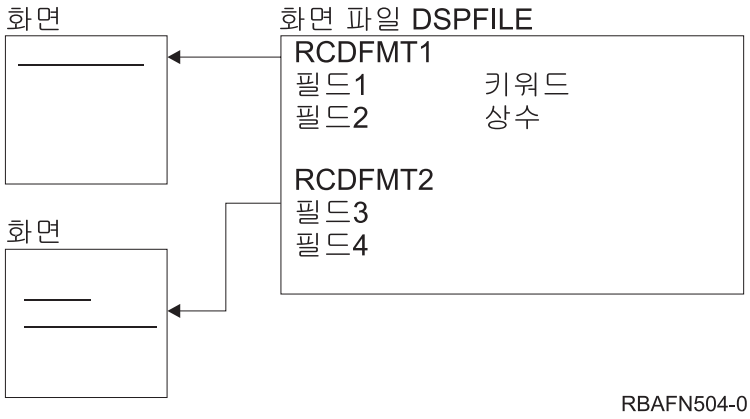
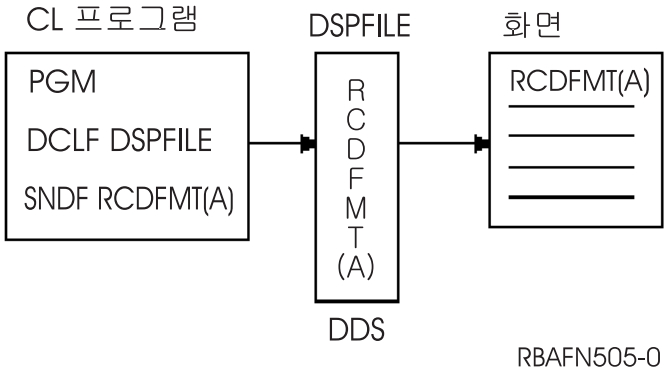
        계속...
계속하려면 Enter 키를 누르십시오.

F3=나감    F12=취소
```

```
                  오브젝트 설명 표시 - 전체
                                                        라이브러리 1의 1
오브젝트 . . . . . :    ORDDTLP       속성 . . . . . :   PF
  라이브러리 . . . . :     DISTLIB     소유자 . . . . . :   QSECOFR
라이브러리 ASP 장치 . :   *SYSBAS        1차 그룹      . . . :   *NONE
유형 . . . . . :    *FILE

저장/복원 정보:
  저장 날짜/시간 . . . . . :
  복원 날짜/시간 . . . . . :
  저장 명령. . . . . :
  장치 유형. . . . . :
                                                          맨 아래
계속하려면 Enter 키를 누르십시오.

F3=나감    F12=취소
```

오브젝트 설명 검색

RTVOBJD(오브젝트 설명 검색) 명령을 사용하면 특정 오브젝트의 설명을 CL 프로시듀어로 리턴시킬 수 있습니다. 설명을 리턴시키기 위해서는 변수가 사용됩니다. 이러한 설명을 이용하여 시스템에서 사용하지 않는 오브젝트를 검출할 수 있습니다.

이 명령은 오브젝트에 대한 변수로서 다음 설명을 리턴할 수 있습니다.

- 오브젝트가 들어 있는 라이브러리명
- 오브젝트의 확장 속성(예: 프로그램이나 파일 유형)
- 사용자 정의 속성
- 오브젝트의 텍스트 설명
- 오브젝트 소유자의 사용자 프로파일명
- 오브젝트의 1차 그룹명

- 오브젝트 ASP 번호
- 라이브러리 ASP 번호
- 오브젝트 ASP 장치
- 라이브러리 ASP 장치
- 오브젝트가 놓인 ASP에서의 오브젝트 넘침 여부 표시
- 오브젝트가 작성된 날짜와 시간
- 오브젝트가 마지막으로 변경된 날짜와 시간
- 오브젝트가 마지막으로 저장된 날짜와 시간
- SAVACT(*LIB, *SYSDFN 또는 *YES) 저장 작업시, 오브젝트가 마지막으로 저장된 날짜와 시간
- 오브젝트가 마지막으로 복원된 날짜와 시간
- 오브젝트 작성자의 사용자 프로파일명
- 오브젝트가 작성된 시스템
- 오브젝트 정의역
- 사용 자료의 수집 여부
- 오브젝트가 마지막으로 사용된 날짜
- 오브젝트가 사용된 일 수
- 사용 계수를 마지막으로 재설정한 날짜
- 오브젝트 자료의 기억장치 상태
- 오브젝트의 압축 상태
- 오브젝트 크기(바이트)
- 오브젝트의 최종 저장 시 기억장치의 크기(바이트)
- 오브젝트 저장에 사용된 명령
- 오브젝트가 테이프에 저장되었을 때 생성된 테이프 순번
- 오브젝트 저장에 사용된 테이프나 디스켓 볼륨
- 오브젝트가 마지막으로 저장된 장치 유형
- 오브젝트가 저장 파일에 저장되었을 경우 저장 파일명
- 오브젝트가 저장 파일에 저장되었을 경우 저장 파일이 들어 있는 라이브러리명
- 오브젝트가 저장될 때 사용된 파일 레이블
- 오브젝트 작성에 사용된 소스 파일명
- 오브젝트 작성에 사용된 소스 파일이 들어 있는 라이브러리명
- 소스 파일 안의 멤버명
- 소스 파일 안의 멤버가 마지막으로 갱신된 날짜와 시간
- 오브젝트가 작성된 오퍼레이팅 시스템의 레벨

- 사용권 프로그램 ID, 릴리스 레벨 및 컴파일러 수정 레벨
- 작성된 오브젝트의 오브젝트 제어 레벨
- QLICOBJDD(오브젝트 설명 변경) API에 의한 오브젝트 변경 가능성 여부에 관한 정보
- QLICOBJD(오브젝트 설명 변경) API에 의한 오브젝트 수정 가능성 여부
- 프로그램이 사용자에 의해 변경되었는지의 여부에 관한 정보
- 검색된 오브젝트가 사용권 프로그램의 일부인 경우 사용권 프로그램의 이름, 릴리스 레벨 및 수정 레벨
- 검색된 오브젝트의 작성으로 인한 프로그램 임시 수정(PTF: Program Temporary Fix) 번호
- APAR(Authorized Program Analysis Report) ID
- 오브젝트에 대한 감사 유형
- 오브젝트가 디지털로 서명되었는지 여부
- 디지털로 서명된 시스템 신뢰 소스
- 디지털로 서명된 복수 서명
- 오브젝트에 대한 현재 저널 상태
- 현재 또는 최종 저널
- 저널 이미지 정보
- 정보가 생략될 저널 항목
- 저널링이 마지막 시작된 날짜와 시간

RTVOBJD 예

다음 CL 프로시듀어에서는 RTVOBJD 명령이 특정 오브젝트의 설명을 검색합니다. 현재 라이브러리(MYLIB)에는 이름이 MOBJ인 오브젝트가 있는 것으로 가정하겠습니다.

```
DCL &LIB TYPE(*CHAR) LEN(10)
DCL  &CRTDATE  TYPE(*CHAR) LEN(13)
DCL  &USEDATE  TYPE(*CHAR) LEN(13)
DCL  &USECNT   TYPE(*DEC)  LEN(5 0)
DCL  &RESET    TYPE(*CHAR) LEN(13)
.
.
.
RTVOBJD    OBJ(MYLIB/MOBJ) OBJTYPE(*FILE) RTNLIB(&LIB)
           CRTDATE(&CRTDATE) USEDATE(&USEDATE)
           USECOUNT(&USECNT) RESETDATE(&RESET)
```

이 경우 다음의 정보가 프로그램으로 리턴됩니다.

- 현재 라이브러리명(MYLIB)이 CL 변수명 &LIB에 놓입니다.
- MOBJ의 작성 날짜가 CL 변수 &CRTDATE에 놓입니다.
- MOBJ가 마지막으로 사용된 날짜가 CL 변수 &USEDATE에 놓입니다.

- MOBJ가 사용된 일 수가 CL 변수 &USECNT에 놓입니다. 이 계수의 시작 날짜는 CL 변수 &RESET에 놓입니다.

오브젝트 정보 작성

다음과 같은 정보가 오브젝트 설명에 제공되고 오브젝트 작성 시 설정됩니다. 이것은 오브젝트 관리 및 유지보수에 유용합니다.

- 오브젝트 작성자
  - 오브젝트 작성자는 작성 조작을 수행 중인 사용자 프로파일입니다. 이것은 사용자 프로파일에 그룹 프로파일이 있으며, 그룹 프로파일에 오브젝트가 있는 경우에도 해당됩니다.
  - 소유권이 변경되어도 오브젝트 작성자는 변경되지 않습니다.
  - 작성자는 오브젝트가 복원될 때 매체상의 오브젝트 작성자입니다.
  - 오브젝트 작성자는 CRTDUPOBJ(오브젝트 사본 작성) 명령을 사용하여 오브젝트를 복제할 때 명령을 실행하는 사용자입니다.
  - IBM 제공 오브젝트의 경우 작성자는 *IBM입니다.
  - 오브젝트의 작성자는 버전 1, 릴리스 3.0 이전 시스템에 이미 있는 사용자 오브젝트의 경우 공백입니다.
- 오브젝트가 작성된 시스템
  - 오브젝트가 복원될 때 작성되는 시스템은 매체상의 오브젝트가 작성된 시스템입니다.
  - IBM 제공 오브젝트의 경우 작성되는 시스템은 00000000입니다.
  - 버전 1, 릴리스 3.0 이전 시스템상의 오브젝트에서는 작성된 시스템이 공백입니다.

시스템에서 사용하지 않는 오브젝트 검출

오브젝트 설명에서 제공되는 정보를 사용하면 시스템에서 더 이상 사용하지 않는 오브젝트를 검출하고 관리할 수 있습니다.

사용하지 않는 오브젝트를 검출하려면 최종 사용 날짜와 최종 변경 날짜를 모두 살펴보아야 합니다. 변경 명령은 명령으로 인해 오브젝트가 삭제되어 다시 작성되거나, 변경 조작으로 인해 오브젝트가 변경의 일부로서 읽히지 않는 한, 최종 사용 날짜를 갱신하지 않습니다.

- 최종 변경 날짜와 시간
  - 오브젝트가 작성되거나 변경될 때 시스템은 변경이 발생했던 날짜와 시간을 나타내는 시간소인을 오브젝트에 기록합니다.
- 최종 사용 날짜

- 최종 사용 날짜는 하루에 한 번만 변경됩니다(하루 중 오브젝트가 처음으로 사용될 때). 시스템 날짜가 사용됩니다.
- 오브젝트를 사용하기 위한 시도가 실패하면 최종 사용 날짜가 변경되지 않습니다. 예를 들어, 권한이 없는 사용자가 오브젝트 사용을 시도한 경우에는 최종 사용 날짜가 변경되지 않습니다.
- 신규 오브젝트의 경우 최종 사용 날짜란이 공백입니다.
- 이미 시스템에 있는 오브젝트가 복원될 때 최종 사용 날짜는 시스템에 있는 오브젝트로부터 옵니다. 복원 시 오브젝트가 없으면 날짜는 공백입니다.
- 복원 조작 중에 삭제되고 재작성된 오브젝트는 최종 사용 날짜를 상실합니다.
- 데이터베이스 파일에 마지막으로 사용된 날짜는 파일의 멤버 수가 0일 경우 갱신되지 **않습니다**. 예를 들어, 오브젝트를 복사할 때 CRTDUPOBJ를 사용하며 데이터베이스 파일에 멤버가 없는 경우에는 최종 사용 날짜가 갱신되지 않습니다.
- 데이터베이스 파일에 대한 최종 사용 날짜는 가장 최근에 마지막으로 사용된 파일 멤버의 최종 사용 날짜입니다.
- 논리 파일인 경우 최종 사용 날짜는 논리 멤버(또는 커서)가 마지막으로 사용된 시간입니다.
- 실제 파일인 경우 최종 사용 날짜는 실제 또는 논리 액세스에 의해 자료 공간에서 자료가 마지막으로 사용된 시간입니다.

표 4에서는 다양한 오브젝트 유형의 최종 사용 날짜가 갱신되도록 하기 위한 조작에 대해 더 자세하게 설명합니다.

표 4. 사용 정보 갱신

오브젝트 유형	**명령 및 조작**
모든 오브젝트 유형	CRTDUPOBJ(오브젝트 사본 작성) 명령과 오브젝트를 복사하기 위해 CRTDUPOBJ 명령을 사용하는 기타 명령들(예: CPYLIB(라이브러리 복사) 명령). GRTOBJAUT(오브젝트 권한 부여) 명령(참조된 오브젝트에 대해)
바인딩 디렉토리	바인드 프로그램을 작성(CRTPGM 명령)하거나 서비스 프로그램을 바운드(CRTSRVPGM 명령)하기 위해 다른 모듈이나 바인딩 디렉토리와 바인드할 때. UPDPGM(프로그램 갱신) 명령이나 UPDSRVPGM(서비스 프로그램 갱신) 명령에서 갱신할 때.
변경 요구 설명	명령 변경 요구 활동 변경(CHGCMDCRQA)
도표 형식	DSPCHT(도표 표시) 명령
C 로케일 설명	RTVCLDSRC(C 로케일 설명 소스 검색) 명령 또는 C 프로그램에서 참조될 때
클래스	작업 시작을 위해 사용될 때

표 4. 사용 정보 갱신 (계속)

오브젝트 유형	명령 및 조작
명령	수행 시 CL 프로그램을 컴파일할 때 소스 입력 유틸리티(SEU) 소스를 입력하는 동안 프롬트될 때 검사 모드로 시스템을 호출할 때 **주:** 명령 행에서 프롬트한 다음 F3 키를 누르는 것은 명령을 사용한 것으로 계산되지 않습니다.
통신측 정보(CSI)	CPI 통신 초기화 대화(CMINIT) 호출이 통신측 정보 오브젝트로부터 여러 가지 대화 특성에 대한 값을 초기화하는 데 사용될 때
연결 리스트	연결 리스트가 연결변환 지연 상태를 초과할 때
시스템 공통 프로그램(CSP: Cross System Product) 맵	CSP 어플리케이션에서 참조될 때
시스템 공통 프로그램(CSP) 표	CSP 어플리케이션에서 참조될 때
제어기 설명	제어기가 연결변환 지연 상태를 초과할 때
장치 설명	장치가 연결변환 지연 상태를 초과할 때
자료 영역	RTVDTAARA(자료 영역 검색) 명령 DSPDTAARA(자료 영역 표시) 명령
자료 대기행렬	다음의 API에 대한 사용 정보는 작업당 한 번만 갱신됨(API 중 하나가 맨 처음 시작될 때) 자료 대기행렬 송신(QSNDDTAQ) API 자료 대기행렬 수신(QRCVDTAQ) API 자료 대기행렬 검색(QMHQRDQD) API 자료 대기행렬 읽기(QMHRDQM) API
파일(그밖에 다른 것이 지정되지 않았으면 데이터베이스 파일만 가능)	닫힐 때(장치 및 저장 파일과 같은 기타 파일도 파일이 닫힐 때 함께 갱신됨) 지워질 때 초기화될 때 재구성될 때 명령: • APYJRNCHG(저널 변경 적용) 명령 • RMVJRNCHG(저널 변경 제거) 명령
폰트 자원	인쇄 조작에서 참조될 때
양식 정의	인쇄 조작에서 참조될 때
그래픽 기호 세트	GDDM* 또는 PGR 그래픽 어플리케이션 프로그램에 의해 참조될 때 내부적으로 로드되거나 GSLSS를 사용할 때

표 4. 사용 정보 갱신 (계속)

오브젝트 유형	명령 및 조작
작업 설명	작업을 설정하는 데 사용될 때
작업 스케줄	시스템이 작업 스케줄 항목에 대해 작업을 제출할 때 사용자가 WRKJOBSCDE 패널에서 옵션 10(즉시 제출)을 사용할 때
작업 대기행렬	항목이 대기행렬에 놓이거나 대기행렬에서 제거될 때
회선 설명	회선이 연결변환 지연 상태를 초과할 때
로케일	로케일 API QLGRTVLC 검색 사용자 프로파일 LOCALE 값에 유효한 *LOCALE 오브젝트에 대한 경로명이 있는 경우 작업을 시작할 때
관리 콜렉션	모든 오브젝트 유형에 영향을 미치는 명령 및 작업에 의해서만 갱신됨
매체 정의	SAVLIB, SAVOBJ, RSTLIB, RSTOBJ, SAVCHGOBJ 명령뿐만 아니라 BRMS 및 QSRSAVO API
메뉴	GO 명령을 사용하여 메뉴를 표시할 때
메세지 파일	메세지가 QCPFMSG, ##MSG1, ##MSG2 또는 QSSPMSG 이외의 메세지 파일로부터 검색되었을 때(작업 기록부 작성 시 메세지 대기행렬이 표시될 때 QHST 기록부 안의 메세지에 도움말이 요구될 때 또는 프로그램이 마크 메세지(mark message) 이외의 메세지를 수신할 때) 메세지 파일이 QCPFMSG, ##MSG1, ##MSG2 또는 QSSPMSG인 경우를 제외한 MRGMSGF(메세지 파일 병합) 명령
메세지 대기행렬	메세지가 QSYSOPR 및 QHST 이외의 메세지 대기행렬과 송수신되거나 나열될 때
모듈	바인드 프로그램을 작성(CRTPGM 명령)하거나 서비스 프로그램을 바운드(CRTSRVPGM 명령)하기 위해 다른 모듈이나 바인딩 디렉토리와 바인드할 때. UPDPGM(프로그램 갱신) 명령이나 UPDSRVPGM(서비스 프로그램 갱신) 명령에서 갱신할 때.
네트워크 인터페이스 설명	네트워크 인터페이스 설명이 연결변환 지연 상태를 초과할 때
노드 리스트	모든 오브젝트 유형에 영향을 미치는 명령 및 작업에 의해서만 갱신됨
출력 대기행렬	항목이 대기행렬에 놓이거나 대기행렬에서 제거될 때
오버레이	인쇄 조작에서 참조될 때
페이지 정의	인쇄 조작에서 참조될 때
페이지 세그먼트	인쇄 조작에서 참조될 때
패널 그룹	도움말 키가 특정 프롬트나 패널에 대한 도움말 정보를 요구하는 데 사용될 경우 사용 날짜는 갱신됨 패널이 패널 그룹으로부터 표시되거나 인쇄될 때
PDF 맵	QPQAPME(PDF 맵 항목 추가) API WRKPDFMAPE(PDF 맵 항목에 대한 작업) 명령
인쇄 설명자 그룹	인쇄 조작에서 참조될 때

표 4. 사용 정보 갱신 (계속)

오브젝트 유형	명령 및 조작
제품 가용성	모든 오브젝트 유형에 영향을 미치는 명령 및 작업에 의해서만 갱신됨
제품 로드	모든 오브젝트 유형에 영향을 미치는 명령 및 작업에 의해서만 갱신됨
프로그램	RTVCLSRC(CL 소스 검색) 명령 시스템 프로그램이 아닌 프로그램 수행 시
PSF 구성	인쇄 조작에서 참조될 때
조회 정의	보고서 생성 시 추출 또는 내보내기
조회 관리자 양식(query manager form)	보고서 생성 시 추출 또는 내보내기
조회 관리자 조회(query manager query)	보고서 생성 시 추출 또는 내보내기
색인 탐색	F11 키가 온라인 도움말 정보에 대해 사용될 때 STRSCHIDX(색인 탐색 시작) 명령이 사용될 때
서버 기억장치	VRYCFG(구성 변환)가 네트워크 서버 설명 오브젝트에 대해 실행됨
서비스 프로그램	바인드 서비스 프로그램이 활성화될 때
SQL 패키지	모든 오브젝트 유형에 영향을 미치는 명령 및 작업에 의해서만 갱신됨
서브시스템 설명	서브시스템 시작 시
철자법 지원 사전(spelling aid dictionary)	또 다른 사전의 작성을 위해 사용될 때 검색 시 철자법 검사 시, 단어가 사전에서 발견되지만, 사전이 IBM 제공 철자법 지원 사전이 아닐 때
표	변환용으로 프로그램에서 사용될 때
시간 존 설명	• 시간 존 설명과 함께 작업 시작 • 일광 절약 시간(DST) 경계를 지나갈 때 새로운 현재 오프셋을 계산하기 위해 시간 존 설명 참조
사용자 프로파일	프로파일에 대해 작업이 초기화될 때 프로파일이 그룹 프로파일이고 작업이 그룹의 멤버를 사용하여 시작될 때 GRTUSRAUT(사용자 권한 부여) 명령(참조 프로파일에 대한)
워크스테이션 사용자 정의	모든 오브젝트 유형에 영향을 미치는 명령 및 작업에 의해서만 갱신됨

다음은 오브젝트 설명에 제공된 추가 오브젝트 사용 정보입니다.

- 사용된 일 수에 대한 카운터
  - 최종 사용 날짜가 갱신되면 계수가 증가합니다.
  - 이미 시스템에 있는 오브젝트가 복원되면 사용일 수는 시스템상의 오브젝트에서 나옵니다. 복원 시 오브젝트가 없으면 계수는 0입니다.
  - 복원 조작 중에 삭제되고 재작성된 오브젝트는 사용일 계수를 상실합니다.
  - 새로운 오브젝트에 대한 사용일 계수는 0입니다.

    주: iSeries 서버는 이전 장치 파일과 새로운 장치 파일 간의 차이점을 판별할 수 *없습니다*. 시스템으로 장치 파일을 복원하는 데 있어서 같은 이름의 장치 파일이 이미 있는 경우에는 기존 파일을 삭제하여 사용일 계수를 0으로 만드십시오. 파일이 삭제되지 않으면 시스템은 이것을 기존 오브젝트의 복원 조작으로 해석하여 사용일 계수를 보유합니다.
  - 데이터베이스 파일에 대한 사용일 계수는 모든 파일 멤버에 대한 사용일 계수의 합계입니다. 합계에 넘침이 있으면, (사용일 계수 필드) 최대값이 표시됩니다.
- 사용일 계수가 재설정됨
  - 사용일 계수가 CHGOBJD(오브젝트 설명 변경) 명령 또는 QLICOBJD(오브젝트 설명 변경) API를 사용하여 재설정되면 사용일 계수가 얼마 동안 활동 상태였는지를 사용자가 알 수 있도록 날짜가 기록됩니다.
  - 파일에 대한 사용일 계수가 재설정되면 모든 멤버의 사용일 계수가 재설정됩니다.

사용일 계수와 최종 사용 날짜를 삭제할 수 있는 일반적인 상황은 다음과 같습니다.

- 시스템에 손상된 오브젝트를 복원하는 경우.
- 시스템이 제한된 상태에 있지 않을 때 프로그램을 복원하는 경우.

DSPOBJD(오브젝트 설명 표시) 명령을 사용하면 오브젝트의 전체 설명을 표시할 수 있습니다. 또한 이 명령을 사용하여 출력 파일에 설명을 기록할 수 있습니다. 설명을 검색하려는 경우 RTVOBJD(오브젝트 설명 검색) 명령을 사용할 수 있습니다.

주: 어플리케이션 프로그래밍 인터페이스(API: Application Programming Interface), QSUROBJD는 오브젝트 설명 검색 명령과 같은 정보를 제공합니다. 자세한 정보는 **iSeries Information Center**의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

RTVMBRD(멤버 설명 검색) 명령과 DSPFD(파일 설명 표시) 명령은 파일 안의 멤버에 대해 유사한 정보를 제공합니다.

다음 오브젝트 유형에 대해서는 오브젝트 사용 정보가 갱신되지 않습니다.

- 경고 표(*ALRTBL)
- 권한 부여 리스트(*AUTL)
- 구성 리스트(*CFGL)
- 서비스 클래스 설명(*COSD)

- 자료 사전(*DTADCT)
- 2바이트 문자 세트 사전(*IGCDCT)
- 2바이트 문자 세트 정렬(*IGCSRT)
- 2바이트 문자 세트 표(*IGCTBL)
- 편집 설명(*EDTD)
- 나감 등록(*EXITRG)
- 필터(*FTR)
- 양식 제어 표(*FCT)
- 폴더(*FLR)
- 인터넷 패킷 교환 설명(*IPXD)
- 저널(*JRN)
- 저널 리시버(*JRNRCV)
- 라이브러리(*LIB)
- 모드 설명(*MODD)
- 네트워크 서버 설명(*NWSD)
- NetBIOS 설명(*NTBD)
- 제품 정의(*PRDDFN)
- 참조 코드 변환 표(*RCT)
- 세션 설명(*SSND)
- S/36 기계 설명(*S36)
- 사용자 정의 SQL 유형(*SQLUDT)
- 사용자 대기행렬(*USRQ)

라이브러리 간의 오브젝트 이동

MOVOBJ(오브젝트 이동) 명령을 사용하면 라이브러리들 간에 오브젝트를 이동시킬 수 있습니다. 오브젝트를 일시적으로 사용이 불가능하게 하여 오브젝트의 구 버전을 신규 버전으로 대체시킬 수 있도록 오브젝트를 한 라이브러리에서 다른 라이브러리로 이동시킵니다. 예를 들면, 신규 1차 파일을 작성하여 이전의 1차 파일이 들어 있는 라이브러리가 아닌 다른 라이브러리에 잠시 놓아둘 수 있습니다. 이전의 1차 파일에 있는 자료가 정상적으로 신규 1차 파일에 복사되기 때문에 신규 1차 파일이 작성될 때까지는 이전의 1차 파일이 삭제되지 않습니다. 신규 1차 파일이 작성된 후에는 이전의 1차 파일이 삭제되며, 신규 1차 파일은 이전의 1차 파일이 들어 있던 라이브러리로 이동합니다.

오브젝트에 대한 오브젝트 관리 권한, 오브젝트가 이동되는 라이브러리에 대한 삭제 및 실행 권한, 그리고 오브젝트가 이동할 라이브러리에 대한 추가 권한과 읽기 권한이 사용자에게 있을 경우에만 오브젝트를 이동할 수 있습니다.

임시 라이브러리인 QTEMP로부터 오브젝트를 이동해 올 수는 있으나 오브젝트를 QTEMP로 이동할 수는 없습니다. 출력 대기행렬은 비어 있지 않는 한, 이동할 수 없습니다.

저널과 저널 리시버 이동은 이러한 유형의 오브젝트가 원래 작성되었던 라이브러리로 다시 이동되는 것 이외에는 허용되지 않습니다. 저널 오브젝트가 RCLSTG(기억장치 재생) 명령에 의해 QRCL에 놓이면, 원래의 라이브러리로 다시 이동되어 조작 가능한 상태로 되어야 합니다.

다음은 이동할 수 없는 오브젝트의 리스트입니다.

- 권한 부여 리스트(*AUTL)
- 서비스 클래스 설명(*COSD)
- 클러스터 자원 그룹(*CRG)
- 구성 리스트(*CFGL)
- 연결 리스트(*CNNL)
- 제어기 설명(*CTLD)
- 자료 사전(*DTADCT)
- 장치 설명(*DEVD)
- 표시장치 메세지 대기행렬(*MSGQ)
- 문서(*DOC)
- 편집 설명(*EDTD)
- 나감 등록(*EXITRG)
- 폴더(*FLR)
- 2바이트 문자 세트(DBCS) 폰트 표(*IGCTBL)
- 이미지 카탈로그(*IMGCLG)
- 인터넷 패킷 교환 설명(*IPXD)
- 작업 스케줄(*JOBSCD)
- 라이브러리(*LIB)
- 회선 설명(*LIND)
- 모드 설명(*MODD)
- NetBIOS 설명(*NTBD)
- 네트워크 인터페이스 설명(*NWID)
- SQL(구조화 조회 언어) 패키지(*SQLPKG)
- System/36™ 기계 설명(*S36)
- 시스템 이력 기록부(QHST)
- 시스템 오퍼레이터 메세지 대기행렬(QSYSOPR)

- 시간 존 설명(*TIMZON)
- 사용자 정의 SQL 유형(*SQLUDT)
- 사용자 프로파일(*USRPRF)

다음 예에서는 파일이 QGPL(작성되었을 때 위치하는)로부터 주문 입력 라이브러리 DISTLIB로 이동하여 다른 주문 입력 파일과 그룹을 이룹니다.



오브젝트를 이동하려면 오브젝트 유형(OBJTYPE)뿐만 아니라 To-라이브러리(TOLIB)도 지정해야 합니다.

```
MOVOBJ  OBJ(QGPL/ORDFILL)  OBJTYPE(*FILE)  TOLIB(DISTLIB)
```

오브젝트 이동 시, 다른 오브젝트들이 종속되어 있는 오브젝트를 이동하지 않도록 주의해야 합니다. 예를 들어, CL 프로시듀어는 수행 시 모듈 작성 당시 존재하던 라이브러리와 동일한 라이브러리에 프로시듀어가 들어 있도록 하기 위해 프로시듀어에서 사용되는 명령의 명령 정의에 따라 달라질 수 있습니다. 컴파일 시와 수행 시에 *LIBL이 지정된 경우 지정 라이브러리나 라이브러리 리스트의 라이브러리에서 명령 정의를 찾을 수 있습니다. 라이브러리명이 지정되면 명령 정의는 수행 시 및 컴파일 시의 라이브러리와 동일한 라이브러리에 들어 있어야 합니다. *LIBL이 지정된 경우 명령 정의가 라이브러리 리스트 안의 라이브러리로 이동되는 한, 이 명령 정의를 컴파일 시와 프로그램 수행 시에서 이동시킬 수 있습니다. 마찬가지로, 사용자가 작성한 어플리케이션 프로그램을 특정 라이브러리에 들어 있는 특정 오브젝트에 종속시킬 수 있습니다.

다른 오브젝트를 참조하는 오브젝트는 해당 오브젝트의 위치에 종속됩니다(오브젝트 위치에 대해 *LIBL이 지정되어 있더라도). 그러므로 오브젝트를 이동할 경우 다른 오브젝트 안에서의 해당 오브젝트(이동되는 오브젝트)에 대한 참조사항도 변경해야 합니다. 다음은 다른 오브젝트를 참조하는 오브젝트의 예입니다.

- 서브시스템 설명은 작업 대기행렬, 클래스, 메세지 대기행렬 및 프로그램을 참조합니다.
- 명령 정의는 REXX 프로시듀어를 포함하고 있는 프로그램, 메세지 파일, 도움말 패널 그룹 및 소스 파일을 참조합니다.
- 장치 파일은 출력 대기행렬을 참조합니다.
- 장치 설명은 변환 표를 참조합니다.
- 작업 설명은 작업 대기행렬과 출력 대기행렬을 참조합니다.

- 데이터베이스 파일은 다른 데이터베이스 파일을 참조합니다.
- 논리 파일은 실제 파일 또는 형식 선택을 참조합니다.
- 사용자 프로파일은 프로그램, 메뉴, 작업 설명, 메세지 대기행렬 및 출력 대기행렬을 참조합니다.
- CL 프로그램은 화면 파일, 자료 영역 및 다른 프로그램을 참조합니다.
- 화면 파일은 데이터베이스 파일을 참조합니다.
- 프린터 파일은 출력 대기행렬을 참조합니다.

주: 시스템 라이브러리 QSYS로부터 오브젝트를 이동할 때는 주의해야 합니다. 이 오브젝트는 시스템이 효율적으로 수행되는 데 반드시 필요한 오브젝트이므로 시스템이 이 오브젝트를 찾을 수 있도록 해야 합니다. 또한 범용 라이브러리 QGPL의 일부 오브젝트, 특히 작업과 출력 대기행렬에 대한 오브젝트일 때도 주의해야 합니다.

MOVOBJ 명령은 한번에 한 오브젝트만 이동합니다.

오브젝트 사본 작성

CRTDUPOBJ(오브젝트 사본 작성) 명령을 사용하면 기존 오브젝트의 사본을 작성할 수 있습니다. 오브젝트 사본은 원래 오브젝트와 오브젝트 유형 및 권한이 동일하며, 원래 오브젝트와 동일한 보조 기억장치 풀(ASP: Auxiliary Storage Pool)에 작성됩니다. 명령을 발행한 사용자가 오브젝트 사본을 소유합니다.

주:

1. 저널된 파일의 오브젝트 사본을 작성할 경우 오브젝트 사본(파일)에 저널링을 수행할 수 없습니다. 그러나 나중에 저널링하기 위해 이 오브젝트를 선택할 수 있습니다. 오브젝트 사본을 작성했으며 오브젝트(파일)에 멤버가 없는 경우 최종 사용 날짜 필드는 공백이며 날짜를 계산할 때 0이 사용됩니다.

오브젝트에 대한 오브젝트 관리 및 사용 권한, 오브젝트 사본이 위치할 라이브러리에 대한 사용 및 추가 권한, 원래 오브젝트가 상주하는 라이브러리에 대한 사용 권한, 그리고 사용자 프로파일 프로세스에 대한 추가 권한이 사용자에게 있으면 오브젝트를 복제할 수 있습니다.

권한 부여 리스트를 복제하려면 오브젝트에 대한 권한 부여 리스트 관리 권한과 라이브러리 QSYS에 대한 추가 및 오브젝트 조작 권한이 있어야 합니다.

작업 대기행렬, 메세지 대기행렬, 출력 대기행렬, 자료 대기행렬의 정의만 복제할 수 있습니다. 작업 대기행렬과 출력 대기행렬은 임시 라이브러리(QTEMP)에는 복제할 수 없습니다. 실제 파일이거나 저장 파일인 경우 파일 안의 자료도 복제할 것인지의 여부를 지정할 수 있습니다.

다음 오브젝트들은 복제할 수 *없습니다*.

- 서비스 클래스 설명(*COSD)
- 클러스터 자원 그룹(*CRG)
- 구성 리스트(*CFGL)
- 연결 리스트(*CNNL)
- 제어기 설명(*CTLD)
- 자료 사전(*DTADCT)
- 장치 설명(*DEVD)
- 자료 대기행렬(*DTAQ)
- 문서(*DOC)
- 편집 설명(*EDTD)
- 나감 등록(*EXITRG)
- 폴더(*FLR)
- DBCS 폰트 표(*IGCTBL)
- 이미지 카탈로그(*IMGCLG)
- 인터넷 패킷 교환 설명(*IPXD)
- 작업 스케줄(*JOBSCD)
- 저널(*JRN)
- 저널 리시버(*JRNRCV)
- 라이브러리(*LIB)
- 회선 설명(*LIND)
- 모드 설명(*MODD)
- 네트워크 인터페이스 설명(*NWID)
- 네트워크 서버 설명(*NWSD)
- 참조 코드 변환 표(*RCT)
- 서버 기억장치(*SVTSTG)
- 철자법 지원 사전(*SPADCT)
- SQL 패키지(*SQLPKG)
- System/36 기계 설명(*S36)
- 시스템 오퍼레이터 메세지 대기행렬(QSYSOPR)
- 시스템 이력 기록부(QHST)
- 시간 존 설명(*TIMZON)
- 사용자 정의 SQL 유형(*SQLUDT)
- 사용자 프로파일(*USRPRF)
- 사용자 대기행렬 (*USRQ)

파일 안의 자료 중 일부는 중복하여 복사할 수 있는데, 이때 선택값을 지정하는 CPYF 명령과 CRTDUPOBJ 명령을 사용합니다.

다음 명령은 주문 헤더 실제 파일의 사본을 작성하여 파일 안의 자료를 복사합니다.

```
CRTDUPOBJ OBJ(ORDHDRP) FROMLIB(DSTPRODLIB) OBJTYPE(*FILE) +
          TOLIB(DISTLIB2) NEWOBJ(*SAME) DATA(*YES)
```

오브젝트 사본을 작성할 때는 다른 오브젝트를 참조하는 오브젝트 사본의 작성 결과를 고려해야 합니다. 대부분의 오브젝트는 다른 오브젝트를 이름으로 참조하고, 대부분의 참조는 특정 라이브러리명으로 규정됩니다. 그러므로 오브젝트 사본이 들어 있는 라이브러리가 아닌 다른 라이브러리에 존재하는 오브젝트를 오브젝트 사본이 참조하는 경우가 있습니다. 파일 이외의 다른 오브젝트 유형인 경우 다른 오브젝트에 대한 참조는 오브젝트 사본에 복제됩니다. 파일인 경우 오브젝트 사본은 원래 파일의 형식을 공유합니다.

From 라이브러리에 존재하면서 논리 파일의 기초가 되는 실제 파일은 To 라이브러리에도 존재해야 합니다. To 라이브러리와 From 라이브러리에 있는 실제 파일의 레코드 형식명과 레코드 레벨 ID가 비교된 후 실제 파일이 일치하지 않으면 논리 파일이 복제되지 않습니다.

논리 파일이 From 라이브러리에 존재하는 형식 선택을 사용할 경우에는 형식 선택이 To 라이브러리에도 존재한다는 것을 전제로 합니다.

오브젝트의 이름 변경

RNMOBJ(오브젝트의 이름 변경) 명령을 사용하면 오브젝트의 이름을 변경할 수 있습니다. 그러나 오브젝트에 대한 오브젝트 관리 권한과 오브젝트가 들어 있는 라이브러리에 대한 갱신 권한 및 실행 권한이 있을 경우에만 오브젝트의 이름을 변경할 수 있습니다.

권한 부여 리스트의 이름을 변경하려면 권한 부여 리스트 관리 권한과 라이브러리 QSYS에 대한 갱신 및 읽기 권한이 있어야 합니다.

다음 오브젝트 이름은 변경시킬 수 *없습니다*.

- 서비스 클래스 설명(*COSD)
- 클러스터 자원 그룹(*CRG)
- 자료 사전(*DTADCT)
- DBCS 폰트 표(*IGCTBL)
- 표시장치 메세지 대기행렬(*MSGQ)
- 문서(*DOC)
- 나감 등록(*EXITRG)

- 폴더(*FLR)
- 작업 스케줄(*JOBSCD)
- 저널(*JRN)
- 저널 리시버(*JRNRCV)
- 모드 설명(*MODD)
- 네트워크 서버 설명(*NWSD)
- SQL 패키지(*SQLPKG)
- System/36 기계 설명(*S36)
- 시스템 이력 기록부(QHST)
- 시스템 라이브러리 QSYS 및 임시 라이브러리 QTEMP
- 시스템 오퍼레이터 메세지 대기행렬(QSYSOPR)
- 시간 존 설명(*TIMZON)
- 사용자 정의 SQL 유형(*SQLUDT)
- 사용자 프로파일(*USRPRF)

출력 대기행렬이 비어 있지 않은 한, 그 출력 대기행렬의 이름도 변경할 수 없습니다. 사용권 프로그램도 IBM 제공 명령을 사용하기 때문에 IBM 제공 명령의 이름을 변경해서는 안됩니다.

오브젝트의 이름을 변경하려면 오브젝트의 현재 이름, 변경될 이름 및 오브젝트 유형을 지정해야 합니다.

다음의 RNMOBJ 명령은 오브젝트 ORDERL을 ORDFILL로 이름을 변경합니다.

```
RNMOBJ   OBJ(QGPL/ORDERL)  OBJTYPE(*FILE)   NEWOBJ(ORDFILL)
```

새로운 오브젝트명에 규정된 이름을 지정할 수 없는데, 그 이유는 오브젝트가 동일한 라이브러리에 남기 때문입니다. RNMOBJ 명령을 발행할 때 이름을 변경할 오브젝트가 사용 중이면 이 명령이 실행되기는 하지만 오브젝트의 이름을 변경하지는 않습니다. 그 결과 시스템이 메세지를 송신합니다.

오브젝트의 이름을 변경할 때는 다른 오브젝트가 종속되어 있는 오브젝트의 이름을 변경하지 않도록 주의해야 합니다. 예를 들어, CL 프로그램은 프로그램 수행 시 또는 프로그램 컴파일 시에 동일한 이름의 프로그램에 사용된 명령의 명령 정의에 종속됩니다. 따라서 명령 정의가 컴파일 시기와 수행 시기에 이름 변경이 발생하면 명령을 찾을 수 없으므로 프로그램이 수행되지 않습니다. 마찬가지로, 사용자가 기록한 어플리케이션 프로그램은 컴파일 또는 수행 시 모두 동일한 이름의 오브젝트에 종속됩니다.

저널, 저널 리시버, 자료 사전, 클러스터 자원 그룹, SQL 패키지가 들어 있는 라이브러리의 이름을 변경할 수 없습니다.

다른 오브젝트를 참조하는 오브젝트는 오브젝트 및 라이브러리명에 종속될 수 있습니다(*LIBL이 라이브러리명으로 지정될 수 있더라도). 따라서 오브젝트의 이름을 변경할 경우 다른 오브젝트에서 이 오브젝트에 대한 참조를 변경해야 합니다. 다른 오브젝트를 참조하는 오브젝트의 리스트를 보려면 153 페이지의 『라이브러리 간의 오브젝트 이동』 부분을 참조하십시오.

실제 파일이나 논리 파일의 이름을 변경할 때 파일 안의 멤버 이름은 변경되지 않습니다. 그러나 RNMM(멤버의 이름 변경) 명령을 사용하면 실제 또는 논리 파일 멤버의 이름을 변경할 수 있습니다.

주: 시스템 라이브러리 QSYS 안의 오브젝트 이름을 변경할 때는 주의해야 합니다. 이 오브젝트는 시스템이 효율적으로 수행되는 데 반드시 필요한 오브젝트이므로 시스템이 이 오브젝트를 찾을 수 있도록 해야 합니다. 범용 라이브러리 QGPL 안의 일부 오브젝트에 대해서도 마찬가지입니다.

오브젝트 압축 또는 압축 해제

CPROBJ(오브젝트 압축) 명령을 사용하면 선택된 오브젝트를 압축시켜 시스템상의 디스크 공간을 절약할 수 있으며, DCPOBJ(오브젝트 압축 해제) 명령을 사용하면 압축된 오브젝트를 압축 해제시킬 수 있습니다. 압축과 압축 해제에 지원되는 오브젝트 유형에는 *PGM, *SRVPGM, *MODULE, *PNLGRP, *MENU(UIM 메뉴만) 및 *FILE(화면 파일 또는 인쇄 파일만)이 있으며, 데이터베이스 파일은 압축할 수 없습니다. OS/400 제공 오브젝트 뿐만 아니라 고객 오브젝트도 압축 및 압축 해제가 가능합니다. DSPOBJD(오브젝트 설명 표시) 명령(*FULL 화면) 또는 RTVOBJD(오브젝트 설명 검색) 명령을 사용하면 오브젝트의 압축 상태를 검색하거나 볼 수 있습니다.

오브젝트 압축

CPROBJ나 DCPOBJ 명령을 사용하면 오브젝트 유형 *PGM, *SRVPGM, *MODULE, *PNLGRP, *MENU, *FILE(화면 파일과 인쇄 파일만)을 압축 또는 압축 해제할 수 있습니다. 다음 두 가지를 모두 만족하는 경우에만 오브젝트를 압축할 수 있습니다.

- 시스템이 오브젝트에 대해 배타적 잠금을 확보할 수 있을 때
- 압축된 크기만큼 디스크 공간이 절약될 때

오브젝트를 압축할 때는 다음 제한사항이 적용됩니다.

- 오퍼레이팅 시스템 버전 1 릴리스 3 이전에 작성된 프로그램은 압축시킬 수 없습니다.
- 버전 3 릴리스 6 이전에 작성되었고 다시 변환시키지 않은 프로그램, 서비스 프로그램, 모듈들은 압축시킬 수 없습니다.
- 프로그램의 페이징 풀(paging pool) 값이 *BASE가 아니면, IBM 제공 라이브러리 QSYS와 QSSP 안의 프로그램은 압축시킬 수 없습니다. DSPPGM(프로그램 표시)

명령을 사용하여 프로그램의 페이징 풀값을 확인하십시오. QSYS와 QSSP 이외의 라이브러리에 있는 프로그램은 페이징 풀값에 관계 없이 압축시킬 수 있습니다.

- 속성이 UIM인 메뉴만 압축시킬 수 있습니다.
- 속성이 DSPF와 PRTF인 파일만 압축시킬 수 있습니다.
- 시스템 라이브러리 안의 프로그램 오브젝트를 압축하기 위해서는 시스템이 제한된 상태(모든 서브시스템이 종료됨)에 있어야 합니다.
- 프로그램이 압축되었을 때 또는 이상 종료될 때에는 시스템에서 프로그램을 수행시키지 않아야 합니다.

다음 표와 같이, 제한되지 않은 상태에서 여러 작업을 사용할 경우에는 압축이 더 빠르게 이루어집니다.

표 5. 여러 작업을 사용한 오브젝트 압축

오브젝트 유형	IBM 제공	사용자 제공
*FILE	작업 3: QSYS	작업 7: USRLIB1
*MENU	작업 2: QSYS	작업 8: USRLIB1
*MODULE	적용 불가	작업 10: USRLIB1
*PGM	제한 상태 전용	작업 5: USRLIB1
*PNLGRP	작업 1: QSYS 작업 4: QHLPSYS	작업 6: USRLIB1
*SRVPGM	작업 11: QSYS	작업 9: USRLIB1

일시적으로 압축 해제된 오브젝트

압축된 오브젝트를 사용할 때는 시스템에 의해 일시적으로 압축이 자동 해제됩니다. 일시적으로 압축 해제된 오브젝트는 다음의 경우가 될 때까지 일시적인 압축 해제 상태에 놓입니다.

- 시스템 IPL. 이로 인해 일시적으로 압축이 해제된 오브젝트가 삭제됩니다(압축된 오브젝트는 남아 있음).
- RCLTMPSTG(임시 기억장치 재생) 명령은 일시적으로 압축이 해제된 오브젝트를 재생하는 데 사용됩니다. 이로 인해 일시적으로 압축이 해제된 오브젝트는 지정된 일수만큼 사용되지 않는 경우 삭제되며, 압축된 오브젝트만 남습니다.
- 일시적으로 압축이 해제된 오브젝트는 3일 이상 또는 같은 IPL에서 6번 이상 사용되면 오브젝트가 영구적으로 압축 해제됩니다.
- DCPOBJ 명령은 오브젝트의 압축 해제에 사용되며, 이 경우 오브젝트가 영구적으로 압축 해제됩니다.
- 시스템이 오브젝트에 대해 배타적 잠금 상태를 가지고 있습니다.

주:

1. 유형이 *PGM, *SRVPGM 또는 *MODULE인 오브젝트는 일시적인 압축 해제가 불가능합니다. 압축된 프로그램을 호출하거나 프로그램을 디버그하면 프로그램은 영구적으로 압축을 자동 해제합니다.

2. 압축된 파일 오브젝트는 열릴 때 자동으로 압축이 해제됩니다.
3. 압축 파일에 대한 설명이 검색될 경우 파일은 일시적으로 압축이 해제됩니다. 파일 검색에 대한 두 가지 예는 다음과 같습니다.
   - 파일의 필드 레벨 정보를 표시하는 데 DSPFFD(파일 필드 설명 표시) 명령을 사용하는 경우
   - 파일을 선언하는 데 DCLF(파일 선언) 명령을 사용하는 경우

오브젝트의 자동 압축 해제

OS/400에서 제공하는 압축 오브젝트나 기타 IBM 사용권 프로그램들은 *사용권 프로그램이 설치된 후* 시스템에 의해 압축이 해제됩니다. 압축 해제는 시스템상에 사용이 가능한 기억장치가 충분할 때만 발생합니다.

QDCPOBJx라는 시스템 작업이 시스템에 의해 자동으로 시작되어 오브젝트의 압축을 해제합니다.

QDCPOBJ 작업의 수는 프로세서 + 1의 수에 기초합니다. 이 작업들은 사용자가 변경, 종료 또는 보류할 수 없는 우선순위 60에서 수행되는 시스템 작업입니다. QDCPOBJx 작업은 WRKACTJOB(활동 작업에 대한 작업) 명령에 의해 다음 상태 중 하나가 될 수 있습니다.

- RUN(수행): 작업이 오브젝트를 압축 해제합니다.
- EVTW(이벤트 대기): 작업이 오브젝트를 압축 해제하지 않습니다. 더 많은 오브젝트가 압축 해제를 필요로 하는 경우에 작업이 활성화됩니다(즉, 사용권 프로그램이 추가로 설치되는 경우).
- DLYW(지연 대기): 작업이 일시적으로 중단됩니다. 다음과 같은 상황은 QDCPOBJx 작업 중단을 일으킵니다.
  - 시스템이 제한 상태에서 실행됩니다. (즉, ENDSYS 또는 ENDSBS *ALL이 실행됩니다.)
  - 사용권 프로그램이 "사용권 프로그램에 대한 작업" 화면으로부터 방금 설치되었습니다. 작업은 압축 해제의 시작에 앞서, 최대 15분 동안 지연 대기 상태에 놓입니다.
- LCKW(잠금 대기): 작업이 내부 잠금 대기 중입니다. 일반적으로, 이와 같은 상황은 QDCPOBJ 작업이 DLYW 상태일 때 발생합니다.

오퍼레이팅 시스템이 기존 오퍼레이팅 시스템상에 설치된 경우에는 다음과 같은 기억장치 요구사항이 적용됩니다.

- QDCPOBJx 작업을 시작하려는 경우 사용되지 않은 250메가바이트보다 큰 시스템이 필요합니다.
- 사용 가능한 기억장치가 750MB보다 큰 시스템에서는 방금 설치된 모든 시스템 오브젝트의 압축을 해제하기 위해 작업이 제출됩니다.

- 사용 가능한 기억장치가 250MB보다 작은 시스템에서는 작업이 제출되지 않고, 오브젝트가 사용될 때 압축이 해제됩니다.
- 사용 가능한 기억장치가 100MB - 750MB 사이인 시스템에서는 자주 사용되는 오브젝트만 자동으로 압축이 해제됩니다.

*자주 사용되는 오브젝트*들은 최소한 5번 이상 사용되었으며, 최근 14일 안에 마지막으로 사용된 오브젝트들입니다. 자주 사용하지 않은 나머지 오브젝트들은 압축 상태로 남습니다.

오퍼레이팅 시스템이 ″사용권 내부 코드 설치(LIC)″ 화면으로부터 옵션 2의 사용권 내부 코드 설치 및 시스템 초기화를 사용하여 초기화된 시스템상에 설치된 경우에는 시스템에 1000MB보다 큰, 사용되지 않은 기억장치가 있어야 합니다.

마지막 시스템 종료 시 QDCPOBJx 작업이 활동 중이던 경우에는 그 다음 IPL 시 그 작업이 다시 시작됩니다.

오브젝트 삭제

해당 오브젝트 유형의 삭제(DLTxxx) 명령을 사용하거나 오브젝트에 대한 작업 화면(라이브러리에 대한 작업(WRKLIB) 화면에서 나옴)에 있는 삭제 옵션을 사용하면 오브젝트를 삭제할 수 있습니다. 오브젝트를 삭제하려면 오브젝트에 대한 오브젝트 존재 권한 및 라이브러리에 대한 실행 권한이 있어야 합니다. 권한 부여 리스트는 권한 부여 리스트의 소유자 또는 *ALLOBJ 특수 권한을 가진 사용자만이 삭제할 수 있습니다.

오브젝트를 삭제할 때는 해당 오브젝트를 사용 중이거나 필요로 하는 다른 사용자가 없는지 확인해야 합니다. 일반적으로 오브젝트가 사용 중인 경우에는 삭제할 수 없습니다. 그러나 호출에 앞서 ALCOBJ(오브젝트 할당) 명령을 사용하여 프로그램을 할당하지 않았으면 프로그램을 삭제할 수 있습니다.

프로그램, 명령 및 장치 파일을 작성하는 데 사용되는 명령과 같은 일부 작성 명령은 REPLACE 옵션을 사용합니다. 이 옵션은 사용자들이 이전에 오브젝트를 대체시킨 구 버전을 계속해서 사용할 수 있도록 허용합니다. 시스템은 이렇게 재작성된 오브젝트의 구 버전을 라이브러리 QRPLOBJ에 저장합니다.

시스템 라이브러리에 존재하는 오브젝트는 주의해서 삭제해야 합니다. 이 오브젝트는 시스템이 올바르게 수행되는 데 있어서 반드시 필요한 것입니다.

대부분의 삭제 명령에는 오브젝트명 대신에 총칭명을 지정할 수 있습니다. 총칭적인 삭제를 사용하기 전에 DSPOBJD 명령을 사용하여 총칭명을 지정한 후 총칭적인 삭제를 통해 삭제하려는 오브젝트만이 삭제되는지를 확인할 수 있습니다. 오브젝트의 총칭적인 지정 방법에 대해 자세히 알려면 127 페이지의 『총칭 오브젝트명 사용』 부분을 참조하십시오.

라이브러리 삭제 방법에 대해 알려면 135 페이지의『라이브러리 삭제 및 지우기』부분을 참조하십시오.

자원 할당

오브젝트는 무결성을 보장하고 가능한 최고 수준의 동시성을 지원하도록 시스템에 할당됩니다. 몇 가지 조작이 시스템상에서 동시에 수행된다고 하더라도 오브젝트는 보호됩니다. 예를 들면, 두 명의 사용자가 오브젝트를 동시에 읽거나 한 사용자가 오브젝트를 읽기만 하는 동안 다른 사용자가 오브젝트를 읽고 갱신할 수 있도록 오브젝트가 할당됩니다.

OS/400은 오브젝트에서 수행되는 기능에 따라 오브젝트를 할당합니다. 예를 들어, 다음과 같은 경우가 있습니다.

- 오브젝트를 화면에 표시하거나 덤프 중인 경우에는 다른 사용자가 그 오브젝트를 읽을 수 있습니다.
- 오브젝트를 변경, 삭제, 이름 변경 또는 이동 중인 경우에는 다른 사용자는 그 오브젝트를 사용할 수 없습니다.
- 사용자가 오브젝트를 저장중인 경우에는 다른 사용자가 그 오브젝트를 읽을 수는 있지만 오브젝트를 갱신하거나 삭제할 수는 없으며, 사용자가 오브젝트를 복원 중인 경우에는 다른 사용자가 오브젝트를 읽거나 갱신할 수 없습니다.
- 한 사용자가 입력용으로 데이터베이스 파일을 여는 경우에는 다른 사용자가 이 파일을 읽을 수 있습니다. 한 사용자가 출력용으로 데이터베이스 파일을 여는 경우에도 다른 사용자가 이 파일을 갱신할 수 있습니다.
- 한 사용자가 장치 파일을 여는 경우에는 다른 사용자가 그 파일을 읽을 수만 있습니다.

일반적으로, 오브젝트는 수요에 따라 할당되는 것으로서 작업 단계가 오브젝트를 필요로 하면 그 오브젝트를 할당하여 사용한 후 다른 작업이 그 오브젝트를 사용할 수 있도록 오브젝트의 할당을 해제합니다. 오브젝트는 오브젝트를 요구하는 첫 번째 작업에 할당됩니다. 프로그램에서 사용자 요청에 의해서 오브젝트가 할당될 수 없는 경우 사용자가 발생한 예외를 처리할 수 있습니다. (메세지 모니터에 대해 자세히 알려면 제 7 장과 제 8 장을 참조하고, 예외 처리에 대해서는 해당 고급 언어 안내서를 참조하십시오.)

때로는 부분적으로만 완료된 기능이 오브젝트를 대기할 필요없이 가용성을 확보하기 위해 작업에서 오브젝트를 필요로 하기 전에 작업에 오브젝트를 할당할 수 있습니다. 이것을 오브젝트의 사전 할당(preallocating)이라고 합니다. ALCOBJ(오브젝트 할당) 명령을 사용하면 오브젝트를 사전에 할당할 수 있습니다.

오브젝트는 사용 목적(읽기 또는 갱신)과 공유 가능 여부(둘 이상의 작업에서 사용됨)를 기준으로 할당됩니다. 파일과 멤버는 항상 *SHRRD로 할당되고, 파일 자료는 잠금

상태와 함께 지정된 잠금 레벨로 할당됩니다. 잠금 상태는 오브젝트의 사용 및 공유 여부를 식별합니다. 5개의 잠금 상태는 다음과 같습니다(괄호 안은 매개변수 값임).

- 배타적 잠금 상태(*EXCL). 오브젝트가 요구한 작업에서 독점 사용할 수 있도록 예약되고, 다른 작업들은 그 오브젝트를 사용할 수 없습니다. 그러나 오브젝트가 이미 다른 작업에 할당되어 있는 경우 작업은 이 오브젝트의 배타적 사용을 예약할 수 없습니다. 이러한 잠금 상태는 수행 중인 기능이 완료될 때까지 다른 사용자가 오브젝트에 액세스하지 못하도록 할 때 적합합니다.
- 배타적 읽기 허용 잠금 상태(*EXCLRD). 오브젝트가 오브젝트를 요구한 작업에 할당되지만 다른 작업들이 그 오브젝트를 읽을 수 있습니다. 이러한 잠금 상태는 다른 사용자가 오브젝트에 대해 읽기 이외의 작업을 하지 못하도록 만들 때 적합합니다.
- 공유된 갱신 잠금 상태(*SHRUPD). 오브젝트가 갱신이나 읽기에 다른 작업과 공유될 수 있습니다. 즉, 다른 사용자가 같은 오브젝트에 대해 공유된 읽기 잠금 상태나 공유된 갱신 잠금 상태를 요구할 수 있습니다. 이러한 잠금 상태는 한 명의 사용자 이외에 다른 사용자가 같은 오브젝트를 읽거나 변경하도록 허용할 때 적합합니다.
- 공유된 갱신 불가 잠금 상태(*SHRNUP). 오브젝트가 작업에서 공유된 갱신 불가 잠금 상태나 공유된 읽기 잠금 상태를 요구할 경우 다른 작업과 공유될 수 있습니다. 이러한 잠금 상태는 오브젝트의 변경을 원하지 않으면서 다른 사용자도 오브젝트를 변경할 수 없도록 할 때 적합합니다.
- 공유된 읽기 잠금 상태(*SHRRD). 사용자가 오브젝트에 대해 배타적 사용을 요구하지 않는 경우 오브젝트를 다른 작업과 공유시킬 수 있습니다. 즉, 다른 사용자가 배타적 읽기 허용 잠금 상태, 공유된 갱신 잠금 상태, 공유된 갱신 불가 잠금 상태 또는 공유된 읽기 잠금 상태를 요구할 수 있습니다.

주: 라이브러리 할당이 라이브러리 안의 오브젝트에 수행될 수 있는 조작을 제한하지는 않습니다. 즉, 한 작업이 라이브러리에 배타적 읽기 허용 잠금 상태나 공유된 갱신 잠금 상태로 놓일 경우 다른 작업들이 더 이상 라이브러리 안에 오브젝트를 넣거나 라이브러리로부터 오브젝트를 제거할 수 없습니다. 그러나 다른 작업은 라이브러리 안에 오브젝트를 계속 갱신할 수 있습니다.

다음 표에서는 오브젝트에 대한 유효 잠금 상태의 조합을 보여줍니다.

표 6. 유효 잠금 상태 조합

한 작업 아래의 잠금 상태를 사용할 경우	다른 작업에서 사용할 수 있는 잠금 상태
*EXCL	없음
*EXCLRD	*SHRRD
*SHRUPD	*SHRUPD 또는 *SHRRD
*SHRNUP	*SHRNUP 또는 *SHRRD
*SHRRD	*EXCLRD, *SHRUPD, *SHRNUP 또는 *SHRRD

대부분의 오브젝트 유형에는 모두 5가지의 잠금 상태(*EXCL, *EXCLRD, SHRUPD, SHRNUP, SHRRD)를 지정할 수 있습니다. 이것은 모든 오브젝트 유형에 적용되는 것은 아닙니다. 다음은 지정된 5개의 잠금 상태를 모두 가질 수 **없는** 오브젝트 유형과 유효 잠금 상태에 대한 표입니다.

표 7. 특정 오브젝트 유형에 대한 유효 잠금 상태

오브젝트 유형	*EXCL	*EXCLRD	*SHRUPD	*SHRNUP	*SHRRD
장치 설명		x			
라이브러리		x	x	x	x
메세지 대기행렬	x				x
패널 그룹	x	x			
프로그램	x	x			x
서브시스템 설명	x				

오브젝트를 할당하려면 오브젝트에 대한 오브젝트 존재 권한, 오브젝트 관리 권한 또는 조작 권한이 있어야 합니다. 할당된 오브젝트는 라우팅 단계가 종료할 때 자동으로 할당 해제됩니다. DLCOBJ(오브젝트 할당 해제) 명령을 사용하면 다른 시점에서 오브젝트의 할당을 해제할 수 있습니다.

호출에 앞서 프로그램을 할당하면 프로그램이 삭제되는 것을 막을 수 있습니다. 프로그램이 동시에 서로 다른 작업에서 수행되지 못하도록 하려면 프로그램이 작업에서 호출되기 전에 배타적 잠금을 각 작업의 프로그램에 설정해야 합니다.

APPC 장치 설명 할당에는 ALCOBJ나 DLCOBJ 명령을 사용할 수 없습니다.

다음 예는 갱신용으로 두 개의 파일 멤버를 필요로 하는 일괄처리 작업입니다. 갱신 중에는 다른 프로그램이 각 파일의 멤버를 읽을 수 있으나 작업이 수행되는 동안 다른 프로그램이 이 멤버를 갱신할 수 없습니다. 각 파일의 첫 번째 멤버는 배타적 읽기 허용 잠금 상태로 사전 할당됩니다.

```
//JOB  JOBD(ORDER)
  ALCOBJ  OBJ((FILEA *FILE *EXCLRD) (FILEB *FILE *EXCLRD))
  CALL  PROGX
//ENDJOB
```

사용자에게 할당된 오브젝트는 다른 사용자가 이 오브젝트를 필요로 할 수 있기 때문에 사용이 끝나는 대로 할당 해제시켜야 합니다. 그러나 할당된 오브젝트는 라우팅 단계가 종료되면 자동으로 할당 해제됩니다.

FILEA와 FILEB의 첫 번째 멤버가 사전 할당되지 않았을 경우 배타적 읽기 허용 제한은 효력을 나타내지 못합니다. 파일을 사용 중에는 파일이 변경되지 않도록 파일을 사전 할당해야 합니다.

주: 하나의 오브젝트가 두 번 이상 할당된 경우(둘 이상의 할당 명령에 의해), 한번의 DLCOBJ 명령만으로는 이 오브젝트를 완전히 할당 해제할 수 없습니다. 각각의 할당 명령에 대해 할당 해제 명령이 각각 하나씩 필요합니다.

잠금 상태가 아니거나 할당에 필요한 특정 잠금 상태가 설정되어 있지 않은 오브젝트에 대해 DLCOBJ 명령이 발행되는 것은 오류가 아닙니다.

다음 예와 같이 오브젝트의 잠금 상태를 변경시킬 수 있습니다.

```
       PGM
ALCOBJ  OBJ((FILEX *FILE *EXCL)) WAIT(0)
CALL PGMA
ALCOBJ  OBJ((FILEX *FILE *EXCLRD))
DLCOBJ  OBJ((FILEX *FILE *EXCL))
CALL PGMB
DLCOBJ  OBJ((FILEX *FILE *EXCLRD))
    ENDPGM
```

파일 FILEX가 PGMA에 대해서는 배타적 잠금 상태로 할당되고 PGMB에 대해서는 배타적 읽기 허용 잠금 상태로 할당되어 있습니다.

파일 안에 자료 레코드를 할당할 때는 레코드 잠금을 사용할 수 있습니다. 시간종료가 발생하기 전에 프로그램이 파일을 기다리는 시간을 지정하려는 경우 파일 작성 명령의 WAITFILE 매개변수를 사용할 수 있습니다.

파일 작성 명령의 WAITRCD 매개변수는 레코드 잠금을 기다려야 하는 시간을 지정합니다. CRTCLS(클래스 작성) 명령의 DFTWAIT 매개변수는 다른 오브젝트를 기다리는 시간을 지정합니다. WAITRCD 매개변수의 설명에 대해서는 백업 및 회복  책을 참조하십시오.

오브젝트 잠금 상태 표시

WRKOBJLCK(오브젝트 잠금에 대한 작업) 명령이나 WRKJOB(작업에 대한 작업) 명령을 사용하면 오브젝트의 잠금 상태를 표시할 수 있습니다.

WRKOBJLCK 명령은 지정된 오브젝트에 대한 시스템 안의 모든 잠금 상태 요구를 표시합니다. 이 명령은 보류 잠금 상태 및 대기 중인 잠금 상태를 모두 표시합니다. 데이터베이스 파일의 경우 WRKOBJLCK 명령은 레코드 레벨이 아닌 파일 레벨(오브젝트 레벨)의 잠금 상태를 표시합니다. 예를 들면, 갱신을 위하여 데이터베이스 파일을 여는 경우 파일의 잠금 상태가 표시되나, 파일 안의 레코드의 잠금 상태는 표시되지 않습니다. 데이터베이스 파일 멤버에 대한 잠금 상태도 WRKOBJLCK 명령을 사용하여 표시할 수 있습니다.

WRKJOB 명령을 사용하면 작업 표시 메뉴에서 잠금 옵션을 선택할 수 있습니다. 이 옵션은 지정된 활동 작업에 대한 미해결 잠금 상태 요구, 작업에 의해 보류된 잠금 상태, 작업이 기다리고 있는 잠금 상태를 모두 표시합니다. 그러나 작업이 데이터베이스 레코드 잠금을 기다리고 있을 경우에는 오브젝트 잠금 상태 화면에 나오지 않습니다.

다음 명령은 논리 파일 ORDFILL에 대한 시스템의 모든 잠금 상태 요구를 표시합니다.

```
WRKOBJLCK  OBJ(QGPL/ORDFILL)  OBJTYPE(*FILE)
```

다음은 명령 사용 후 표시되는 화면입니다.

```
                                                  오브젝트 잠금에 대한 작업
                                                      시스템:   SYSTEM01
 오브젝트:   ORDFILL        라이브러리:   QGPL          유형:   *FILE-LGL

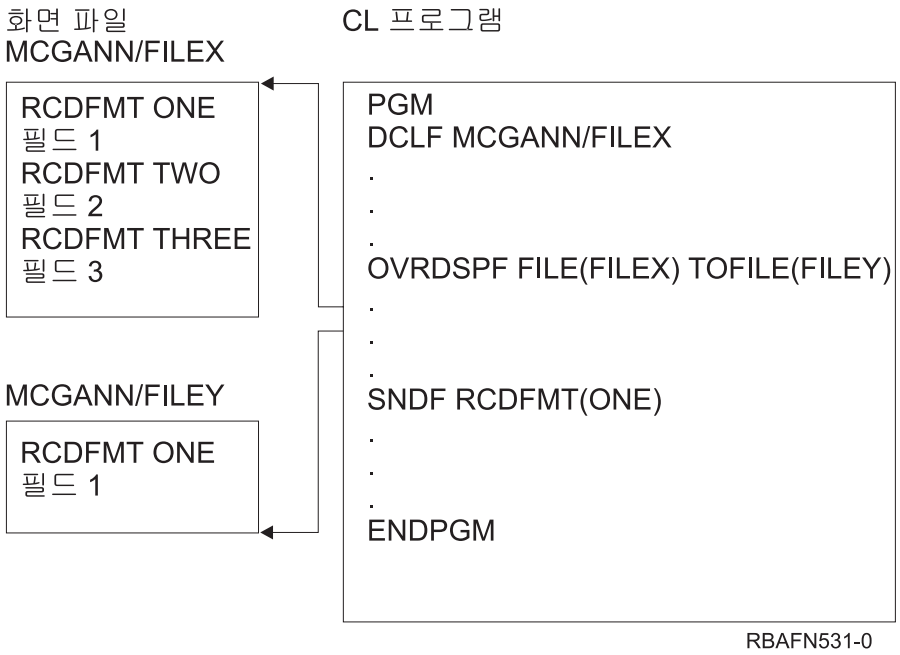
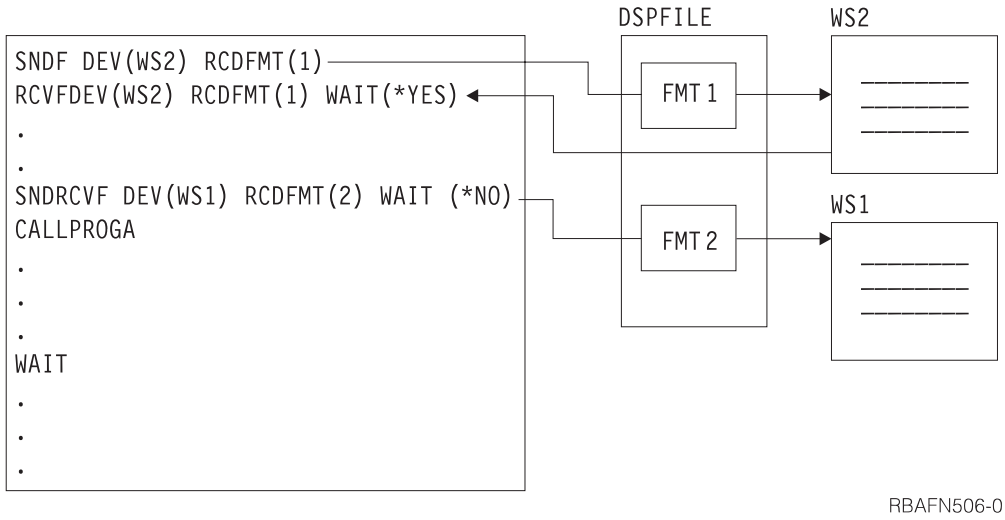
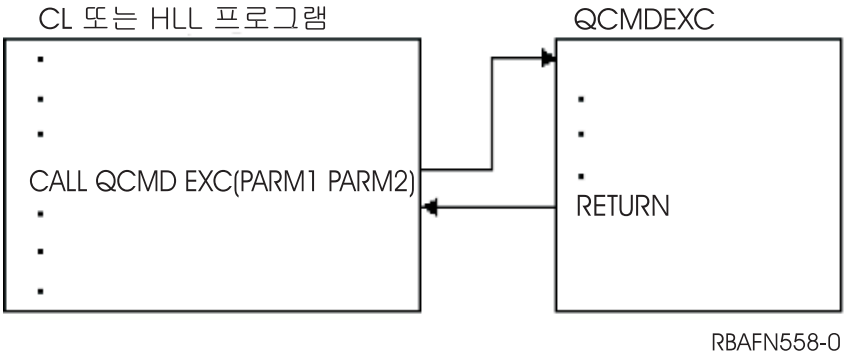
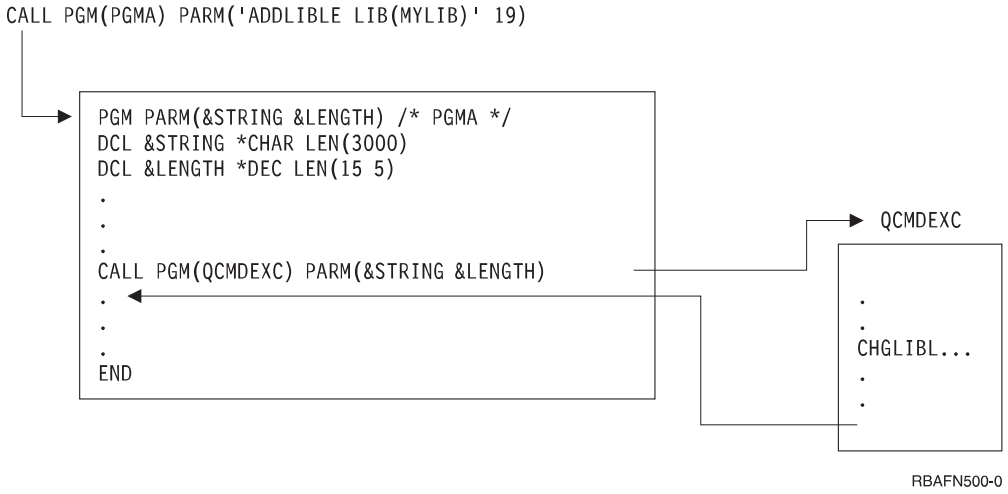
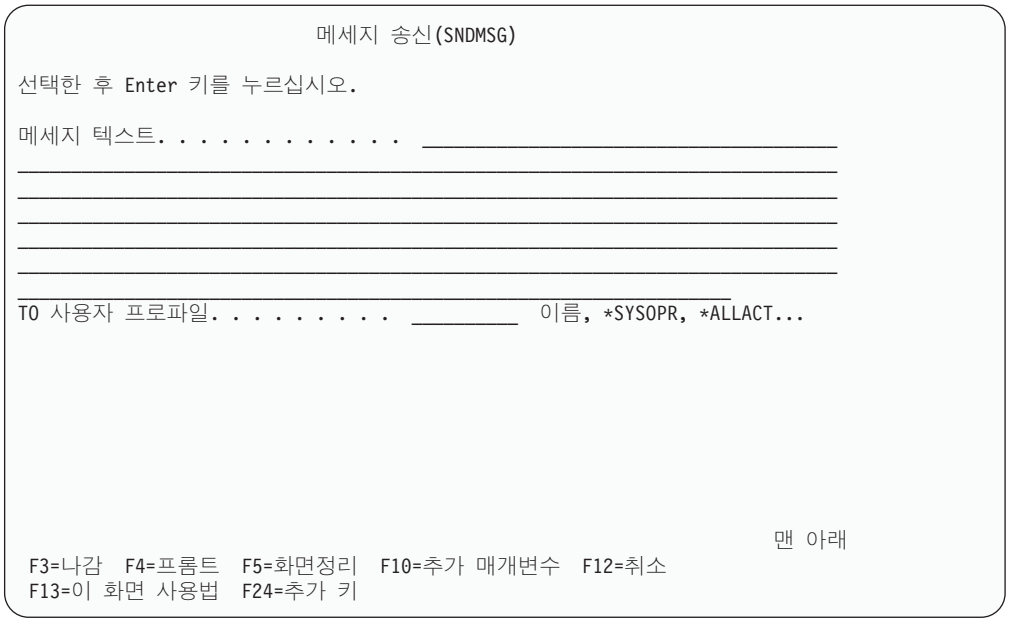
옵션을 입력한 후 Enter 키를 누르십시오.
   4=작업 종료   5=작업에 대한 작업   8=작업 잠금에 대한 작업

 Opt      작업           사용자          잠금        상태      범위      스레드
  _       WORKST04       QSECOFR         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                         *SHRRD      HELD      *JOB
                                                                      계속...
 F3=나감    F5=화면정리    F6=멤버 잠금에 대한 작업    F12=취소
```

제 5 장 CL 프로시듀어와 프로그램 안의 오브젝트에 대한 작업

CL 프로그램 안의 오브젝트 액세스

CL 프로그램 명령과 프로시듀어에서 오브젝트를 참조하는 규칙은 개별적으로 처리되는 (프로그램 안에 없는) 명령에서 오브젝트를 참조하는 규칙과 같습니다. 오브젝트명은 규정된 이름일 수도 있고 아닐 수도 있습니다. 라이브러리 리스트를 탐색하여 규정되지 않은 오브젝트명을 찾아내십시오.

CL 프로시듀어와 프로그램에서 참조되는 오브젝트의 대부분은 이들을 참조하는 명령이 수행될 때까지 액세스되지 않습니다. 오브젝트의 이름(*라이브러리/이름*)을 규정하려면 그 오브젝트를 참조하는 명령이 수행될 때 오브젝트가 지정된 라이브러리에 있어야 합니다. 그러나 오브젝트는 프로그램을 작성할 때 해당 라이브러리에 없어도 됩니다. 이는 대부분의 오브젝트가 단순히 실행 시 위치에만 기초를 둔 CL 소스문에 규정될 수 있다는 것을 나타냅니다. 이에 대한 예외는 170 페이지의 『예외: 명령 정의, 파일 및 프로시듀어 액세스』 부분에서 설명합니다.

CL 소스문에 오브젝트명을 규정하지 않고 대신 라이브러리 리스트(*LIB/이름)를 참조할 경우 모든 오브젝트에 대해 이러한 실행 시 고려사항을 주의하지 않아도 됩니다. 컴파일 시 라이브러리 리스트를 참조할 경우 오브젝트는 수행 시 라이브러리 리스트 안의 어느 라이브러리나 될 수 있습니다. 이것은 서로 다른 라이브러리에 중복되는 이름의 오브젝트가 없을 때 가능합니다. 라이브러리 리스트를 사용하면 프로시듀어 작성과 명령 처리 간에 오브젝트를 다른 라이브러리로 이동시킬 수 있습니다.

오브젝트를 참조하는 명령이 실행되기까지 오브젝트는 없어도 됩니다. 이로 인해 CL 프로그램은 컴파일 시 PAYROLL 프로그램이 없어도 성공적으로 컴파일됩니다.

```
PGM /*TEST*/
DCL...
MONMSG...
.
.
.
CALL PGM(QGPL/PAYROLL)
.
.
.
ENDPGM
```

실제로, TEST 프로그램을 활성화할 때에는 PAYROLL이 없어도 되지만 CALL 명령을 처리할 때에는 필요합니다. 따라서 호출된 프로그램이 호출 프로그램 안에서 CALL 명령 바로 전에 작성됩니다.

```
PGM /*TEST*/
DCL...
.
.
.
MONMSG
.
.
.
CRTCLPGM PGM(QGPL/PAYROLL)
CALL PGM(QGPL/PAYROLL)
.
.
.
ENDPGM
```

(CRTCLPGM)이나 (CRTDTAARA)와 같은 작성 명령에 있어서, 컴파일 시 또는 수행 시 액세스되는 오브젝트의 경우 작성되는 오브젝트가 아니라 작성 명령 정의라는 점에 유의하십시오. 작성 명령을 사용 중이면, 작성 명령 정의가 컴파일 시 명령을 규정하는 데 사용되는 라이브러리에 있어야 합니다. (또한 *LIBL을 사용하는 경우에는 라이브러리 리스트상의 한 라이브러리에 있어야 합니다.)

예외: 명령 정의, 파일 및 프로시듀어 액세스

명령 정의나 파일을 참조하는 소스문으로부터 CL 프로그램을 작성할 때는 두 가지 요구사항이 있습니다.

- 프로그램을 작성할 때 오브젝트가 있어야 합니다.
- 오브젝트를 참조하는 명령이 수행될 때 오브젝트가 있어야 합니다.

이것은 DCLF(파일 선언) 명령을 사용하는 경우 파일을 참조하는 프로그램을 작성하기 전에 파일을 작성해야 한다는 의미입니다.

명령 정의 액세스

명령 정의에 대한 액세스는 프로그램 작성 시간 중이나 명령 실행 시 발생합니다. 구문 검사를 허용하려면 그 명령을 사용하는 프로그램을 작성하는 동안 명령이 있어야 합니다. 작성 시 명령이 규정되는 경우 작성하는 동안 참조되는 라이브러리에 명령이 있어야 하며 처리 시에도 같은 라이브러리에 있어야 합니다. 명령이 라이브러리에 규정된 것이 아닌 경우에는 작성 시 및 수행되는 동안에 라이브러리 리스트상의 어느 라이브러리에나 있어도 됩니다.

다음과 같은 경우 프로그램에 명령어가 규정되어야 합니다.

- 프로그램 실행 시 라이브러리 리스트를 통해 명령의 정의를 액세스할 수 없는 경우
- 수행 시 명령의 특정 인스턴스를 예상하고 있으나 여러 개의 명령 정의가 동일한 이름으로 존재할 경우

명령어는 프로그램을 수행했 때와 시스템이 프로그램을 작성했을 때 서로 같아야 합니다. 명령을 참조하는 프로그램이 작성된 후 명령어가 변경되면 오류가 발생합니다. 이

는 프로그램이 수행될 때 명령을 찾을 수 없기 때문입니다. 그러나 명령 매개변수의 디폴트 값이 변경되면 그 명령이 실행될 때 새로운 디폴트 값이 사용됩니다. 명령을 재작성하지 않고도 명령에서 변경할 수 있는 속성에 대한 자세한 정보는 iSeries Information Center의 **프로그래밍** 범주에 나오는 *CL* 주제에서 CHGCMD(명령 변경) 명령 설명을 참조하십시오.

파일 액세스

컴파일러는 DCLF(파일 선언) 명령을 가진 프로그램 모듈을 컴파일할 때 파일에 액세스합니다. 파일을 사용하는 CL 모듈이나 OPM 프로그램을 컴파일할 때에는 그 파일이 반드시 있어야 합니다. 모듈을 사용하는 프로그램이나 서비스 프로그램을 작성할 때에는 파일이 없어도 됩니다.

소스 파일을 작성하기 전에 소스 파일에 자료 서술 스펙(DDS)을 입력하십시오. DDS는 레코드 형식과 레코드 안의 필드를 설명하며, 시스템이 CRTDSPF(화면 파일 작성) 명령을 통해 파일 오브젝트를 작성하기 위해 이 정보를 컴파일합니다.

주: DDS로부터 사용자는 서로 다른 유형의 파일을 작성할 수 있으며, 각 유형마다 자신의 명령이 있습니다. CRTPF(실제 파일(PF) 작성) 및 CRTLF(논리 파일 작성)는 파일을 작성하기 위해 CL 프로그램과 프로시듀어에서 사용할 수 있는 두 가지 명령의 예입니다.

DDS에 서술된 필드들은 입력 필드 또는 출력 필드(또는 둘 다)일 수 있습니다. 프로그램이나 모듈을 컴파일할 때 시스템이 CL 프로그램이나 프로시듀어에 있는 필드를 변수로 선언합니다. CL 프로그램은 이러한 변수들을 통해 표시되는 자료를 조작합니다.

실제 파일을 작성하기 위해 DDS를 사용하지 않을 경우에는 시스템이 전체 레코드를 포함시키기 위해 CL 변수를 선언합니다. 이 변수는 파일명과 동일하며 변수의 길이도 파일의 레코드 길이와 동일합니다.

CL 프로그램과 프로시듀어는 특정 CL 명령을 제외한 화면 파일과 데이터베이스 파일 이외의 다른 어떤 유형의 파일에 있는 자료도 조작할 수 없습니다.

파일을 작성한 후 DDS를 삭제할 수 있지만 바람직한 것은 아닙니다. 시스템이 파일을 참조하는 CL 프로그램이나 모듈을 컴파일한 후에는 파일을 삭제할 수 있습니다. 이것은 RCVF(파일 수신)와 같이 파일을 참조하는 명령이 프로그램에서 처리될 때 있던 파일이 제공되는 경우에 해당됩니다.

명령 정의에 대해 지금까지 설명한 규정된 이름에 관한 규칙은 파일에도 적용됩니다. 파일에 대해 자세히 알려면 173 페이지의 『CL 프로시듀어 안의 파일에 대한 작업』 부분을 참조하십시오.

프로시듀어 액세스

CALLPRC(바인드된 프로시듀어 호출) 명령에 의해 지정되는 프로시듀어는 그 프로시듀어를 참조하는 모듈을 작성할 때 없어도 됩니다. 프로시듀어를 사용하는 프로그램이나 서비스 프로그램을 작성하기 위해 시스템에 그 프로시듀어가 반드시 있어야 할 필요는 없습니다. 호출된 프로시듀어는 다음과 같은 곳에 있을 수 있습니다.

- CRTPGM(프로그램 작성) 또는 CRTSRVPGM 명령의 MODULE 매개변수에 지정된 모듈
- BNDSRVPGM 매개변수에 지정된 서비스 프로그램. 서비스 프로그램은 수행 시 사용이 가능해야 합니다.
- CRTPGM 명령이나 CRTSRVPGM 명령의 BNDDIR 매개변수에 지정된 바인딩 디렉토리에 나오는 서비스 프로그램이나 모듈. 바인딩 디렉토리와 모듈은 수행 시 사용이 가능해야 할 필요는 없습니다.

오브젝트 존재 검사

프로그램에서 오브젝트를 사용할 때는 먼저 그 오브젝트가 존재하는지 그리고 그것을 사용하는 데 필요한 권한이 있는지를 검사하여 판별하십시오. 이것은 기능이 한 번에 둘 이상의 오브젝트를 사용할 때 유용합니다.

오브젝트 존재 여부를 검사하려는 경우 CHKOBJ(오브젝트 검사) 명령을 사용할 수 있습니다. 이 명령은 프로시듀어나 프로그램의 어느 곳에서도 사용할 수 있습니다. CHKOBJ 명령의 형식은 다음과 같습니다.

```
CHKOBJ  OBJ(library-name/object-name)  OBJTYPE(object-type)
```

기타 옵션 매개변수를 사용하면 오브젝트 권한 부여 검증을 수행할 수 있습니다. 권한 부여를 검사하고 파일을 열려면 조작 권한 및 자료 권한을 모두 검사해야 합니다.

이 명령이 실행될 때 오브젝트 검사 결과를 보고하기 위해 시스템이 프로그램이나 프로시듀어를 요구하는 메세지를 송신합니다. 사용자는 이 메세지를 모니터하여 원하는 대로 처리할 수 있습니다. 예를 들어, 다음과 같은 경우가 있습니다.

```
                  CHKOBJ  OBJ(OELIB/PGMA)  OBJTYPE(*PGM)
          MONMSG  MSGID(CPF9801)  EXEC(GOTO NOTFOUND)
          CALL OELIB/PGMA
          .
          .
          .
NOTFOUND:  CALL FIX001 /*PGMA Not Found Routine*/
    ENDPGM
```

이 예에서 MONMSG 명령은 단지 '오브젝트 없음'이라는 이탈 메세지만 검사합니다. CHKOBJ 명령이 송신할 수 있는 모든 메세지의 리스트를 보려면 CHKOBJ 명령에 대한 온라인 도움말 정보를 참조하십시오. 메세지 모니터에 관한 추가 정보에 대해서는 60 페이지의 『MONMSG(메세지 모니터) 명령 사용』, 제 7 장 그리고 제 8 장을 참조하십시오.

CHKOBJ 명령은 오브젝트를 할당하지 않습니다. 많은 어플리케이션이 충분하지 않은 기능의 존재 검사를 사용하는 경우 어플리케이션이 오브젝트를 할당할 수 있어야 합니다. ALCOBJ(오브젝트 할당) 명령으로 존재 검사와 할당 기능을 모두 수행할 수 있습니다.

특정 테이프나 디스켓이 드라이브에 놓여 있으며 준비 완료 상태인지를 확인하려면 CHKTAP(테이프 검사) 또는 CHKDKT(디스켓 검사) 명령을 사용하십시오. 이 명령들은 또한 CL 프로그램에서 사용자가 모니터할 수 있도록 이탈 메세지를 제공합니다.

CL 프로시듀어 안의 파일에 대한 작업

화면 파일과 데이터베이스 파일의 두 가지 파일 유형이 CL 프로시듀어와 프로그램에서 지원됩니다. 사용자는 워크스테이션으로 표시 화면을 송신하고 프로시듀어나 프로그램에서 사용하기 위해 워크스테이션에서 입력을 수신하거나 프로시듀어나 프로그램에서 사용하기 위해 데이터베이스 파일에서 자료를 읽을 수 있습니다.

주: 데이터베이스 파일은 DCLF 및 RCVF 명령을 통하여 CL 프로시듀어나 프로그램 안에서 사용할 수 있도록 만들어집니다.

CL 프로시듀어나 프로그램에서 파일을 사용하려면 다음과 같이 하십시오.

- DDS 소스로서 입력되는 필드와 조건을 식별하도록 표시장치 또는 데이터베이스 레코드를 형식화하십시오. 데이터베이스 파일에 DDS를 반드시 사용해야 하는 것은 아닙니다.
- CRTDSPF(화면 파일 작성) 명령, CRTPF(실제 파일 작성) 명령 또는 CRTLF(논리 파일 작성) 명령을 사용하여 파일을 작성하십시오. CL 프로시듀어나 프로그램은 서브파일(메세지 서브파일은 제외)을 지원하지 않습니다.
- 데이터베이스 파일인 경우 ADDPFM(실제 파일 멤버 추가) 명령 또는 ADDLFM(논리 파일 멤버 추가) 명령을 사용하여 멤버를 파일에 추가시키십시오. 멤버가 CRTPF 또는 CRTLF 명령에 의해 추가되었을 경우에는 이 과정이 필요없습니다. 프로시듀어나 프로그램이 처리될 때는 파일이 멤버를 갖고 있어야 하지만, 프로시듀어나 프로그램이 작성될 때는 파일이 멤버를 갖고 있을 필요가 없습니다.
- DCLF 명령을 사용하여 CL 프로시듀어에서 파일을 참조하고, CL 소스 안의 적절한 자료 조작을 위해 CL 명령의 레코드 형식을 참조하십시오.
- CL 모듈을 작성하십시오.
- 프로그램이나 서비스 프로그램을 작성하십시오.

CL 프로시듀어에서는 최대 5개의 화면 파일이나 데이터베이스 파일을 참조할 수 있습니다. 데이터베이스 파일이나 화면 파일에 대한 지원은 같은 명령이 사용되는 경우와 유사합니다. 그러나 다음과 같은 몇 가지 차이점이 있습니다.

- 다음 사항들은 CL 프로시듀어나 프로그램과 함께 사용되는 데이터베이스 파일에만 적용됩니다.
  - 하나의 레코드 형식으로 된 데이터베이스 파일만이 CL 프로시듀어나 프로그램에서 사용될 수 있습니다.
  - 파일은 실제 파일 또는 논리 파일이 될 수 있으며, 논리 파일은 여러 실제 파일 멤버에 대해 정의할 수 있습니다.
  - RCVF 명령으로는 입력 조작만 할 수 있습니다. RCVF 명령의 WAIT 및 DEV 매개변수는 데이터베이스 파일에서 사용할 수 없습니다. 또한 데이터베이스 파일에서는 SNDF, SNDRCVF 및 ENDRCV 명령을 사용할 수 없습니다.
  - CL 프로시듀어나 프로그램에서 참조되는 실제 파일을 작성하는 데 DDS가 반드시 필요한 것은 아닙니다. 실제 파일을 작성하는 데 DDS를 사용하지 않는 경우 파일은 파일과 이름이 같은 레코드 형식을 가지며, 파일과 이름 및 파일의 레코드 길이가 같은 레코드 형식 안에 하나의 필드를 가집니다(CRTPF 명령의 RCDLEN 매개변수).
  - 모듈이나 프로그램을 작성하기 위해 파일을 작성할 경우에는 파일이 멤버를 갖고 있을 필요가 없습니다. 그러나 파일이 프로그램에 의해 처리될 때는 반드시 멤버를 가지고 있어야 합니다.
  - 첫 번째 RCVF 명령이 처리될 때만 입력을 위해 파일이 열립니다. 이때는 파일이 반드시 존재해야 하며 멤버를 가지고 있어야 합니다.
  - 프로시듀어나 OPM 프로그램이 리턴하거나 파일 끝에 도달할 때까지 파일은 열린 채로 있습니다. 파일의 끝에 도달하면 메세지 CPF0864가 CL 프로시듀어나 프로그램으로 송신되며 이에 대해 더 이상의 파일에 대한 조작이 불가능해집니다. 프로시듀어나 프로그램에서는 이 메세지를 모니터하여 파일 끝에 도달할 경우 적합한 조치를 수행해야 합니다.
- 다음 사항들은 CL 프로시듀어나 프로그램과 함께 사용되는 화면 파일에만 적용됩니다.
  - 화면 파일은 레코드 형식을 최대 99개까지 가질 수 있습니다.
  - 화면 파일에는 모든 자료 조작 명령(SNDF, SNDRCVF, RCVF, ENDRCV, WAIT)을 사용할 수 있습니다.
  - 화면 파일은 DDS로 정의되어야 합니다.
  - 첫 번째 SNDF, SNDRCVF 또는 RCVF 명령이 처리될 때 화면 파일은 입력과 출력용으로 모두 열립니다. 프로시듀어나 OPM 프로그램이 리턴될 때 파일은 열린 채로 있습니다.

주: 첫 번째 송신 또는 수신이 발생할 때까지, 두 유형의 파일은 열리지 않습니다. 이로 인해, 사용될 파일은 프로시듀어나 프로그램 수행 시 작성될 수 있고 대체는 첫 번째 송신 또는 수신 이전에 수행될 수 있습니다. 그러나 모듈이나 프로그램을 컴파일하려면 파일이 존재해야 합니다.

화면의 형식은 DDS 레코드 형식에 의해 식별됩니다. 각 레코드 형식에는 필드(입력, 출력, 입/출력), 조건/인디케이터 및 상수가 들어 있습니다. 화면 파일에는 여러 개의 레코드 형식을 입력할 수 있습니다. 화면 파일명, 레코드 형식명 및 필드명은 CL 프로시듀어나 프로그램이 요구하지 않더라도, 다른 HLL이 이를 요구할 수 있으므로 고유해야 합니다.



어플리케이션 화면 프로그래밍 책에 있는 방법 또는 화면 설계 유틸리티(SDA)를 사용하여 화면 파일에 있는 레코드와 필드에 대한 DDS 소스를 입력할

하나의 CL 프로시듀어나 프로그램은 자료 조작 명령이라는 여러 개의 명령을 사용할 수 있습니다. 이 명령을 통해 사용자들은 화면 파일을 참조함으로써 자료를 장치 화면으로 송신하고 장치 화면으로부터 수신하게 됩니다. 이 명령을 사용하여 데이터베이스 파일로부터 레코드를 읽기 위해 데이터베이스 파일을 참조할 수 있습니다. 자료 조작 명령은 다음과 같습니다.

- DCLF(파일 선언). 프로시듀어나 프로그램에서 사용될 화면 파일이나 데이터베이스 파일을 정의합니다. 파일 안의 필드가 프로시듀어나 프로그램에서 사용될 변수로 자동으로 선언됩니다.
- SNDF(파일 송신). 표시장치에 자료를 송신합니다.
- RCVF(파일 수신). 표시장치나 데이터베이스로부터 자료를 수신합니다.
- SNDRCVF(파일 송/수신). 자료를 표시장치로 송신한 후 표시장치로부터의 입력을 요구하고 선택적으로 자료를 수신합니다.
- OVRDSPF(화면 파일로 대체). 화면 파일과 함께 프로시듀어나 프로그램이 사용하는 파일의 수행 시 대체를 허용합니다.
- OVRDBF(데이터베이스 파일로 대체). 데이터베이스 파일과 함께 프로시듀어나 프로그램이 사용하는 파일의 수행 시 대체를 허용합니다.

이들 명령으로 실행 프로그램이 DDS가 제공한 표시 기능을 사용하는 장치 화면과 통신할 수 있고, 데이터베이스 파일로부터 레코드를 읽을 수 있습니다. DDS는 메뉴를 작성하는 기능을 제공하고, 대부분의 CL 어플리케이션의 특성인 기본 어플리케이션 위주

의 자료 요구를 수행하는 기능을 제공합니다.



표시장치나 레코드의 필드들은 파일에 대한 DDS에서 식별됩니다. CL 프로시듀어나 프로그램 순서에서 필드를 사용하려면 DCLF 명령이 CL 프로시듀어나 프로그램에서 파일을 참조해야 합니다. 참조 결과 파일 안의 필드와 인디케이터가 프로시듀어나 프로그램에서 자동으로 변수로 선언됩니다. CL 명령에서 어떤 방법으로든 이들 변수를 사용할 수 있으나 변수의 1차 목적은 정보를 송신하고 표시장치로부터 정보를 수신하는 것입니다. DCLF 명령은 수행 시 사용이 불가능합니다.

표시장치 형식과 필드에 대한 옵션은 장치 파일에서 지정되며 인디케이터를 사용하여 제어됩니다. DDS와 CL 지원에서 인디케이터 값은 99개까지 사용할 수 있습니다. 인디케이터 변수는 DCLF 명령에서 참조되는 장치 파일 레코드 형식으로 나타나는 각 인디케이터에 대해 &IN01-&IN99의 이름을 가진 논리 변수의 형식으로 CL 프로시듀어나 프로그램 안에 선언됩니다. 인디케이터를 사용하여 필드를 표시하고 자료 관리 표시 기능을 제어하며, 장치 화면의 프로시듀어나 프로그램에 응답 정보를 제공할 수 있습니다. 인디케이터는 데이터베이스 파일에는 사용되지 않습니다.

CL 프로시듀어에서 파일 참조

파일 안의 각 필드에 대해 변수가 선언될 수 있도록 CL 모듈이나 프로그램을 작성할 때 DCLF 명령을 컴파일하는 동안 파일이 액세스됩니다.

컴파일 시 파일명을 규정했으면, 파일은 수행 시 그 라이브러리 안에 들어 있어야 합니다. 컴파일 시 라이브러리 리스트를 사용했으면 수행 시 파일은 라이브러리 리스트상의 라이브러리 안에 있어야 합니다.

CL 프로시듀어에서 파일 열기 및 닫기

CL 지원을 사용할 때, 참조된 파일이 첫 번째 송신, 수신 또는 송/수신 조작 시 내재적으로 열립니다. 열린 화면 파일은 그것을 연 프로시듀어나 OPM 프로그램이 제어를 리턴시키거나 전송할 때까지 열린 상태로 남습니다. 열린 테이타 베이스 파일은 파일 끝에 이르거나 그것을 연 프로시듀어나 OPM 프로그램이 제어를 리턴시키거나 전송할 때 닫힙니다. 일단 데이터베이스 파일이 닫히면, 같은 프로시듀어나 OPM 프로그램을 호출하는 동안에는 다시 열리지 않습니다.

데이터베이스 파일을 열 때 먼저 OVRDBF 명령을 사용하여 다른 멤버(MBR 매개변수)를 지정하지 않는 한, 파일 안의 첫 번째 멤버가 열립니다. 오류로 인해 프로시듀어나 OPM 프로그램이 종료하면 파일이 닫힙니다. 열린 파일은 그것을 연 프로시듀어나 OPM 프로그램이 종료할 때까지 열린 상태로 있습니다. 이로 인해, 실행 프로시듀어와 프로그램 사이에서 쉽게 열린 자료 경로를 공유할 수 있습니다. 한 프로시듀어나 프로그램에서 하나의 파일을 열 수 있습니다. 그러고 나면 다음 중 하나의 조건에서 열린 자료 경로를 다른 프로시듀어나 프로그램과 공유할 수 있습니다.

- 파일이 SHARE(*YES) 속성으로 작성되었거나 이 속성을 갖도록 변경된 경우
- SHARE(*YES)를 지정한 파일에 대한 대체가 유효한 경우

이와 같은 방법으로 두 프로시듀어나 프로그램 사이에서 파일들을 공유할 수 있습니다. 시스템이 열린 자료 경로를 공유할 때 사용할 수 있는 기능의 자세한 설명은 온라인 도움말을 참조하십시오. 또한 IBM에서는 CRTDSPF, CRTPF, CRTLF 명령의 SHARE 매개변수에 관한 설명을 온라인으로 제공합니다. iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오. CL 프로시듀어나 OPM 프로그램에서 열린 화면 파일은 입/출력에 대해 모두 항상 열립니다. CL 프로시듀어나 OPM 프로그램에서 열린 데이터베이스 파일은 입력에 대해서만 열립니다.

파일을 사용하는 CL 프로시듀어나 프로그램에서 RCLRSC(자원 재생) 명령에는 LVI(*CALLER)을 지정하지 않아야 합니다. LVL(*CALLER)을 지정했다면, 프로시듀어나 OPM 프로그램이 열었던 모든 파일이 즉시 닫히고, 파일에 액세스하려는 어떤 시도도 이상 종료됩니다.

파일 선언

DCLF(파일 선언) 명령은 화면 파일이나 데이터베이스 파일을 CL 프로시듀어나 프로그램에 선언하는 데 사용됩니다. 테이프, 프린터 및 혼합 파일과 같은 파일을 선언하는 데에는 DCLF 명령을 사용할 수 없습니다. 하나의 CL 프로시듀어나 OPM 프로그램에서 최대 5개의 파일을 선언할 수 있습니다. DCLF 명령의 매개변수는 다음과 같습니다.

```
DCLF   FILE(library-name/file-name)
    RCDFMT(record-format-names)
    OPNID(open_id_name)
```

모듈이나 프로그램이 컴파일되기 위해서는 파일이 반드시 존재해야 합니다.

프로시듀어나 프로그램에서 화면 파일을 사용하려면 DDS에서 입/출력 필드를 지정해야 합니다. 이 필드는 프로시듀어나 프로그램에서 변수로 처리됩니다. DCLF 명령이 처리될 때 CL 컴파일러가 각 필드에 대해 CL 변수를 선언하고 파일의 각 레코드 형식에서 옵션 인디케이터를 선언합니다. CL 변수명은 필드의 경우 필드명 앞에 앰퍼샌드(&)를 사용하고 옵션 인디케이터의 경우 인디케이터 앞에 &IN을 사용합니다.

OPNID(열린 파일 ID) 매개변수를 사용하여 여러 파일을 선언할 수 있도록 선언된 파일의 인스턴스를 고유하게 식별할 수 있습니다. OPNID 매개변수를 사용할 때 CL 변수명은 필드의 경우 앞에 앰퍼샌드(&), OPNID 값 및 밑줄(_)이 오는 필드명입니다. 옵션 인디케이터의 경우 CL 변수명은 앞에 앰퍼샌드(&), OPNID 값, 밑줄 및 『IN』이 오는 인디케이터입니다.

예를 들어, INPUT이라는 필드와 인디케이터 10이 DDS에서 선언되면 DCLF 명령이 자동으로 그것을 &INPUT 및 &IN10으로 선언합니다. CL 모듈이나 프로그램이 컴파일될 때 이 선언이 수행됩니다. 하나의 명령에서 레코드 형식명을 최대 50개까지 지정할 수 있지만, 그 중 하나라도 변수여서는 안됩니다. 하나의 데이터베이스 파일에 대해서는 하나의 레코드 형식만 지정할 수 있습니다.

라이브러리 MCGANN에서 화면 파일 CNTRLDSP를 작성하기 위해 다음 DDS가 사용되는 경우를 보겠습니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R MASTER
     A                                      CA01(01 'F1 RESPONSE')
     A            TEXT         300      2  4
     A            RESPONSE      15   1  8  4 BLINK
     A
     A
```

이 경우 화면 파일에서 세 개의 변수 &IN01, &TEXT 및 &RESPONSE를 사용할 수 있습니다. 이 화면 파일을 참조하는 CL 프로시듀어에서는 DCLF 소스문만 입력하십시오.

```
   DCLF MCGANN/CNTRLDSP
```

컴파일러가 이 명령문을 확장하여 모든 화면 파일 변수를 개별적으로 선언합니다. 컴파일러 리스트에서 확장된 선언은 다음과 같습니다.

```
  •
  •
  •
  00500- DCLF(MCGANN/CNTRLDSP)
04/02/03
         QUALIFIED FILE NAME - MCGANN/CNTRLDSP
            RECORD FORMAT NAME - MASTER
          CL VARIABLE     TYPE     LENGTH     PRECISION (IF *DEC)
           &IN01          *LGL        1
           &TEXT          *CHAR     300
           &RESPONSE      *CHAR      15
  •
  •
  •
```

DCLF 소스문에 OPNID 매개변수가 포함된 경우

```
DCLF MCGANN/CNTRLDSP OPNID(OPENID1)
```

컴파일러 리스트에서 확장된 선언은 다음과 같습니다.

```
  •
  •
  •
 00500- DCLF FILE(MCGANN/CNTRLDSP) OPNID(OPENID1)                    04/02/03
        QUALIFIED FILE NAME - MCGANN/CNTRLDSP
           RECORD FORMAT NAME - MASTER
             CL VARIABLE               TYPE     LENGTH     PRECISION     TEXT
              &OPENID1_IN01            *LGL        1
              &OPENID1_TEXT            *CHAR     300
              &OPENID1_RESPONSE        *CHAR      15
  •
  •
  •
```

화면 파일에 대한 자료 송/수신

CL 프로시듀어와 프로그램에서 자료를 송/수신하기 위해 화면 파일과 함께 사용할 수 있는 명령들은 SNDF, RCVF 및 SNDRCVF뿐입니다.

SNDF 명령을 실행할 때 시스템이 레코드 형식의 출력 또는 입/출력 필드와 관련된 변수의 내용을 형식화합니다. 또한 시스템이 표시장치로 그 내용을 송신합니다. 이것은 RCVF 명령을 실행할 때와 유사합니다. 화면에서 레코드 형식의 입력 또는 입/출력 필드와 관련된 필드값이 해당 CL 변수에 놓입니다.

SNDRCVF 명령은 CL 변수의 내용을 표시장치로 송신한 후 RCVF 명령과 같은 기능을 수행하여 표시장치로부터 갱신된 필드를 얻습니다. CL이 존 10진수를 지원하지 않는다는 점에 유의해야 합니다. 그러므로 존 10진수로 정의된 화면 파일의 필드들이 CL 프로시듀어나 프로그램에서 *DEC 필드로 정의됩니다. *DEC 필드는 내부적으로 팩 십진수로 지원되며, CL 명령은 필요에 따라 팩 및 존 자료 유형을 변환시킵니다. 표시 위치가 서로 일치하기 때문에 화면 파일에서 중첩되는 필드들은 중첩되지 않는 CL 변수로 개별적으로 정의됩니다. 부동 소수점 자료가 들어 있는 레코드 형식은 CL 프로시듀어나 프로그램에서는 사용할 수 없습니다.

주: 워크스테이션에 대한 SNDRCVF 또는 RCVF 명령이 WAIT(*NO)를 나타내는 경우 시스템은 자료를 수신할 때 WAIT 명령을 사용합니다. INVITE DDS 키워드가 들어 있는 레코드 형식으로 SNDF 명령이 발행된 경우에도 마찬가지입니다.

메세지 서브파일을 제외한 서브파일 레코드를 송신하거나 수신하려는 경우에는 수행 시 오류가 발생합니다. DDS의 화면 파일에 지정된 대부분의 기능을 사용할 수 있으나 일부 기능(예: 변수 시작 행 번호 사용)은 사용할 수 없습니다. CL 프로시듀어와 프로그램에 있는 메세지와 서브파일에 대해 더 자세히 알려면 제 8 장을 참조하십시오.

다음 예는 대표적인 오퍼레이터 메뉴를 작성하고 SNDRCVF 명령을 사용하여 자료를 송/수신하기 위해 필요한 단계입니다. 메뉴는 다음과 같습니다.

```
  Operator Menu

   1. Accounts Payable
   2. Accounts Receivable
   90. Signoff

Option:
```

먼저, 다음의 DDS 소스를 입력하십시오. 레코드 형식은 MENU이며, OPTION은 입력이 가능한 필드입니다. OPTION 필드는 DSPATR(MDT)을 사용합니다. 따라서 오퍼레이터의 입력이 없더라도 시스템이 이 필드에 대해 유효한 값을 검사합니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R MENU
     A                                  1  2'Operator Menu'
     A                                  3  4'1. Accounts Payable'
     A                                  5  4'2. Accounts Receivable'
     A                                  5  4'90. Signoff'
     A                                  7  2'Option'
     A            OPTION         2Y 01   + 2VALUES(1 2 90) DSPATR(MDT)
     A
     A
```

CRTDSPF 명령을 입력하여 화면 파일을 작성할 수 있습니다. CL 프로그래밍에서는 화면 파일명(INTMENU)이 레코드 형식명(MENU)과 같지만 OS/400용 RPG와 같은 일부 언어에는 해당되지 않습니다.

화면 설계 유틸리티(SDA: Screen Design Aid)를 사용하여 화면 파일을 작성할 수도 있습니다.

그 다음 CL 소스를 입력하여 메뉴를 수행하십시오.

이 메뉴의 CL 소스는 다음과 같습니다.

```
                  PGM /* OPERATOR MENU */
         DCLF INTMENU
BEGIN:   SNDRCVF RCDFMT(MENU)
         IF COND(&OPTION *EQ 1) THEN(CALL ACTSPAYMNU)
         IF COND(&OPTION *EQ 2) THEN(CALL ACTSRCVMNU)
         IF COND(&OPTION *EQ 90) THEN(SIGNOFF)
         GOTO BEGIN
    ENDPGM
```

이 소스가 컴파일되면 DCLF 명령이 자동으로 프로시듀어 안의 입력 필드 OPTION을 CL 변수로 선언합니다.

SNDRCVF 명령의 디폴트는 WAIT(*YES)입니다. 즉, 프로그램이 입력을 수신할 때까지 프로그램이 대기합니다.

메뉴 제어를 위한 CL 프로그램 작성

다음 예는 메뉴를 표시하고 제어하기 위한 CL 프로시듀어 작성 방법입니다. 메뉴를 작성하고 제어하는 기타 방법은 Application Display Programming  책을 참조하십시오.

다음 예에서는 구매부의 일반 메뉴 표시를 제어하고 메뉴에서 선택된 옵션에 기초하여 어떤 HLL 프로시듀어를 호출할 것인지 결정하는 CL 프로시듀어 ORD040C를 보여줍니다. 프로시듀어가 표시장치에 메뉴를 보여줍니다.

구매부의 일반 메뉴는 다음과 같습니다.

```
  Order Dept General Menu

  1 Inquire into customer file
  2 Inquire into item file
  3 Customer name search
  4 Inquire into orders for a customer
  5 Inquire into an existing order
  6 Order entry
 98 End of menu

Option:
```

화면 파일 ORD040C에 대한 DDS는 다음과 같습니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* MENU ORDO40CD ORDER DEPT GENERAL MENU
     A
     A          R MENU                        TEXT('General Menu')
     A                                   1  2'Order Dept General Menu'
     A                                   3  3'1 Inquire into customer file'
     A                                   4  3'2 Inquire into item file'
     A                                   5  3'3 Customer name search'
     A                                   6  3'4 Inquire into orders for a custom+
     A                                       er'
     A                                   7  3'5 Inquire into existing order'
     A                                   8  3'6 Order Entry'
     A                                   9  2'98 End of menu'
     A                                  11  2'Option'
     A            RESP           2Y001 11 10VALUES(1 2 3 4 5 6 98)
     A                                       DSPATR(MDT)
     A
     A
```

ORD040C에 대한 소스 프로시듀어는 다음과 같습니다.

```
       PGM /* ORD040C Order Dept General Menu */
       DCLF FILE(ORD040CD)
START: SNDRCVF RCDFMT(MENU)
       SELECT
         WHEN (&RESP=1) THEN(CALLPRC CUS210)  /* Customer inquiry */
         WHEN (&RESP=2) THEN(CALLPRC ITM210)  /* Item inquiry     */
         WHEN (&RESP=3) THEN(CALLPRC CUS220)  /* Cust name search */
         WHEN (&RESP=4) THEN(CALLPRC ORD215)  /* Orders by cust   */
         WHEN (&RESP=5) THEN(CALLPRC ORD220)  /* Existing order   */
         WHEN (&RESP=6) THEN(CALLPRC ORD410C) /* Order entry      */
         WHEN (&RESP=98) THEN(RETURN)         /* End of Menu      */
       ENDSELECT
       GOTO START
     ENDPGM
```

DCLF 명령은 SNDRCVF 명령이 처리될 때 구매부의 일반 메뉴를 형식화하기 위해 시스템에 필요한 필드 속성이 어느 파일에 들어 있는지를 표시합니다. 레코드 형식이 SNDF, RCVF 또는 SNDRCVF 명령에서 사용될 경우 시스템은 지정된 파일의 해당 레코드 형식에 각 필드에 대한 변수를 자동으로 선언합니다. 자동으로 선언된 각 필드의 변수명은 앞에 앰퍼샌드(&)가 오는 필드명입니다. 예를 들면, ORD040C에서 응답 필드 RESP의 변수명은 &RESP입니다.

이 메뉴의 조작에 관한 이외의 유의사항은 다음과 같습니다.

SNDRCVF 명령은 메뉴를 표시장치로 송신하고 선택된 옵션을 표시장치로부터 수신하기 위해 사용됩니다.

메뉴에서 선택한 옵션이 98일 경우 ORD040C는 이를 호출한 프로시듀어로 리턴됩니다.

ELSE문은 응답을 상호 배타적 대체 방법으로 처리하기 위해 필요합니다.

주: 이 메뉴는 CALL 명령을 사용하여 수행됩니다. GO 명령을 사용하여 그러한 메뉴를 실행하는 것에 관해서는 Application Display Programming 책을 참조하십시오.

CL 프로그램에서 화면 파일 대체

OVRDSPF(화면 파일로 대체) 명령을 사용하면 CL 프로시듀어나 프로그램에서 명명된 화면 파일을 대체하거나 기존 화면 파일의 특정 매개변수를 변경할 수 있습니다. 이것은 모듈이나 프로그램이 컴파일된 이후로 이름이 변경되었거나 이동되었던 파일에 있어서 특히 유용합니다.

OVRDSPF 명령의 초기 매개변수는 다음과 같습니다.

```
OVRDSPF   FILE(overridden-file-name) TOFILE(new-file-name)
          DEV(device-name)
```

OVRDSPF 명령은 모듈이나 프로그램이 작성될 때 DCLF 명령에서 지정된 파일이 화면 파일인 경우에만 CL 프로시듀어나 프로그램이 참조한 파일에 유효합니다. 프로그램이 실행될 때 사용된 파일은 모듈이나 프로그램이 작성될 때 참조한 파일과 같은 유형이어야 합니다.

대체시킬 파일을 열기 전에 OVRDSPF 명령을 먼저 실행시켜야 합니다. 열기는 첫 번째 송신 또는 수신 명령에 의해 이루어집니다. 시스템이 다음 조건이 있는지 찾기 위해 파일을 대체합니다.

- OVRDSPF 명령이 있는 프로시듀어나 프로그램이 파일을 엽니다.
- CALLPRC 명령을 사용하여 제어를 전송한 다른 프로시듀어에서 파일을 엽니다.
- CALL 명령을 사용하여 제어를 전송한 다른 프로그램에서 파일을 엽니다.

다른 파일로 대체할 때 SNDF, RCVF 또는 SNDRCVF 명령에서 참조한 레코드 형식 명만 대체 파일에 필요합니다. 다음 그림에서 화면 파일 FILEY에는 레코드 형식 TWO 또는 THREE가 필요없습니다.



RBAFN531-0

원래 파일명과 대체 파일명에 대해 참조된 레코드 형식이 동일한 필드 정의와 인디케이터명을 같은 순서로 가지고 있는지를 확인하도록 하십시오. LVLCHK(*NO)를 지정할 경우 예기치 않은 결과가 발생할 수 있습니다.

OVRDSPF 명령이 적용될 때는 SNDF, RCVF, SNDRCVF 명령의 DEV 매개변수에 대해 고려해야 할 사항이 또 있습니다. 이 명령들의 DEV 매개변수에 *FILE이 지정되면 시스템은 대체된 파일에 대한 올바른 장치로 자동으로 조작을 지시합니다. RCVF, SNDF 또는 SNDRCVF 명령의 DEV 키워드에 특정 장치가 지정되면 다음 중 하나가 발생할 수 있습니다.

- 하나의 장치 화면 파일을 사용하고 있을 때 화면 파일이 RCVF, SNDF 또는 SNDRCVF 명령에 지정되지 않은 장치로 대체되면 오류가 발생합니다.
- 복수 장치 화면 파일을 사용하고 있을 때 RCVF, SNDF 또는 SNDRCVF 명령에 지정된 장치가 OVRDSPF 명령에 지정된 장치 중 하나가 아니면, 오류가 발생합니다.

복수 장치 화면 파일에 대한 작업

시스템의 정상 조작 모드는 워크스테이션 사용자가 사인 온한 후 대화식 작업의 리퀘스터가 되도록 하는 것입니다. 각 사용자가 프로시듀어의 화면 파일을 포함하여 프로시듀어의 논리 사본을 사용하므로 많은 사용자가 동시에 이것을 수행할 수 있습니다. 이와 같은 용도로 각 리퀘스터가 개별 작업을 호출합니다. 이것은 복수 장치 화면 사용으로 간주되지 않습니다.

한 명의 리퀘스터가 호출한 하나의 작업이 하나의 화면 파일을 통해 여러 개의 표시장치와 통신할 때 복수 장치 화면 구성이 발생합니다. CL 프로시듀어에 의해서는 하나의 화면 파일만 처리될 수 있는 반면, 화면 파일 또는 장치 화면 파일 안의 서로 다른 레코드 형식들은 여러 개의 장치 화면으로 송신될 수 있습니다. 복수 장치 화면 파일에 주로 사용되는 명령은 다음과 같습니다.

- ENDRCV(수신 종료). 이 명령은 만족스럽지 못한 입력에 대한 요구를 취소합니다.
- 대기(WAIT). 명령에 WAIT(*NO)가 지정되었을 경우 이전의 여러 RCVF나 SNDRCVF 명령에 의해 또는 이전의 여러 SNDF 명령에 의해, INVITE DDS 키워드를 포함한 레코드 형식으로, 사용자 자료를 요구한 장치 화면으로부터 입력을 받아들입니다.

복수 장치 화면 파일을 사용할 경우 장치명은 화면 파일이 작성될 때 CRTDSPF 명령의 DEV 매개변수에 지정되어야 하고, 화면 파일이 변경될 때 CHGDSPF 명령의 DEV 매개변수 또는 대체 명령의 DEV 매개변수에 지정되어야 합니다. 또한 장치의 갯수는 CRTDSPF 명령의 MAXDEV 매개변수에 지정된 숫자 이하여야 합니다.

복수 장치 화면 구성은 SNDRCVF 및 RCVF 명령에 영향을 미치며, 사용자는 WAIT나 ENDRCV 명령을 사용해야 할 경우도 있습니다. RCVF나 SNDRCVF 명령이 복수 장치 화면에 사용되면 디폴트 값 WAIT(*YES)는 입력 가능 필드가 DEV 매개변수에 명명된 장치로부터 프로그램으로 리턴할 때까지 더 이상 처리되지 않습니다. 응답이 지연될 수 있으므로 WAIT(*NO)를 지정하여 수신 작업이 완료되기를 기다리지 않고 프로시듀어나 프로그램이 다른 명령을 계속 수행하도록 하는 것이 좋습니다.

RCVF 또는 SNDRCVF 명령을 사용하면서 WAIT(*NO)를 지정하면 WAIT 명령이 처리될 때까지 CL 프로시듀어나 프로그램이 계속 실행됩니다.

DDS INVITE 키워드가 있는 레코드 형식의 SNDF 명령을 사용하는 것은 WAIT(*NO)가 지정된 SNDRCVF 명령을 사용하는 것과 같습니다. DDS INVITE 키워드는 SNDRCVF 및 RCVF 명령의 경우에는 무시됩니다.

자료 레코드에 액세스하려면 WAIT 명령을 발행해야 합니다. 자료를 전혀 사용할 수 없다면, 장치 화면에서 자료를 수신할 때까지 또는 CRTDSPF, CHGDSPF 또는 OVRDSPF 명령상의 화면 파일에 대한 WAITRCD 매개변수에서 지정된 제한 시간이 전달될 때까지 처리가 일시중단됩니다. 제한 시간이 지나면 메세지 CPF0889가 발행됩니다.

WAIT는 ENDJOB, ENDSYS, PWRDWNSYS 및 ENDSBS 명령의 제어 옵션으로 취소될 작업에 의해서도 충족될 수 있습니다. 이 경우 메세지 CPF0888이 발행되며 자료는 리턴되지 않습니다. 선행되는 수신 요구(예: RCVF . . . WAIT(*NO))가 없이 WAIT 명령이 실행되면 처리 오류가 발생합니다.

다음은 대표적인 복수 장치 화면 구성(코드 포함)입니다.



RBAFN506-0

위의 예에서 처음 두 개의 명령은 디폴트가 사용된 일반적인 순서를 보여줍니다. 처리는 WS2에서 수신 조작이 완료되기를 기다립니다. WS2가 DEV 매개변수에 지정되었으므로 다른 스테이션으로부터 이전의 미해결 요구(표시되지는 않음)가 충족되었더라도 RCVF 명령은 WS2가 응답할 때까지 수행되지 않습니다.

그러나 WAIT(*NO)가 지정된 SNDRCVF 명령은 WS1로부터의 응답을 기다리지 않습니다. 그 대신, 처리가 계속되고 PROGA가 호출됩니다. 그런 후 미해결 요구가 워크스테이션에 의해 충족되거나 기능이 시간종료될 때까지 WAIT 명령 위치에서 처리가 중단됩니다.

다음은 WAIT 명령의 형식입니다.

```
WAIT DEV(CL-variable-name)
```

DEV 매개변수가 지정될 경우 CL 변수명은 응답한 장치의 이름입니다. (디폴트는 *NONE입니다.) 여러 수신 요구(예: RCVF. . . WAIT(*NO))가 있는 경우 이 변수는 WAIT 명령을 발견한 후 응답하는 첫 번째 장치명을 사용하여 처리를 계속합니다. 수신된 자료는 장치 화면의 필드와 관련된 프로그램 변수에 놓입니다.

WAIT(*YES)가 지정된 RCVF 명령은 특정한 장치로부터의 자료를 기다리는 데 사용될 수 있습니다. 수신 요구를 시작하는 조작과 RCVF 명령에는 모두 같은 레코드 형식명이 지정되어야 합니다.

어떤 경우에는 여러 개의 수신 요구가 미해결 상태이더라도, 특정한 장치 화면으로부터의 응답이 없으면 더 이상 처리되지 않습니다. 다음 예에서는 세 개의 명령이 WAIT(*NO)를 지정하고 있으나 WS3이 응답하기까지는 처리가 레이블 LOOP에서 더 이상 진행될 수 없습니다.

```
                PGM
          .
          .
          .
          SNDF DEV(WS1) RCDFMT(ONE)
          SNDF DEV(WS2) RCDFMT(TWO)
          SNDRCVF DEV(WS3) RCDFMT(THREE) WAIT(*NO)
          RCVF DEV(WS2) RCDFMT(TWO) WAIT(*NO)
          RCVF DEV(WS1) RCDFMT(ONE) WAIT(*NO)
          CALL...
          CALL...
          .
          .
          RCVF DEV(WS3) RCDFMT(THREE) WAIT(*YES)
LOOP:     WAIT DEV(&WSNAME)
          MONMSG CPF0882 EXEC(GOTO REPLY)
          .
          .
          .
          GOTO LOOP
REPLY:    CALL...
          .
          .
          .
     ENDPGM
```

또한 CL 프로시듀어와 프로그램도 ENDRCV 명령을 지원합니다. 이 명령으로 아직 충족되지 않은 입력 요구를 취소할 수 있습니다. 또한 SNDF나 SNDRCVF 명령도 아직 충족되지 않은 입력 요구를 취소시킵니다. 그러나 SNDF나 SNDRCVF 명령이 처리되었을 때 자료 사용이 가능했으면, 메세지 CPF0887이 송신됩니다. 이 경우 자료가 WAIT 명령 또는 RCVF 명령에 의해 수신되거나 SNDF 또는 SNDRCVF 명령이 처리되기 전에 ENDRCV 명령을 사용하여 입력 요구를 명백하게 취소시켜야 합니다.

데이터베이스 파일로부터 자료 수신

데이터베이스 파일로부터 자료를 수신하기 위해 사용할 수 있는 유일한 명령은 RCVF 명령입니다.

RCVF 명령을 수행하면 파일 액세스 경로상의 그 다음 레코드가 읽히고, 데이터베이스 레코드 형식에 정의된 필드값이 해당 CL 변수에 놓입니다. CL이 존 10진수나 2진수를 지원하지 않는다는 점에 유의하십시오. 그러므로 존 10진수나 2진수로 정의된 데이터베이스 파일의 필드는 CL 프로시듀어나 프로그램에서 *DEC 필드로 정의됩니다. *DEC 필드는 내부적으로 팩 십진수로 지원되며, RCVF 명령은 필요에 따라 존 십진수와 2진수를 팩 십진수로 변환시킵니다. 부동 소수점 자료가 들어 있는 데이터베이스 파일은 CL 프로시듀어나 프로그램에서 사용될 수 없습니다.

파일의 끝에 이르면, 메세지 CPF0864가 프로시듀어나 OPM 프로그램으로 송신됩니다. 레코드 형식에 대해 선언된 CL 변수는 이 메세지가 송신될 때 RCVF 명령 처리에 의해 변경되지 않습니다. 사용자는 이 메세지를 모니터하여 파일의 끝이 되면 적절한 조치를 수행해야 합니다. 파일의 끝에 도달한 후 추가로 RCVF 명령의 수행을 시도하면 메세지 CPF0864가 다시 송신됩니다.

CL 프로시듀어나 프로그램 안의 데이터 베이스 파일 대체

OVRDBF(데이터베이스 파일로 대체) 명령을 사용하면 CL 프로시듀어나 프로그램에서 명명된 데이터베이스 파일을 대체하거나 기존 데이터베이스 파일의 특정 매개변수를 변경할 수 있습니다. 이것은 프로시듀어나 프로그램이 작성된 이후 이름이 변경되거나 이동된 파일에 특히 유용합니다. 이는 첫 번째 멤버를 제외한 파일 멤버에 액세스하는 데도 사용될 수 있습니다.

다음은 OVRDBF 명령의 초기 매개변수입니다.

```
OVRDBF FILE(overridden-file-name) TOFILE(new-file-name)
       MBR(member-name)
```

OVRDBF 명령은 모듈이나 프로그램이 작성될 때 DCLF 명령에서 지정된 파일이 데이터베이스 파일인 경우에만 CL 프로시듀어나 프로그램이 참조한 파일에 유효합니다. 프로그램이 처리될 때 사용된 파일은 모듈이나 프로그램이 작성될 때 참조된 파일과 유형이 같아야 합니다.

OVRDBF 명령은 대체될 파일이 열려 사용되기 전에 처리되어야 합니다(RCVF 명령의 첫 번째 사용으로 파일이 열림). 파일이 OVRDBF 명령이 들어 있는 프로시듀어나 OPM 프로그램에서 열리거나 CALL 명령이 제어를 전송한 다른 프로그램에서 열리거나 CALLPRC 명령이 제어를 전송한 다른 프로시듀어에서 열리는 경우 파일이 대체됩니다.

다른 파일로 대체할 경우 대체 파일은 하나의 레코드 형식만 가져야 합니다. DDS에서 정의된 복수 레코드 형식을 가진 논리 파일은 이 파일이 단 하나의 실제 파일 멤버에

대해 정의되는 경우 사용할 수 있습니다. DDS에서 정의된 단 하나의 레코드 형식을 가진 논리 파일은 둘 이상의 실제 파일 멤버에 대해 정의할 수 있습니다. 형식명은 프로그램 작성 시 참조된 형식명과 같지 않아도 됩니다. 대체 파일의 자료 형식은 원래 파일의 형식과 같아야 합니다. LVLCHK(*NO)를 지정할 경우 예기치 않은 결과가 발생할 수 있습니다.

표시 명령으로부터의 출력 파일 참조

많은 IBM 표시 명령들을 사용하면 명령에서 나오는 출력을 데이터베이스 파일에 위치시킬 수 있습니다(OUTFILE 매개변수). 이 파일의 자료는 직접 CL 프로시듀어나 프로그램으로 수신되어 처리될 수 있습니다.

다음 CL 프로시듀어는 두 개의 매개변수(사용자명과 라이브러리명)를 허용합니다. 프로시듀어는 라이브러리 안의 모든 프로그램, 파일 및 자료 영역의 이름을 판별하고 지정된 사용자에게 정상적인 권한을 부여합니다.

```
     PGM PARM(&USER &LIB)
     DCL &USER *CHAR 10
     DCL &LIB *CHAR 10
(1)  DCLF QSYS/QADSPOBJ
(2)  DSPOBJD OBJ(&LIB/*ALL) OBJTYPE(*FILE *PGM *DTAARA) +
     OUTPUT(*OUTFILE) OUTFILE(QTEMP/DSPOBJD)
(3)  OVRDBF QADSPOBJ TOFILE(QTEMP/DSPOBJD)
(4)  READ:  RCVF
(5)  MONMSG CPF0864 EXEC(RETURN)  /* EXIT WHEN END OF FILE REACHED  */
(6)  GRTOBJAUT OBJ(&ODLBNM/&ODOBNM) OBJTYPE(&ODOBTP) +
     USER(&USER) AUT(*CHANGE)
     GOTO READ                    /*GO BACK FOR NEXT RECORD*/
     ENDPGM
```

(1) 선언된 파일인, QSYS의 QADSPOBJ는 DSPOBJD 명령이 사용하는 IBM 제공 파일입니다. 이 파일은 1차 파일로서 출력 파일을 작성할 때 명령에 의해 참조됩니다. CL 컴파일러는 이 파일을 참조하여 레코드 형식을 결정하고 레코드 형식 안의 필드에 대한 변수를 선언합니다.

(2) DSPOBJD 명령은 QTEMP 라이브러리에 DSPOBJD라는 파일을 작성합니다. 이 파일은 QADSPOBJ 파일과 같은 형식을 가집니다.

(3) OVRDBF 명령은 DSPOBJD 명령에 의해 작성된 파일에 대한 QADSPOBJ(선언 파일)를 대체합니다.

(4) RCVF 명령은 DSPOBJD 파일로부터 레코드를 읽습니다. 레코드 안의 필드값은 해당 CL 변수에 복사되며, 이 변수는 DCLF 명령에 의해 내재적으로 선언됩니다. OVRDBF 명령이 사용되었기 때문에 QSYS/QADSPOBJ 대신 파일 QTEMP/DSPOBJD가 읽힙니다. (파일 QSYS/QADSPOBJ는 읽히지 않습니다.)

(5) 메세지 CPF0864가 모니터됩니다. 이는 파일의 끝에 도달해서 프로시듀어가 호출 프로시듀어로 제어를 리턴시킴을 나타냅니다.

(6) GRTOBJAUT 명령은 RCVF 명령에 의해 읽히는 오브젝트명, 라이브러리명 및 오브젝트 유형에 대한 변수를 사용하여 처리됩니다.

제 6 장 확장 프로그래밍

이 장에서는 확장 프로그래밍에 대해 다음과 같은 내용을 설명합니다.

- 고급 언어(HLL: High-Level Language) 프로그램에서 호출할 수 있는 특수 기능(CL 프로그램 포함)
- 프로그램 소스를 입력하기 위한 프롬팅 및 프로그래머 메뉴 사용

고급 기능 명령 처리에 대한 정보는 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

이 장의 마지막 부분에 몇 개의 프로그램 샘플이 나옵니다.

QCAPCMD 프로그램 사용

QCAPCMD(명령 처리) API는 명령 스트링에 대한 명령 분석기 처리를 수행합니다. 이 API를 통해 사용자는 다음을 수행할 수 있습니다.

- 명령 스트링을 실행하기 전에 구문을 검사합니다.
- 명령 프롬트 및 변경된 명령 스트링을 수신합니다.
- 고급 언어에서 명령을 사용합니다.
- 명령을 위한 도움말을 표시합니다.

QCAPCMD API에 대한 정보는 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

QCMDEXC 프로그램 사용

QCMDEXC(명령 실행)는 하나의 명령을 수행하는 IBM 제공 프로그램입니다. 이 명령은 다른 명령을 활성화하기 위해서 다음과 같이 사용됩니다.

- 고급 언어(HLL) 프로그램 안으로부터
- CL 프로시듀어 안으로부터
- 컴파일 시 어느 명령이 수행될 것인지 또는 어떤 매개변수가 사용될 것인지 등이 알려지지 않은 프로그램으로부터

QCMDEXC 프로그램은 HLL 또는 CL 프로시듀어 또는 프로그램에서 호출됩니다. 실행될 명령은 CALL 명령의 매개변수로서 전달됩니다.



명령이 실행된 후에는 제어가 고급 언어(HLL), CL 프로시듀어 또는 프로그램으로 리턴합니다.

명령은 프로그램에 없었던 것처럼 수행됩니다. 그러므로 명령이 값을 CL 변수로 리턴시킬 수 없기 때문에 변수가 명령에 사용될 수 없습니다. 뿐만 아니라, CL 프로시듀어나 프로그램에서만 사용할 수 있는 명령은 QCMDEXC 프로그램을 사용하여 실행할 수 없습니다. 다음은 QCMDEXC 프로그램을 호출하는 형식입니다.

```
CALL PGM(QCMDEXC) PARM(command command-length)
```

실행하려는 명령을 첫 번째 매개변수에 문자 스트링으로 입력하십시오. 반드시 명령 라이브러리를 지정해야 합니다.

```
CALL PGM(QCMDEXC )  PARM('QSYS/CRTLIB LIB(TEST)' 22)
```

명령에 공백이 포함되어 있는 경우에는 반드시 어포스트로피로 묶어야 한다는 것을 기억하십시오. 문자 스트링의 최대 길이는 6000자입니다. 분리문자(어포스트로피)는 스트링의 일부로 계산하지 않습니다. PARM 매개변수의 두 번째 값으로 지정되는 길이는 명령으로 전달되는 문자 스트링의 길이입니다. 길이는 소수 자릿수가 5인 길이 15의 팩 10진값입니다.

그러므로 라이브러리 리스트를 대체하기 위해 QCMDEXC 프로그램을 호출하려면 다음과 같이 입력해야 할 것입니다.

```
CALL PGM(QCMDEXC) PARM('CHGLIBL LIBL(QGPL NEWLIB QTEMP)' 31)
```

이 명령문을 HLL이나 CL 프로그램 안에 코딩하여 프로그램이 수행될 때 라이브러리 리스트를 대체할 수 있습니다. 이와 같은 방식으로 사용되는 경우 QCMDEXC 프로그램은 실행 시 융통성을 제공하지 않습니다.

실행 시 융통성은 다음에 의해 제공될 수 있습니다.

1. 매개변수 리스트에 있는 상수에 대해 변수를 대체합니다.
2. HLL 또는 CL 프로그램을 호출할 때 변수에 값을 지정합니다.

다음 예를 보십시오.



그림 2. 호출 프로그램의 예

QCMDEXC 프로그램에 두 번째 매개변수로 전달되는 명령 길이는 전달될 명령 스트링의 최대 길이입니다. 명령 스트링이 인용 스트링으로서 전달되면 명령 길이는 정확히 인용 스트링의 길이가 됩니다. 명령 스트링이 변수로 전달되면 명령 길이는 CL 변수의 길이입니다. 명령 길이를 변수 내 명령 스트링의 실제 길이로 축소할 수도 있지만 반드시 그렇게 할 필요는 없습니다.

모든 명령이 QCMDEXC 프로그램을 사용하여 수행될 수 있는 것은 아닙니다. QCMDEXC 프로그램에 대한 호출로 전달되는 명령은 호출이 수행되는 현재 환경(대화식 또는 일괄처리) 안에서 유효한 것이어야 합니다. 다음은 유효한 명령이 될 수 없습니다.

- 입력 스트림 제어 명령(BCHJOB, ENDBCHJOB, DATA)
- CL 프로그램 안에서만 사용할 수 있는 명령

호출에서 CL 명령을 QCMDEXC 프로그램으로 전달할 수 있는지 여부를 알려면 iSeries Information Center **프로그래밍** 범주의 *CL* 섹션에 나와 있는 명령 문서를 참조하십시오. 명령을 실행할 수 있는 환경을 확인하려면 Information Center 문서 파일의 시작 부분에서 CL 명령에 대한 '실행 허용 위치' 값을 찾으십시오. DSPCMD(명령 표시) 명령을 사용해서도 명령을 사용할 수 있는 위치를 확인할 수 있습니다.

CL 명령 앞에 물음표(?)를 놓거나 프롬트를 요구하거나 대화식 작업에서 QCMDEXC를 호출할 때 선택 프롬팅을 사용할 수 있습니다.

명령이 QCMDEXC 프로그램을 통해 처리되는 동안 오류가 발견되면 이탈 메세지가 송신됩니다. MONMSG(메세지 모니터) 명령을 사용하면 CL 프로시듀어나 프로그램에서 이탈 메세지를 모니터할 수 있습니다. 메세지 모니터에 대해 자세히 알려면 제 7 장과 제 8 장을 참조하십시오.

구문 오류가 발견되면 메세지 CPF0006이 송신됩니다. 명령의 처리 과정에서 오류가 발견되면 명령에 의해 송신되는 이탈 메세지가 QCMDEXC 프로그램에 의해 리턴됩니다. CL 프로시듀어와 프로그램에 들어 있는 명령에서 메세지를 모니터하는 것과 같은 방법으로, QCMDEXC 프로그램을 통해 명령 실행 메세지를 모니터하십시오.

고급 언어 프로그램이 호출 오류를 처리하는 방법에 대한 정보는 해당 고급 언어 참조 서적을 참조하십시오.

DBCS 자료가 있는 QCMDEXC 프로그램 사용

QCMDEXC를 사용하여 2바이트 문자 세트(DBCS: Double Byte Character Set) 입력 자료가 명령과 함께 입력되도록 요구할 수 있습니다. 다음은 QCMDEXC가 2바이트 자료를 프롬트하기 위해 사용되는 명령 형식입니다.

```
CALL QCMDEXC ('command' command-length IGC)
```

QCMDEXC 프로그램의 세 번째 매개변수인 IGC는 시스템에서 2바이트 자료를 받아들일 것을 지시합니다. 예를 들어, 다음의 CL 프로그램은 사용자에게 메세지를 2바이트 텍스트로 입력할 것을 요구합니다. 그런 후 시스템은 다음과 같은 메세지를 송신합니다.

```
      PGM
CALL QCMDEXC ('?SNDMSG' 7 IGC)
  ENDPGM
```

다음은 시스템 메세지에 관한 설명입니다.

- ? 문자는 시스템에 SNDMSG(메세지 송신) 명령에 대한 명령 프롬트를 표시하도록 지시합니다.
- 값 7은 SNDMSG 명령에 의문 부호를 더한 길이입니다.
- IGC 값을 사용하면 2바이트 자료를 요구할 수 있습니다.

다음은 QCMDEXC 프로그램이 수행된 후의 화면입니다. 이 화면에서 2바이트 변환을 사용할 수 있습니다.



QCMDCHK 프로그램 사용

QCMDCHK는 하나의 명령에 대해 구문 검사를 수행하고 선택적으로 명령에 대해 프롬트하는 IBM 제공 프로그램입니다. 명령은 수행되지 않습니다. 프롬팅이 요구되면 프롬트를 통해 입력된 대로 갱신된 값을 가진 호출 프로시듀어나 프로그램으로 명령 스트링이 리턴됩니다. QCMDCHK 프로그램은 CL 프로시듀어나 프로그램 또는 고급 언어(HLL) 프로시듀어나 프로그램에서 호출될 수 있습니다.

일반적인 QCMDCHK의 사용은 다음과 같습니다.

- 명령을 사용자에게 프롬트한 후 나중에 처리하기 위해 명령을 보관합니다.
- 사용자가 지정한 옵션을 판별합니다.
- 처리된 명령을 기록합니다. 먼저, QCMDCHK를 프롬트하여 QCMDEXC를 수행한 다음, 처리된 명령을 기록합니다.

다음은 QCMDCHK의 호출 형식입니다.

```
CALL PGM(QCMDCHK) PARM(command  command-length)
```

QCMDCHK로 전달되는 첫 번째 매개변수는 검사되거나 프롬트될 명령이 들어 있는 문자 스트링입니다. 첫 번째 매개변수가 변수이고 프롬트가 요구되면 워크스테이션 사용자가 입력한 명령이 변수에 놓입니다.

두 번째 매개변수는 전달될 명령 스트링의 최대 길이입니다. 명령 스트링이 인용 스트링으로서 전달되면 명령 길이는 정확히 인용 스트링의 길이가 됩니다. 명령 스트링이 변수로 전달되면 명령 길이는 CL 변수의 길이입니다. 두 번째 매개변수는 소수 자릿수가 5인 길이 15의 팩 10진값이어야 합니다.

QCMDCHK 프로그램은 전달된 명령 스트링에 대한 구문 검사를 수행합니다. 이 프로그램은 필수 매개변수가 모두 코드화되어 있는지, 모든 매개변수가 허용 가능한 값을 가지고 있는지를 확인합니다. 이 프로그램은 처리 환경을 검사하지 않습니다. 즉, 명령이 일괄처리로만 허용되는지, 대화식으로만 허용되는지 또는 일괄처리나 대화식 CL 프로그램에 대해 모두 허용되는지에 대해 검사할 수 있습니다. QCMDCHK는 명령 정의문에 대해서는 검사할 수 없습니다.

명령에 구문 오류가 발견되면 메세지 CPF0006이 송신됩니다. 명령에 오류가 발생했는지를 알려면 이 메세지를 모니터해야 합니다. 메세지 CPF0006 앞에는 오류를 식별하는 하나 이상의 진단 메세지가 나옵니다. 다음 예에서는 CRTCLPGM 명령의 PGM 매개변수에 대해 값 123이 유효하지 않기 때문에 프로그램 안의 레이블 ERROR로 제어가 전달됩니다.

```
CALL QCMDCHK ('CRTCLPGM PGM(QGPL/123)' 22)
MONMSG CPF0006 EXEC(GOTO ERROR)
```

명령어 앞에 물음표를 사용하거나 명령 스트링 안의 여러 키워드명 앞에 선택 프롬트 문자를 사용하여 명령에 대한 프롬팅을 요구할 수 있습니다.

명령을 검사하고 프롬팅하는 동안 오류가 발견되지 않으면 갱신된 명령 스트링은 첫 번째 매개변수에 대해 지정된 변수에 놓입니다. 프롬트 요구 문자는 명령 스트링에서 제거됩니다. 다음이 그 예입니다.

```
DCL &CMD *CHAR 2000
 .
 .
CHGVAR &CMD '?CRTCLPGM'
CALL QCMDCHK (&CMD 2000)
```

QCMDCHK 프로그램의 호출이 수행되고 나면, 변수 &CMD에 프롬터를 통해 입력된 모든 값을 가진 명령 스트링이 포함됩니다. 이것은 다음과 같을 수 있습니다.

```
CRTCLPGM PGM(PGMA) SRCFILE(TESTLIB/SOURCE) USRPRF(*OWNER)
```

명령어 앞부분의 물음표는 제거됩니다.

QCMDCHK 프로그램에서 프롬팅이 요구될 때는 명령 스트링이 CL 변수로 전달되어야 합니다. 그렇지 않으면 갱신된 명령 스트링이 프로시듀어나 프로그램에 리턴되지 않습니다. 명령 스트링의 변수는 프롬터로부터 리턴되어 갱신된 명령 스트링이 들어가기에 충분할 정도의 길이를 가져야 합니다. 그렇지 않으면 메세지 CPF0005가 송신되고 명령 스트링이 들어 있는 변수는 변경되지 않습니다. 선택 프롬팅이 없으면, 프롬터는 사용자가 입력한 항목을 리턴시킵니다.

변수의 길이는 두 번째 매개변수의 값에 의해 결정되며, 이는 실제 변수 길이가 아닙니다. 다음 예에서는 변수가 적절한 길이로 선언되었으나 이 길이가 너무 짧아 갱신된 명령이 들어갈 수 없으므로 이탈 메세지 CPF0005가 송신됩니다.

```
DCL &CMD *CHAR 2000
 .
 .
CHGVAR &CMD '?CRTCLPGM'
CALL QCMDCHK (&CMD 9)
```

QCMDCHK가 실행되는 동안 F3 키나 F12 키를 눌러 프롬터 화면에서 나오면, 메세지 CPF6801이 QCMDCHK를 호출한 프로시듀어나 프로그램으로 송신되며, 명령 스트링이 들어 있는 변수는 변경되지 않습니다.

PARM, ELEM 또는 QUAL 명령 정의문에 PASSATR(*YES)이 지정되고 CHGCMDDFT 명령을 사용하여 디폴트 값이 변경되면 마치 디폴트 값이 아닌 사용자 지정 값인 것처럼 디폴트 값이 강조표시됩니다. 변경된 PARM, ELEM 또는 QUAL 명령 정의문의 디폴트 값이 원래의 디폴트 값으로 다시 변경되면 디폴트 값은 더 이상 강조표시되지 않습니다.

CL 프로그램이나 프로시듀어 안의 메세지 서브파일 사용

CL 프로시듀어와 프로그램에서는 메세지 서브파일이 유일하게 지원되는 서브파일 유형입니다. 서브파일 메세지 지원을 사용하려면 서브파일 메세지 제어 레코드를 사용하여 SNDF 또는 SNDRCVF 명령을 수행하십시오. DDS에서는 SFLPGMQ 자료를 제공하고 항상 SFLINZ가 활성화 상태를 유지하도록 만드십시오.

CL 프로시듀어와 프로그램에서 메세지 서브파일을 사용할 때는 프로시듀어나 프로그램을 명명해야 합니다. DDS에서 SFLPGMQ 키워드에는 *(별표)를 지정할 수 없습니다. 프로시듀어나 OPM 프로그램명을 지정하면 해당 프로시듀어나 프로그램의 메세지 대기행렬로 송신된 모든 메세지가 호출 메세지 대기행렬로부터 나와서 메세지 서브파일에 놓입니다. 현재 요구와 관련된 모든 메세지가 CALL 메세지 대기행렬로부터 메세지 서브파일에 놓입니다.

메세지 서브파일은 제어 프로시듀어나 프로그램이 하나 이상의 오류 메세지를 표시할 수 있도록 합니다.

수행 시 CL 명령의 사용자 변경 허용

대부분의 CL 프로시듀어와 프로그램에서 워크스테이션 사용자는 이 프로시듀어나 프로그램으로 전달되는 매개변수 값을 지정하거나 표시장치 프롬트에서 입력이 가능한 필드에 정보를 입력하는 방식으로 프로시듀어나 프로그램에 입력합니다.

또한 다음 방법으로 CL 프로시듀어나 프로그램에 입력하라는 프롬트를 워크스테이션 사용자에게 표시할 수도 있습니다.

- ?를 CL 프로시듀어나 프로그램 소스 안의 CL 명령 앞에 입력하는 경우 시스템이 CL 명령에 대한 프롬트를 표시합니다. 프로시듀어나 프로그램에는 사용자가 이미 지

정한 매개변수 값이 채워지며 워크스테이션 사용자는 이 값을 변경할 수 없습니다. 이 섹션의 뒤에 나오는『CL 프로시듀어 또는 프로그램 내에서 OS/400 프롬터 사용』부분을 참조하십시오.

- QCMDEXC 프로그램을 호출하여 선택 프롬팅을 요구하는 경우 시스템이 CL 명령에 대한 프롬트를 표시하지만 처리 시 CL 명령이 사용될 CL 프로그램 소스를 지정할 필요가 없습니다. QCMDEXC 프로그램에 대해 자세히 알려면 191 페이지의『QCMDEXC 프로그램 사용』부분을 참조하십시오.

CL 프로시듀어 또는 프로그램 내에서 OS/400 프롬터 사용

사용자는 CL 프로시듀어나 프로그램의 대화식 처리 안에서 프롬팅을 요구할 수 있습니다. 예를 들어, 다음 프로시듀어를 컴파일하여 실행할 수 있습니다.

```
         PGM
.
.
.
?DSPLIB
.
.
.
     ENDPGM
```

이 경우 프로그램을 처리하는 동안 DSPLIB(라이브러리 표시) 명령에 대한 프롬트가 화면에 표시됩니다. 사용자가 필수 매개변수의 값을 입력한 후 Enter 키를 누르면 DSPLIB 명령이 처리됩니다.

소스 프로시듀어에서 지정된 값은 오퍼레이터(또는 사용자)가 직접 변경할 수 없습니다. 예를 들어, 다음과 같은 경우가 있습니다.

```
         PGM
.
.
.
?SNDMSG TOMSGQ(WS01 WS02)
.
.
.
     ENDPGM
```

프로시듀어가 호출되고 SNDMSG(메세지 송신) 명령 프롬트가 나올 때 오퍼레이터(또는 사용자)가 MSG, MSGTYPE 및 RPYMSGQ 매개변수에 값을 입력할 수 있으나 TOMSGQ 매개변수에 대한 값을 변경할 수는 없습니다. 예를 들어, 오퍼레이터(또는 사용자)가 WS03을 추가하거나 WS02를 삭제할 수 없습니다. 이와 같은 제한의 예외 경우에 대해 자세히 알려면 203 페이지의『CL 프로시듀어나 프로그램에서 프롬팅되는 QCMDEXC』부분을 참조하십시오. 다음은 처리 시 CL 프로시듀어 안에서의 프롬트 사용에 대한 제한사항입니다.

- 프롬터가 CL 프로시듀어나 프로그램으로부터 호출되면 이 프롬트에서 변수명을 입력하거나 매개변수 값으로 표현식을 입력할 수 없습니다.
- IF, ELSE 또는 MONMSG 명령이 내장 명령에서는 프롬팅을 요구할 수 없습니다.

  맞는 예

```
IF (&A=5) THEN(DO)
             ?SNDMSG
 ENDDO
```

  틀린 예

```
IF (&A=5) THEN(?SNDMSG)
```

- 다음은 프롬팅을 사용할 수 없는 명령입니다.

CALL	**CALLPRC**	**CHGVAR**	**COPYRIGHT**
DCL	**DCLF**	**DO**	**DOFOR**
DOUNTIL	**DOWHILE**	**ELSE**	**ENDDO**
ENDPGM	**ENDRCV**	**ENDSELECT**	**GOTO**
IF	**ITERATE**	**LEAVE**	**MONMSG**
OTHERWISE	**PGM**	**RCVF**	**RETURN**
SELECT	**SNDF**	**SNDRCVF**	**WAIT**
WHEN			

- 일괄처리 작업에서는 프롬팅을 사용할 수 없습니다.

CL 소스 파일 멤버에서 명령에 프롬팅 요구(?)를 입력하면 명령에 관한 진단 메세지를 수신하고도 계속해서 정상적인 컴파일을 할 수 있습니다. 이 경우 사용자는 메세지를 읽어, 프로시듀어나 프로그램 실행 시 프롬트 화면을 통해 입력된 값으로 오류가 정정될 수 있는지를 확인해야 합니다.

대화식 환경에 있을 때는 앞에 나열된 명령을 제외한 권한이 있는 모든 명령에 대해 어떤 모드에서나 프롬트할 수 있습니다. 그러나 CL 프로시듀어나 프로그램을 처리하는 동안에는 프롬트시킬 수 없습니다. 따라서 사용자는 워크스테이션에 있는 동안 어떠한 명령에 대해서도 프롬트를 표시할 수 있으므로 다양한 명령과 해당 매개변수를 설명하는 매뉴얼을 수시로 참조해야 하는 불편함이 없어집니다.

명령이 실행되는 동안 프롬트된 명령을 취소하기 위해 F13 키 또는 F12 키를 누르면 이탈 메세지(CPF6801)가 CL 프로시듀어나 프로그램으로 송신됩니다. 이 메세지는 CL 프로시듀어나 프로그램에서 MONMSG 명령을 사용하여 모니터할 수 있습니다.

명령을 프롬트하면 사용자의 프로시듀어나 프로그램이 사용자가 입력한 명령 스트링을 수신하지 않습니다. 이렇게 하려면 QCMDCHK를 사용하여 프롬트한 다음, QCMDEXC를 사용하여 명령을 수행해야 합니다. 또한 QCAPCMD를 사용하여 프롬트하고 명령을 수행할 수 있습니다.

CL 명령의 선택 프롬팅

사용자는 명령 안에서 선택된 매개변수에 대해 프롬트를 요구할 수 있습니다. 이는 비교적 긴 명령을 사용하는 경우와 일부 매개변수에 대해서 프롬트를 원하지 않을 경우에 특히 유용합니다.

대화식 프롬팅에서 선택 프롬팅을 사용하거나 CL 프로시듀어나 프로그램 안에서 사용하도록 소스(SEU에서)로 입력할 수 있습니다. SEU로 선택 프롬팅에 대한 소스를 입력할 수 있으나 명령을 SEU로 입력하는 동안에는 선택 프롬팅을 사용할 수 없습니다.

다음 경우 선택 프롬팅을 사용할 수 있습니다.

- 프롬팅이 필요한 매개변수를 선택하기 위해
- 어느 매개변수를 보호할 것인지를 판별하기 위해
- 프롬트에서 매개변수를 생략하기 위해

다음은 선택 프롬팅에 적용되는 제한사항입니다.

- 다음의 경우에는 ?(의문 부호)가 명령 이름이나 레이블 앞에 선행되어야 합니다.
  - 선택 프롬트 옵션 중 하나 이상이 ?-(의문 부호, 빼기 부호)인 경우
  - CPF6805 메세지(컴파일이 성공적이더라도 명령상의 진단 문제점을 나타내는 메세지)가 표시되지 않도록 할 경우
- 위치에 따라 매개변수를 지정할 수 있으나 그 앞에 선택 프롬트 문자가 있어서는 안됩니다.
- 매개변수를 선택적으로 프롬트하기 위해서는 키워드 형식이어야 합니다.
- 선택 프롬트 문자와 키워드 사이에는 공백을 입력할 수 없습니다.
- 선택 프롬팅은 매개변수 레벨에만 적용시킬 수 있습니다. 즉, 값의 리스트에 특정 키워드 값을 지정할 수 없습니다.
- ?-는 프롬트 대체 프로그램에 사용될 수 없습니다.
- 필수 매개변수인 경우에는 ?? 선택 프롬트를 사용해야 합니다.

  명령이 프롬트되면 입력 슬롯이 강조표시되기 때문에 매개변수가 필수 매개변수임을 알 수 있습니다.

사용자 지정 값은 선택 프롬팅과 일반 프롬팅에서 모두 값 앞에 특수 기호(>)를 사용하여 표시합니다. 매개변수 프롬트의 사용자 지정 값 앞에 이 기호가 없으면, 명령 디폴트가 명령 처리 프로그램으로 전달됩니다.

PASSATR(*YES)을 PARM, ELEM 또는 QUAL 명령 정의문에 지정하고 CHGCMDDFT 명령을 사용하여 디폴트 값을 변경하면 디폴트 값이 아닌 사용자 지정 값(기호 >를 사용)으로 디폴트 값이 표시됩니다. 변경된 PARM, ELEM 또는 QUAL 명령 정의문의 디폴트 값이 다시 원래의 디폴트 값으로 변경되면 기호 >가 제거됩니다.

선택 프롬팅을 사용하는 중에 F5 키를 누르면 화면에 맨 처음 표시되었던 값들을 다시 표시할 수 있습니다.

선택 프롬팅에 의해 표시되는 매개변수의 값을 CL 변수를 사용하여 지정하면 프롬트의 값을 변경할 수 있으며, 변경된 값은 명령 수행 시 사용됩니다. 프로시듀어나 프로그램 안의 변수 값은 변경되지 않습니다. CL 프로시듀어에 다음이 포함되어 있다고 가정해 보겠습니다.

```
OVRDBF ?*FILE(FILEA) ??TOFILE(&FILENAME) ??MBR(MBR1)
```

이 경우 세 개의 매개변수(FILE, TOFILE 및 MBR)가 프롬트 화면에 표시됩니다. FILE 매개변수에 지정된 값은 사용자가 변경할 수 없으나, TOFILE과 MBR 매개변수에 지정된 값은 변경할 수 있습니다. CL 변수 &FILENAME의 값이 FILE1이고, 사용자가 이 값을 FILE2로 변경한다고 가정해 보겠습니다. 이 경우 명령이 실행될 때 FILE2의 값이 사용되지만 &FILENAME의 값은 프로시듀어에서 변경되지 않습니다. 다음 표는 여러 가지 선택 프롬팅 문자와 그 결과를 나열한 것입니다.

입력 내용	표시되는 값	보호 여부	지정된 것이 없을 경우 **CPP**로 전달되는 값	기호 > 표시 여부
??KEYWORD()	디폴트	아니오	디폴트	아니오
??KEYWORD(VALUE)	값	아니오	값	예
?*KEYWORD()	디폴트	예	디폴트	아니오
?*KEYWORD(VALUE)	값	예	값	예
?<KEYWORD()	디폴트	아니오	디폴트	아니오
?<KEYWORD(VALUE)	값	아니오	디폴트	아니오
?/KEYWORD()	디폴트	예	디폴트	아니오
?/KEYWORD(VALUE)	값	예	디폴트	아니오
?-KEYWORD()	없음	N/A	디폴트	N/A
?-KEYWORD(VALUE)	없음	N/A	값	N/A
?&KEYWORD()	디폴트	아니오	디폴트	아니오
?&KEYWORD(VALUE)	값	아니오	디폴트	아니오
?%KEYWORD()	디폴트	예	디폴트	아니오
?%KEYWORD(VALUE)	값	예	디폴트	아니오

입력 내용	**F5** 키를 누르거나 공백일 경우의 표시 값	설명
??KEYWORD()	디폴트	명령 디폴트를 가진 정상 키워드 프롬트
??KEYWORD(VALUE)	값	프로그램 지정 디폴트를 가진 정상 키워드 프롬트
?*KEYWORD()	디폴트	명령 디폴트만이 사용된 보호 프롬트 표시(정보로서)
?*KEYWORD(VALUE)	값	프로그램 지정 값만이 사용된 보호 프롬트 표시(정보로서). 예를 들면, 값이 정보로만 표시되고 변경되지는 않는 경우입니다.

입력 내용	F5 키를 누르거나 공백일 경우의 표시 값	설명
?<KEYWORD()	디폴트	명령 디폴트를 가진 정상 키워드 프롬트
?<KEYWORD(VALUE)	값	프로그램 지정 디폴트를 가진 정상 키워드 프롬트
?/KEYWORD()	디폴트	IBM용으로 예약
?/KEYWORD(VALUE)	값	IBM용으로 예약
?&KEYWORD()	디폴트	명령 디폴트를 가진 정상 키워드 프롬트
?&KEYWORD(VALUE)	값	프로그램 지정 디폴트를 가진 정상 키워드 프롬트
?%KEYWORD()	디폴트	명령 디폴트만이 사용된 보호 프롬트 표시(정보로서)
?%KEYWORD(VALUE)	값	프로그램 지정 값만이 사용된 보호 프롬트 표시(정보로서). 예를 들면, 값이 정보로만 표시되고 변경되지는 않는 경우입니다.

선택 프롬팅은 QCMDEXC 또는 QCMDCHK 프로그램에 사용될 수 있습니다. 다음은 호출 형식입니다.

```
CALL PGM(QCMDEXC or QCMDCHK) PARM(command command-length)
```

다음은 선택 프롬팅 문자에 대한 간단한 설명입니다.

선택 프롬팅 문자	설명
??	매개변수가 표시되고 입력이 가능합니다.
?*	매개변수가 표시되나 입력이 가능하지 않습니다. 모든 사용자 지정 값이 명령 처리 프로그램으로 전달됩니다.
?<	매개변수가 표시되고 입력이 가능하지만, 매개변수에 표시된 값이 변경되지 않으면 명령 디폴트가 CPP로 송신됩니다.
?/	IBM용으로 예약
?-	매개변수가 표시되지 않습니다. 지정된 값(또는 디폴트)이 CPP로 전달됩니다. 프롬트 대체 프로그램에는 허용되지 않습니다.
?&	F9(모든 매개변수) 키를 누르면 매개변수가 표시됩니다. 일단 표시되면 입력이 가능합니다. 매개변수에 표시된 값이 변경되지 않으면 명령 디폴트가 CPP로 송신됩니다.
?%	F9(모든 매개변수) 키를 누르면 매개변수가 표시됩니다. 일단 표시되면 입력이 가능하지 않습니다. 명령 디폴트가 CPP로 송신됩니다.

QCMDEXC 또는 QCMDCHK에 대해 더 자세히 알려면 191 페이지의 『QCMDEXC 프로그램 사용』 부분과 195 페이지의 『QCMDCHK 프로그램 사용』 부분을 참조하십시오.

CL 프로시듀어나 프로그램에서 프롬팅되는 QCMDEXC

QCMDEXC 프로그램은 프롬터를 호출하는 데 사용될 수 있습니다. CL 프로시듀어와 프로그램에서 프롬팅과 함께 QCMDEXC를 사용하면 명령어 자체를 제외한 명령상의 모든 값을 변경할 수 있습니다. 이것은 소스(앞 섹션 참조)에 지정되지 않은 값만 입력할 수 있는 프롬터의 직접 사용 시보다 훨씬 융통성이 있습니다. 다음의 명령을 사용하여 프롬터를 직접 호출한 예를 보겠습니다.

```
?OVRDBF FILE(FILEX)
```

이 경우 FILE을 제외한 모든 매개변수에 값을 지정할 수 있습니다. 그러나 QCMDEXC 프로그램을 사용하여 프로그램을 처리하는 동안 다음과 같이 명령이 호출되는 경우

```
CALL QCMDEXC PARM('?OVRDBF FILE(FILEX)' 19)
```

FILE을 포함한 모든 매개변수에 값을 지정할 수 있습니다. 이 예에서는 FILEX가 디폴트입니다.

이 장의 앞에서 설명한 선택 프롬팅을 사용하면 수정이 가능한 지정 값을 가진 프롬팅도 할 수 있습니다. 그러나 각 키워드는 명시적으로 선택해야 합니다. 프롬트는 다음과 같은 명령으로 직접 호출될 수 있습니다.

```
OVRDBF ??FILE(FILEX) ??TOFILE(*N) ??MBR(*N)
```

프로그래머 메뉴 사용

QPGMMENU 프로그램을 호출하거나 STRPGMMNU(프로그래머 메뉴 시작) 명령을 사용하여 프로그래머 메뉴를 직접 호출할 수도 있습니다. 이 명령을 사용하면 프로그래머 메뉴와 함께 사용할 디폴트를 미리 지정할 수 있습니다. 또한 STRPGMMNU 명령은 프로그래머 메뉴의 사용을 조정하는 데 사용할 수 있는 다른 옵션도 지원합니다.

STRPGMMNU 명령 및 그 매개변수의 설명은 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

STRPGMMNU(프로그래머 메뉴 시작) 명령의 사용

다음 경우에는 프로그래머 메뉴 시작 명령을 사용할 수 있습니다.

- QPGMMENU 호출과 같은 기능을 수행하는 경우
- 표준 입력 필드를 채우려는 경우

  다음 네 개의 명령 매개변수를 사용하면 메뉴의 맨 아래에 있는 표준 입력 필드를 채울 수 있습니다. 이 매개변수는 다음과 같습니다.

  – 소스 파일
  – 소스 라이브러리
  – 오브젝트 라이브러리
  – 작업 설명

이 명령은 메뉴의 초기값을 제어하는 여러 매개변수와 함께 사용할 수 있습니다. 사용자는 이것을 사인 온이나 특정한 사용자 작성 기능을 호출하는 상황을 위한 초기 프로그램의 일부로 설계할 수 있습니다. 다음은 각각 다른 초기 값을 필요로 하는 각각의 어플리케이션 영역에 대해 개별적인 기능을 가진 프로그램의 예입니다.

```
        PGM
      CHGLIBL   LIBL(PGMR1 QGPL QTEMP)
LOOP:
      STRPGMMNU SRCLIB(PGMR1) OBJLIB(PGMR1) JOBD(PGMR1)
      MONMSG    MSGID(CPF2320) EXEC(GOTO END) /* F3 or F12 to leave menu */
         GOTO LOOP
END:       ENDPGM
```

- 프로그래머 메뉴 옵션을 제어하기 위해

  다른 매개변수는 사용자가 메뉴와 그 기능을 제어할 수 있도록 도와줍니다. 예를 들면, 메뉴에 표시된 값을 변경하지 못하도록 ALWUSRCHG(*NO)를 지정할 수 있습니다. 메뉴 사용자가 STRPGMMNU 명령을 호출하여 개별적인 호출에서 그 값을 변경시킬 수 있기 때문에 이 매개변수는 보안 피처로 간주될 수 없습니다. (또한 F10 키를 사용하여 명령 입력 화면을 호출하여 기능을 시작할 수도 있습니다.) STRPGMMNU 명령에 의해 메뉴가 표시되면 사용자가(권한 부여로) QPGMMENU 프로그램을 직접 호출하지 못하게 할 수 있으나, STRPGMMNU 명령의 다른 호출을 요구하지 못하게 할 수 없습니다.

- 메뉴 작성 옵션을 채택하기 위해

  EXITPGM과 DLTOPT 매개변수를 사용하여 메뉴 작성 옵션(옵션 3)에 대한 사용자 지원을 제공할 수 있습니다. 사용자가 옵션 3을 요구하면 시스템이 사용자 프로그램을 호출할 수도 있습니다. IBM에서는 사용자 프로그램으로 전달되는 매개변수와 매개변수 리스트를 설명하는 온라인 정보를 제공합니다. iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오. 다음은 EXITPGM 매개변수 사용에 관한 설명입니다.

EXITPGM 매개변수

EXITPGM 매개변수는 다음과 같은 목적으로 사용될 수 있습니다.

- 옵션 3에 의해 제출된 작성 명령에서 사용되는 디폴트를 변경하기 위해

  예를 들어, F4(프롬트) 키가 사용되지 않으면 사용자 자신의 디폴트 요구사항을 지정하기 위해 EXITPGM 매개변수가 여러 작성 명령을 변경시킬 수 있습니다. F4 키가 사용되면 EXITPGM 매개변수는 프로그래머가 입력한 명령을 제출할 수 있습니다(매개변수의 변경 없이).

- 프로그래머가 F4 키를 사용하는 것과 관계 없이 매개변수를 변경하기 위해

  이것은 &RQSDTA512 매개변수의 값(나감 프로그램으로 전달된)이 이미 사용되었는지 보기 위해 이 값의 검색과 필요한 값의 대체를 필요로 합니다.

- SBMJOB 명령에서 기타 매개변수를 변경하기 위해

예를 들어, SBMJOB 명령의 사용자 매개변수를 *CURRENT의 값 대신 작업 설명의 값을 지정하도록 변경시킬 수 있습니다. RTVJOBA 명령을 사용하여 속성을 특정한 값으로 입력하여 여러 작업 속성 값을 검색하는 것도 가능합니다.

- 로컬 프로그래밍 규칙을 강제하기 위해

  예를 들어, 모든 실제 파일의 이름이 P로 끝나면서 길이에 7자를 사용하는 명명 기준의 경우 이 기준에 맞지 않으면 나감 프로그램이 CRTPF 명령을 시도하지 않습니다.

명령 분석기 나감점

나감 프로그램 등록 기능에서는 시스템에 두 개의 나감점을 제공합니다.

- QIBM_QCA_CHG_COMMAND 나감점은 특정 명령에 대해 단 하나의 나감점만 등록할 수 있습니다. 명령 분석기가 프롬터로 제어를 전달하기 전에 이 나감점에 대해 지정된 프로그램을 호출합니다.
- QIBM_QCA_RTV_COMMAND 나감점을 위한 각 명령에 대해 최대 10개까지 나감 프로그램을 등록할 수 있습니다. 명령 분석기는 유효성 검사 프로그램(VCP)을 실행한 후 그리고 명령에 대해 명령 처리 프로그램(CPP)을 실행하기 전에 나감 프로그램을 호출합니다.

나감점에 관한 모든 설명은 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

DBCS 자료를 위한 어플리케이션 프로그래밍

2바이트 자료를 처리하기 위해 어플리케이션 프로그램을 설계하거나 영숫자 어플리케이션 프로그램을 2바이트 프로그램으로 변환시킬 때는 특별히 고려해야 할 사항이 있습니다.

DBCS 어플리케이션 프로그램 설계

다음 사항을 추가로 고려하여 영숫자 자료를 처리하기 위한 어플리케이션 프로그램을 설계하는 것과 같은 방식으로 2바이트 자료를 처리하기 위한 어플리케이션 프로그램을 설계하십시오.

- 데이터베이스 파일에서 사용되는 2바이트 자료(있는 경우)를 알아보십시오.
- 2바이트 자료가 사용될 수 있도록 표시장치와 프린터 형식을 설계하십시오.
- 필요한 경우 대화식 어플리케이션에 대해 자료를 입력하는 방식으로 2바이트 변환을 제공하십시오. 화면 파일에 DBCS 변환을 지정하려면 2바이트 변환용 DDS 키워드(IGCCNV)를 사용하십시오.
- 프로그램에 의해 표시될 2바이트 오류 메세지를 기록하십시오.

- 시스템이 모든 2바이트 자료를 인쇄하고 표시할 수 있도록 확장 문자 처리를 지정하십시오.
- 2바이트 문자(있는 경우)가 정의되어야 하는지를 판별하십시오. ADTS/400: Character Generator Utility  는 DBCS 지원 국가용 2바이트 문자의 정의 방법에 대한 설명을 제공합니다.

DBCS 자료를 처리하기 위한 영숫자 프로그램의 변환

영숫자 어플리케이션 프로그램이 외부 서술 화면 파일을 사용할 경우 파일만 변경하여 어플리케이션 프로그램을 2바이트 어플리케이션 프로그램으로 변경할 수 있습니다. 어플리케이션 프로그램을 변환하려면 다음과 같이 하십시오.

1. 변경하려는 영숫자 파일 소스문의 중복 사본을 작성하십시오.
2. 영숫자 상수와 리터럴을 2바이트 상수와 리터럴로 변경하십시오.
3. 파일 안의 필드를 DBCS 자료를 입력하는 다음 자료 유형 중 하나로 변경하십시오.
   - DBCS 개방(O) 자료 유형
   - DBCS 전용(J) 자료 유형
   - DBCS 전용/개방(E) 자료

   필드 길이는 변경할 필요가 없습니다.
4. 변환된 화면 파일을 별도의 라이브러리에 저장하십시오. 파일에 영숫자 버전과 같은 이름을 지정하십시오.
5. 작업에서 변환된 파일을 사용하려면 CHGLIBL(라이브러리 리스트 변경) 명령을 사용하여 파일이 사용될 작업의 라이브러리 리스트를 변경하십시오. 그러면 파일의 영숫자 버전이 저장된 라이브러리보다 2바이트 화면 파일이 저장된 라이브러리가 먼저 검사됩니다.

CL 프로그램에서의 DBCS 자료 사용

다음 프로그램은 CL 프로그램 안에서의 기타 키보드 시프트 사용법을 보여줍니다. 2바이트 자료가 이 프로그램에서는 텍스트 값으로만 사용되는 방법에 주의하십시오. 명령 자체는 영숫자 문자입니다.

실행 시 이 프로그램은 여러 가지 키보드 시프트가 DDS 화면 파일에 어떻게 사용되는지를 보여줍니다.

```
PGM

      DCLF    IGCTEST

START: CHGVAR &OUTPUTA  'ABCDEFGHIJ'

       CHGVAR &OUTPUTJ  'ABCD'
       CHGVAR &BOTHJ    'ABCD'
       CHGVAR &OUTPUTE  'EFGH'
       CHGVAR &OUTPUTO  'A B C  D  F'


LOOP:  SNDRCVF

      IF &INO1 RETURN
       CHGVAR &OUTPUTA  &INPUTA
       CHGVAR &OUTPUTJ  &INPUTJ
       CHGVAR &OUTPUTE  &INPUTE
       CHGVAR &BOTHE    &INPUTE
       CHGVAR &OUTPUTO  INPUTO


    GOTO LOOP

ENDPGM
```

RV3W197-0

CL 프로그램 샘플

다음 샘플 프로그램은 CL 프로그램의 융통성, 간편성, 다양성을 보여주는 것입니다. 다음 프로그램은 기능별 및 사용자별로 설명됩니다.

주: V4R3 이상의 릴리스에서 ILE CL 컴파일러가 생성한 코드는 스레드세이프입니다. 그러나 많은 명령들이 스레드세이프가 아닙니다. 따라서 CL 프로시듀어가 사용하는 모든 명령이 스레드세이프가 아닌 한 CL 프로시듀어를 스레드세이프로 간주해서는 안됩니다. 명령이 스레드세이프인지 판별하려면 DSPCMD(명령 표시) 명령을 사용하십시오. 스레드에 대한 추가 정보를 보려면 iSeries Information Center에 액세스하여 **프로그래밍** 범주에 있는 주제를 여십시오.

설정용 초기 프로그램(프로그래머)

```
         PGM
CHGLIBL LIBL(TESTLIB QGPL QTEMP)
CHGJOB OUTQ(WSPRTR)
TFRCTL QPGMMENU
    ENDPGM
```

테스트 라이브러리가 라이브러리 리스트의 맨 처음에 놓이며, 출력 대기행렬은 편리한 프린터에 선택되고, 프로그래머 메뉴가 표시됩니다.

테스트 라이브러리에서 제품 라이브러리로 오브젝트 이동(프로그래머)

```
PGM PARM(&OBJ &OBJTYPE &OPER)
DCL &OBJ *CHAR LEN(10)
DCL &OBJTYPE *CHAR LEN(7)
DCL &OPER *CHAR LEN(1)  /* R=Replace M=Move */
IF ((&OPER *NE 'M') *AND (&OPER *NE 'R')) THEN(DO)
        SNDPGMMSG MSG('Operation code must be "R" or "M" ')
          RETURN
 ENDDO
IF ((&OBJTYPE *NE *PGM) *AND (&OBJTYPE *NE *FILE) *AND (&OBJTYPE +
    *NE *DTAARA)) THEN(DO)
        SNDPGMMSG MSG('Object' *BCAT &OBJ *BCAT ' must be *PGM, +
                  *FILE, or *DTAARA')
          RETURN
 ENDDO
CHKOBJ BLDLIB/&OBJ OBJTYPE(&OBJTYPE)
        MONMSG MSGID(CPF9801) EXEC(DO)
        SNDPGMMSG MSG('Object or object type does not exist +
                  in BLDLIB')
          RETURN
 ENDDO
IF (&OPER *EQ 'M') THEN(DO)
        MOVOBJ BLDLIB/&OBJ OBJTYPE(&OBJTYPE) TOLIB(PRODLIB)
        MONMSG MSGID(CPF3208) EXEC(DO)
                        SNDPGMMSG MSG('Object' *BCAT &OBJ *BCAT ' +
                                  already exists in PRODLIB')
          RETURN
 ENDDO
        CHKOBJ PRODLIB/&OBJ OBJTYPE(&OBJTYPE)
        MONMSG MSGID(CPF9801) EXEC(DO)
        SNDPGMMSG MSG('Object or object type does not +
                  exist in PRODLIB')
          RETURN
 ENDDO
 ENDDO
          RETURN
    ENDPGM
```

오브젝트명, 오브젝트 유형, OP 코드는 또 다른 프로그램이나 프로시듀어로부터 전달됩니다. OP 코드와 오브젝트 유형이 맞는지 그리고 그 오브젝트가 테스트 라이브러리에 존재하는지를 확인하기 위해 검사가 수행됩니다. 오브젝트가 제품 라이브러리 아직 없으면 이동합니다. 그러면 이동을 확정 받습니다. 오브젝트에는 추가로 권한을 부여하거나 추가 예외 및 추가 오브젝트 유형을 처리하기 위해 더 많은 명령을 추가시킬 수 있습니다.

어플리케이션에서 특정 오브젝트 저장(시스템 오퍼레이터)

예

```
         PGM
SAVOBJ OBJ(FILE1 FILE2) LIB(LIBA) OBJTYPE(*FILE) DEV(TAP01) +
  CLEAR(*YES)
```

```
SAVOBJ OBJ(DTAARA1) LIB(LIBA) OBJTYPE(*DTAARA) DEV(TAP01)
SNDPGMMSG MSG('Save of daily backup of LIBA completed') +
                    MSGTYPE(*COMP)
    ENDPGM
```

이 프로그램은 규칙적으로 반복되는 프로시듀어에 대해 일관성 있는 명령 입력을 보장합니다.

또한 SAVOBJ(오브젝트 저장) 명령도 추가할 수 있습니다. 그러나 오퍼레이터가 각 어플리케이션의 정기적인 백업마다 알맞는 디스켓이나 테이프를 선택해야 합니다. 이는 각 저장 조작에 설정된 디스켓이나 테이프별로 고유한 이름을 할당하여 제어할 수 있습니다. 예를 들면, 급여 파일을 4주 동안 매주마다 별도로 분리시켜 저장하려면 각 디스켓 또는 테이프를 다르게 명명하여 디스켓 또는 테이프의 이름이 그 주에 맞는 이름인지를 비교하는 프로그램을 작성하면 됩니다.

비정상 종료의 회복(시스템 오퍼레이터)

```
         PGM
DCL &SWITCH *CHAR LEN(1)
RTVSYSVAL SYSVAL(QABNORMSW) RTNVAR(&SWITCH)
IF (&SWITCH *EQ '1') THEN(DO)  /*CALL RECOVERY PROGRAMS*/
        SNDPGMMSG MSG('Recovery programs in process. +
          Do not start subsystems until notified') +
          MSGTYPE(*INFO) TOMSGQ(QSYSOPR)
CALL PGMA
CALL PGMB
        SNDPGMMSG MSG('Recovery programs complete. +
          Startup subsystems') +
          MSGTYPE(*INFO) TOMSGQ(QSYSOPR)
          RETURN
 ENDDO
    ENDPGM
```

작업 제출(시스템 오퍼레이터)

```
PGM  /*DAILYAC*/
SBMJOB JOB(DAILYACCRC) JOBD(ACCRC2) +
            CMD(CALL ACCRC305 PARM(DAILY))
SNDPGMMSG   MSG('Daily Accounts Receivable job DAILYACCRC +
            submitted to batch') MSGTYPE(*COMP)
    ENDPGM
```

시스템 오퍼레이터는 모든 매개변수를 입력하는 대신 DAILYAC를 호출하는 것만으로 작업을 제출할 수 있습니다.

장치 화면으로부터의 입력 대기 중 시간종료

```
            DCLF FILE(QGPL/MENU)
DOWHILE '1' /* DO FOREVER */
  SNDRCVF DEV(*FILE) RCDFMT(MENUFMT) WAIT(*NO)
  WAIT  MONMSG MSGID(CPF0889) EXEC(SIGNOFF)
```

```
            CHGVAR VAR(&IN99) VALUE('0')
  IF COND(&IN01) THEN(ITERATE)
  SELECT
    WHEN (&OPTION *EQ '1') (CALL ORDENT) /* OPTION 1-ORDER ENTRY      */
    WHEN (&OPTION *EQ '2') (CALL ORDDSP) /* OPTION 2-ORDER DISPLAY    */
    WHEN (&OPTION *EQ '3') (CALL ORDCHG) /* OPTION 3-ORDER CHANGE     */
    WHEN (&OPTION *EQ '4') (CALL ORDPRT) /* OPTION 4-ORDER PRINT      */
    WHEN (&OPTION *EQ '9') (SIGNOFF)     /* OPTION 9-SIGNOFF          */
    OTHERWISE DO                         /* OPTION SELECTED NOT VALID */
            CHGVAR VAR(&IN99) VALUE('1')
 ENDDO
  ENDSELECT
 ENDDO
     ENDPGM
```

이 프로그램은 사용자가 옵션을 입력할 때까지 지정 시간 동안 대기할 CL 프로그램을 화면 파일을 사용하여 작성하는 방법을 보여줍니다. 옵션을 입력하지 않으면 사인 오프됩니다.

화면 파일은 다음의 명령으로 작성되었습니다.

```
CRTDSPF FILE(MENU) SRCFILE(QGPL/QDDSSRC) SRCMBR(MENU)  +
    DEV(*REQUESTER) WAITRCD(60)
```

화면 파일은 *REQUESTER 장치를 사용할 것입니다. WAIT 명령이 발행되면 WAITRCD 키워드에 지정된 시간(60초)만큼 대기합니다. 다음은 화면 파일에 대한 DDS 입니다.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ... ... 8

 0100       A                                      PRINT CA01(01)
 0200       A          R MENUFMT                   BLINK
 0300       A                                      TEXT('Order Entry Menu')
 0400       A                                    1 31'Order Entry Menu'
 0500       A                                    2  2'Select one of the following:    '
 0600       A                                    3  4'1.  Enter Order'
 0700       A                                    4  4'2.  Display Order'
 0800       A                                    5  4'3.  Change Order'
 0900       A                                    6  4'4.  Print Order'
 1000       A                                    7  4'9.  Sign Off'
 1100       A                                   23  2'Option:'
 1200       A            OPTION        1   I 23 10
 1300       A  99                                  ERRMSG('Invalid option selected.')

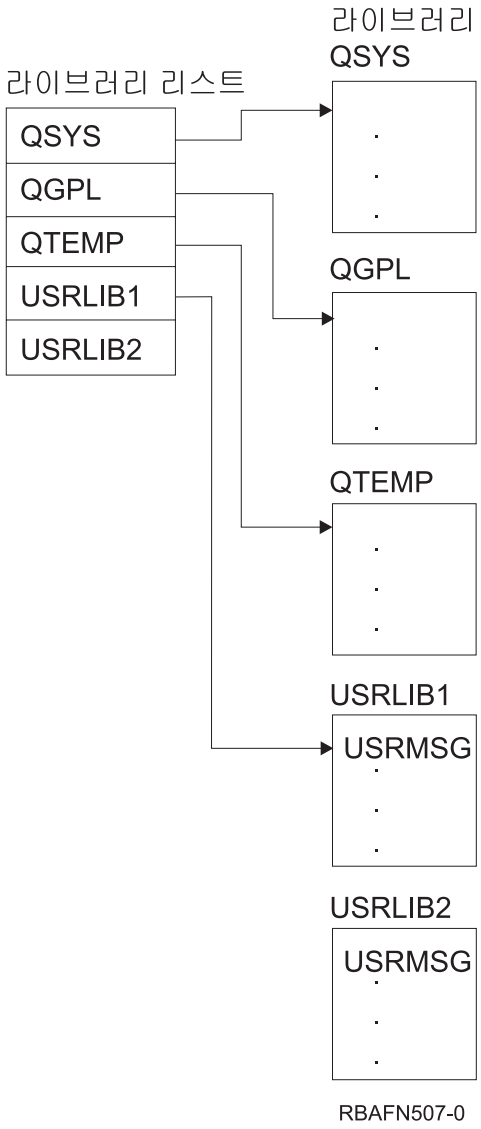
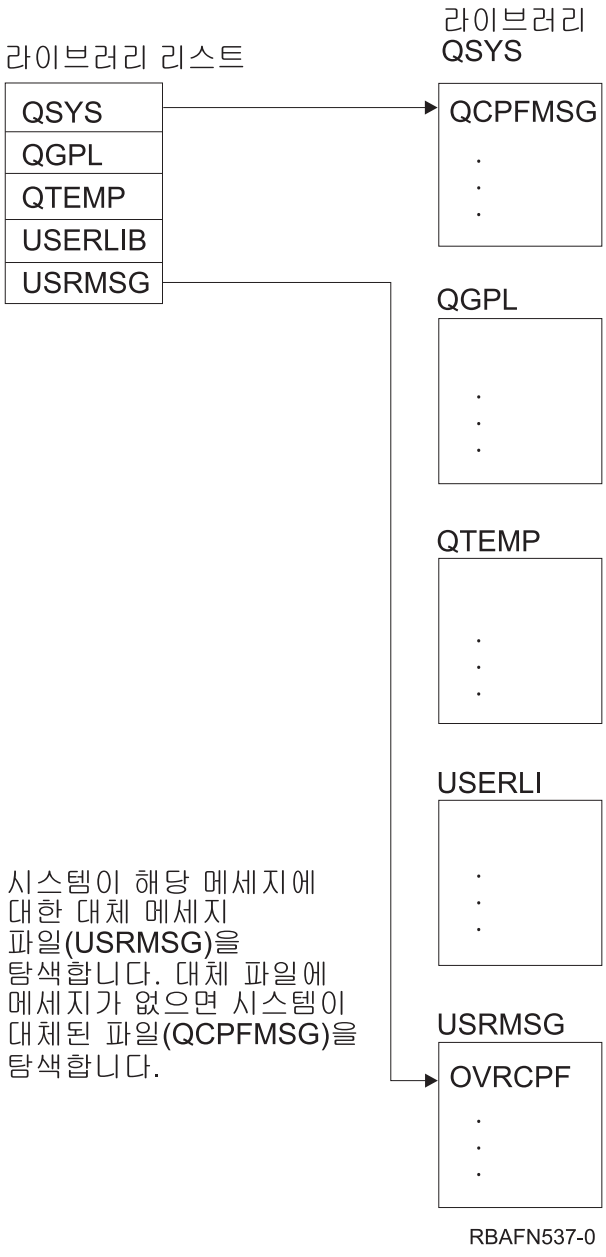
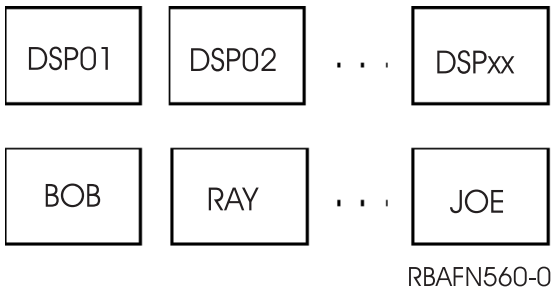
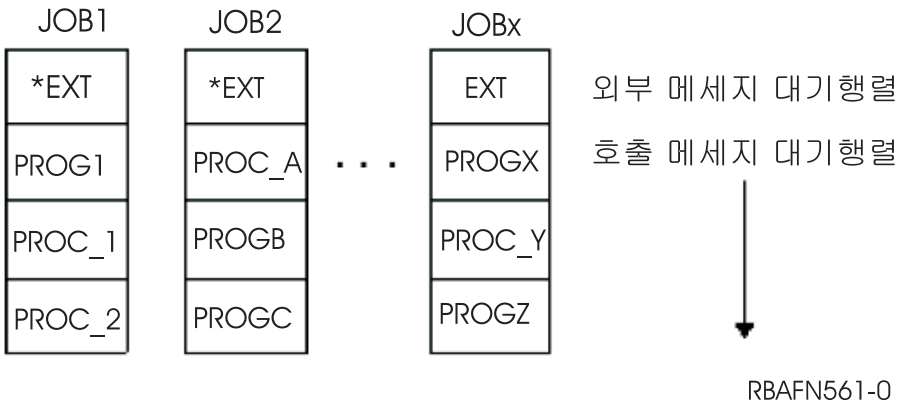
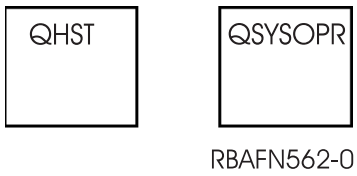
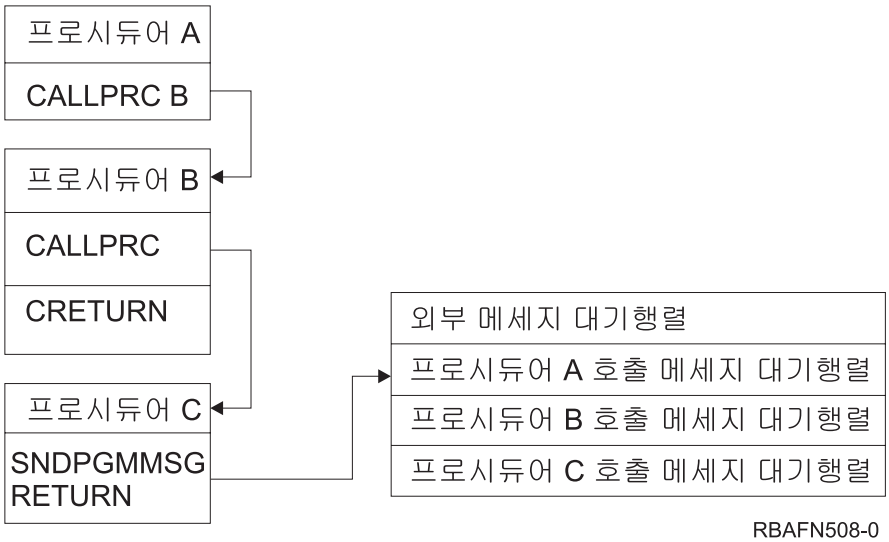
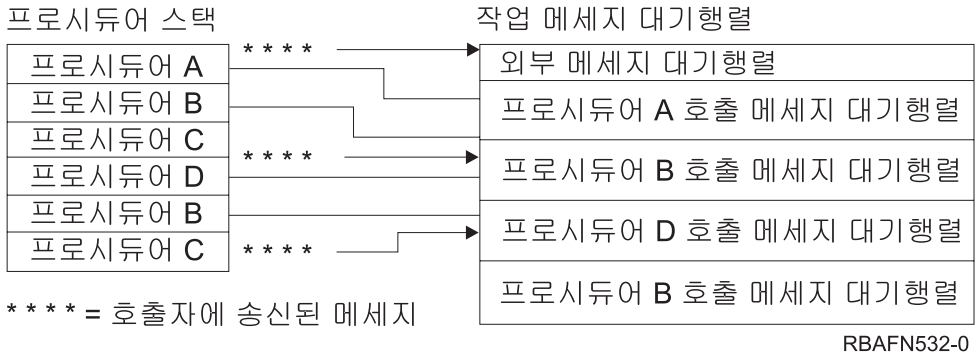
                          * * * * * *  E N D  O F  S O U R C E  * * * * *
```

프로그램은 SNDRCVF WAIT(*NO)를 수행하여 메뉴를 표시하고 사용자로부터의 옵션 입력을 요구합니다. 그런 후 WAIT 명령을 발행하여 사용자로부터 옵션을 받아들입니다. 사용자가 1-4 중 하나를 입력하면 해당 프로그램이 호출됩니다. 사용자가 9를 입력하면 SIGNOFF 명령이 발행됩니다. 사용자가 유효하지 않은 옵션을 입력하면 메뉴에 ″유효하지 않은 옵션이 선택됨″이라는 메세지가 표시됩니다. 그러고 나면 사용자가 다른 유효한 옵션을 입력할 수 있습니다. 사용자가 60초 안에 응답하지 않으면 CPF0889 메세지가 프로그램에 발행되고 MONMSG 명령이 SIGNOFF 명령을 발행합니다.

INVITE DDS 키워드가 들어 있는 레코드 형식을 사용하는 SNDF 명령이 SNDRCVF WAIT(*NO) 대신 사용될 수 있습니다. 기능은 같을 것입니다.

날짜 산술 수행

```
/* Calculate new date from current system date.  Pass negative     */
/* number to subtract, positive number to add                      */
/*                                                                 */
/* The first parameter is a character 8 date in YYYYMMDD format    */
/* or the special value *CURRENT                                   */
/*                                                                 */
/* The second parameter is a decimal value for the number of days  */
/* to adjust the first parameter by                                */
/*                                                                 */
/* Test cases:  CALL CALCDATE (*CURRENT -5)                        */
/*              CALL CALCDATE (*CURRENT  5)                        */
/*              CALL CALCDATE ('20030225' -90)                     */
/*              CALL CALCDATE ('30020228' 90)                      */
/*                                                                 */
/* There is no error handling in this sample, so make sure the     */
/* input dates are valid (that is, no 20031325).  The valid date   */
/* date range is Oct 14 1582 to Dec 31 9999                        */
/*                                                                 */
             PGM        PARM(&curdate &DAYSTOCHG)
             DCL        VAR(&CURDATE) TYPE(*CHAR) LEN(8)
             DCL        VAR(&DAYSTOCHG) TYPE(*DEC) LEN(15 5)
             DCL        VAR(&DATETIME) TYPE(*CHAR) LEN(17)
             DCL        VAR(&DATE) TYPE(*CHAR) LEN(8)
             DCL        VAR(&LILDATEINT) TYPE(*CHAR) LEN(4)
             DCL        VAR(&LILDATEDEC) TYPE(*DEC) LEN(10 0)
             DCL        VAR(&ERRCOD) TYPE(*CHAR) LEN(4) +
                          VALUE(X'00000000')
             DCL        VAR(&MSG) TYPE(*CHAR) LEN(50)
             IF         COND(&CURDATE = '*CURRENT') THEN(DO)
             CALL       PGM(QWCCVTDT) PARM('*CURRENT' ' ' '*YYMD' +
                          &DATETIME &ERRCOD) /* Get current system +
                          date and time in YYYYMMDD */
             CHGVAR     VAR(&DATE) VALUE(%SST(&DATETIME 1 8)) /* Get +
                          just the date portion */
 ENDDO
             ELSE       CMD(CHGVAR VAR(&DATE) VALUE(&CURDATE)) /* +
                          Use the date provided */
             CALLPRC    PRC(CEEDAYS) PARM(&DATE 'YYYYMMDD' +
                          &LILDATEINT *OMIT) /* Get Lilian date for +
                          current date */
             CHGVAR     VAR(&LILDATEDEC) VALUE(%BIN(&LILDATEINT)) /* +
                          Get Lilian date in decimal format */
             CHGVAR     VAR(&LILDATEDEC) VALUE(&LILDATEDEC + +
                          &DAYSTOCHG) /* Adjust specified number +
                          of days */
             CHGVAR     VAR(%BIN(&LILDATEINT)) VALUE(&LILDATEDEC) /* +
                          Get Lilian date in integer format */
             CALLPRC    PRC(CEEDATE) PARM(&LILDATEINT 'YYYYMMDD' +
                          &DATE *OMIT) /* Return calculated date in +
```

```
                                YYYYMMDD format */
                    CHGVAR     VAR(&MSG) VALUE('The new date is ' *CAT &DATE)
                    SNDPGMMSG  MSG(&MSG) TOPGMQ(*EXT)
               ENDPGM
```

이 프로그램은 현재 시스템 날짜에 주어진 일 수를 더하거나 빼는 CL 프로그램 작성 방법을 보여줍니다.

프로그램 속성 검색

DAPPGM(프로그램 표시) 명령을 사용하면 프로그램의 속성을 표시할 수 있습니다. 일부 속성(프로그램 유형, 소스 멤버, 텍스트, 작성 날짜 등)들을 CL 변수로 검색하려는 경우에는 DSPOBJD(오브젝트 설명 표시) 명령을 사용하여 출력 파일을 만들 수 있습니다. 그러면 시스템이 DCLF(파일 선언) 및 RCVF(파일 수신) 명령을 사용하여 CL 프로시듀어나 프로그램을 읽을 수 있습니다. DSPPGM 명령의 일부 다른 속성(USRPRF와 같은)에 액세스하려는 경우에는 프로그램 정보 검색 API(QCLRPGMI)를 사용할 수 있습니다.

테이프나 광 매체로부터 어플리케이션 로드 및 수행

LODRUN(매체 프로그램 로드 및 수행) 명령을 사용하면 다른 사용자나 소프트웨어 밴더가 작성한 어플리케이션을 다른 사용자가 제공한 테이프나 광 매체로부터 로드하여 수행할 수 있습니다.

LODRUN 명령이 수행되면 다음 상황이 발생합니다.

- 매체가 반드시 QINSTAPP라고 명명된 사용자 작성 프로그램에 대해 탐색됩니다. 테이프가 사용된 경우에는 먼저 다시 감아야 합니다.
- QINSTAPP 프로그램이 사용자 시스템상의 QTEMP 라이브러리 안에 이미 존재하면 이 프로그램이 삭제됩니다.
- RSTOBJ 명령을 사용하여 QINSTAPP 프로그램이 QTEMP 라이브러리로 복원됩니다.
- 시스템 제어가 QINSTAPP 프로그램으로 전달됩니다. 예를 들면, 기타 어플리케이션을 사용자의 시스템으로 복원하여 그와 같은 어플리케이션을 수행하기 위해 QINSTAPP 프로그램이 사용될 수 있습니다.

어플리케이션 출력기의 역할

QINSTAPP 프로그램을 제공하는 사용자는 그 프로그램을 작성하고 지원할 책임이 있습니다. IBM*에서는 QINSTAPP 프로그램을 제공하지 않습니다. 프로그램은 여러 가지 서로 다른 타스크를 달성하도록 설계될 수 있습니다. 예를 들어, 프로그램에서 다음과 같은 일을 할 수 있습니다.

- 다른 프로그램이나 어플리케이션을 복원한 후 수행합니다.
- 라이브러리를 복원합니다.

- 다른 프로그램이나 어플리케이션을 삭제합니다.
- 특정 환경을 작성합니다.
- 기존 어플리케이션에서 문제점을 정정합니다.

그림 3에서는 QINSTAPP 프로그램의 예를 보여줍니다. 프로그램은 프로그램 출력기에 의해 테이프나 광 매체에 저장되었다가 LODRUN 명령을 사용하여 시스템에 로드됩니다. LODRUN 명령은 시스템의 제어를 프로그램으로 전달해서 프로그램에 작성된 타스크를 수행합니다.

```
PGM         PARM(&DEV)  /* "Device" is only Parm allowed           */
DCL         VAR(&DEV)   TYPE(*CHAR) LEN(10)
DCL         VAR(&MODEL) TYPE(*CHAR) LEN(4)

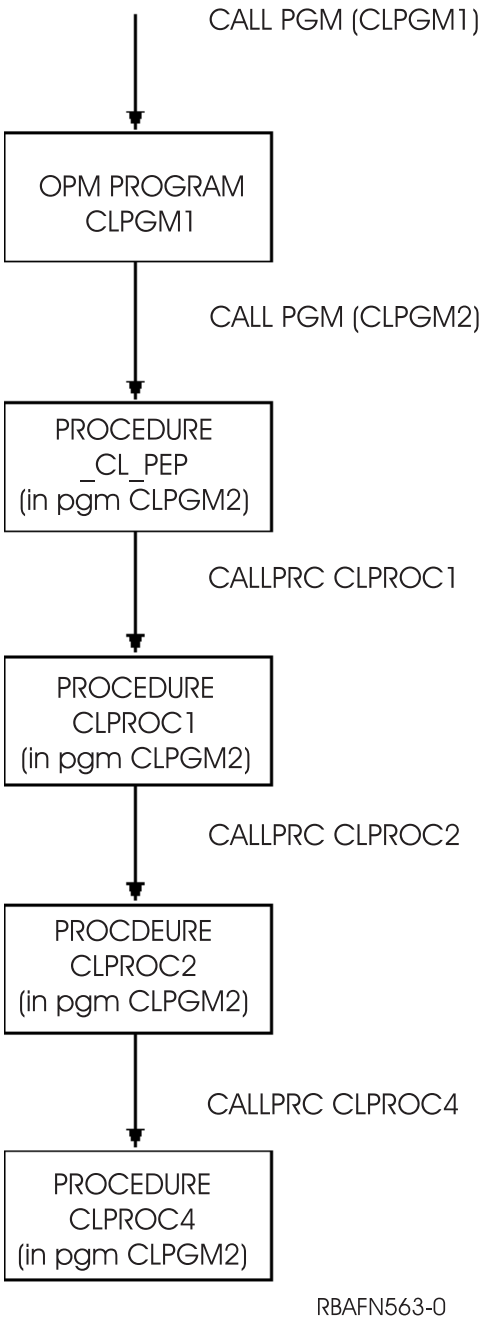
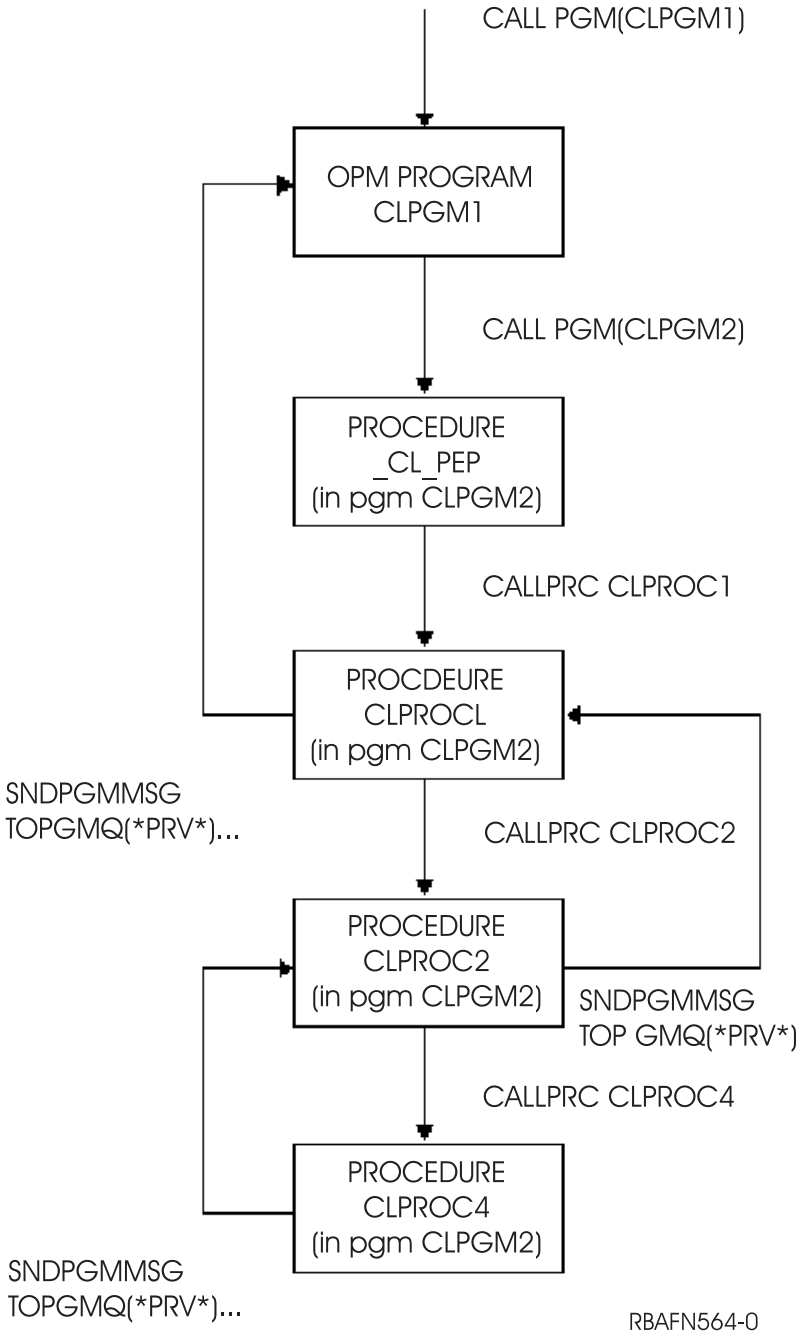
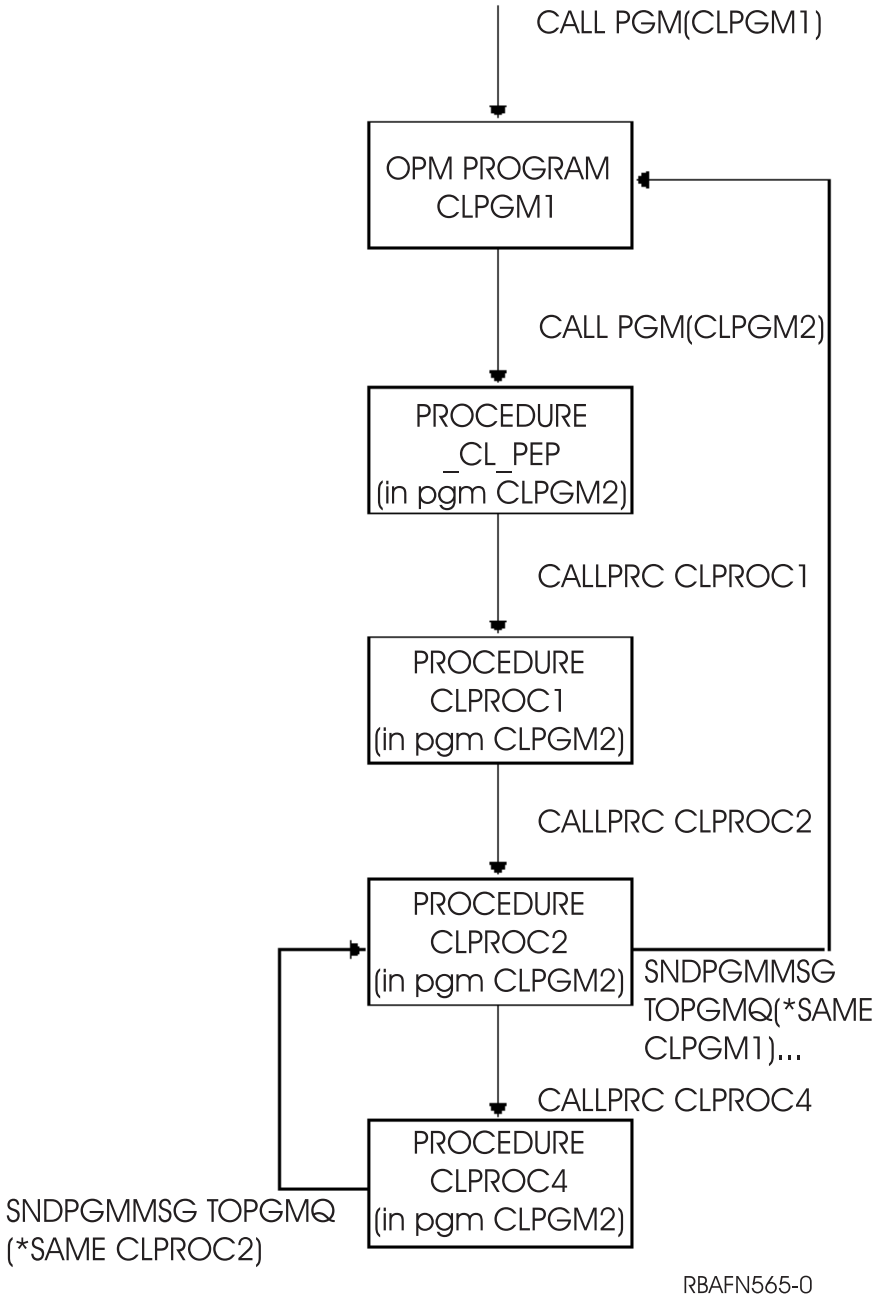
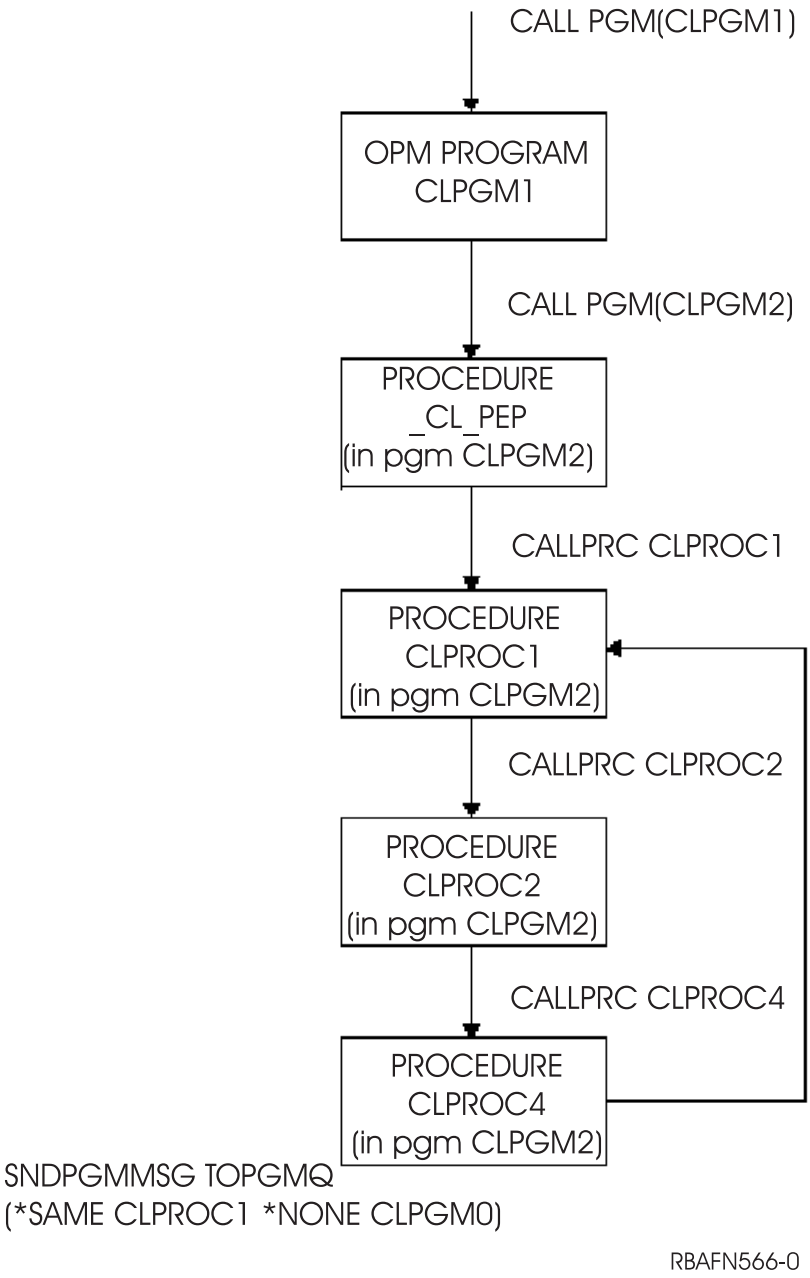
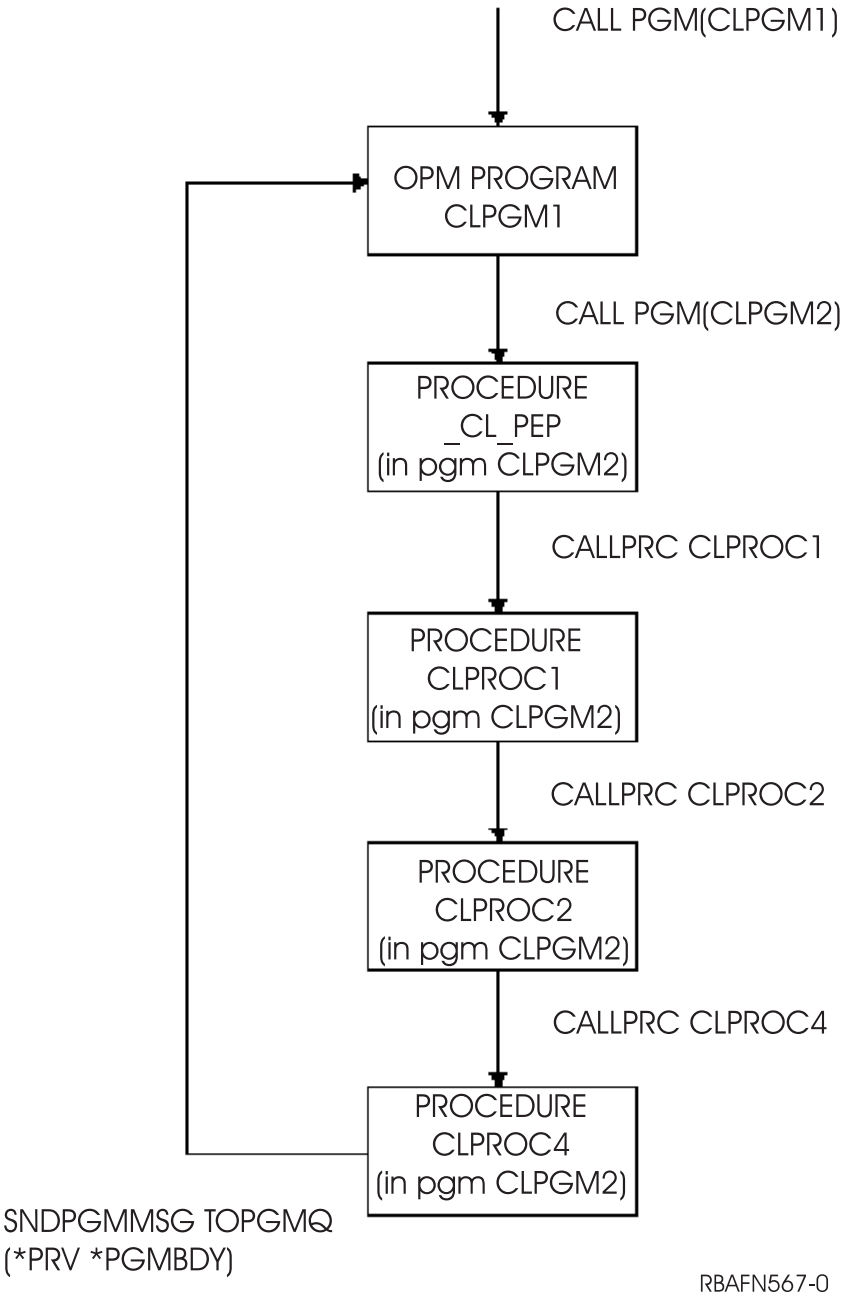
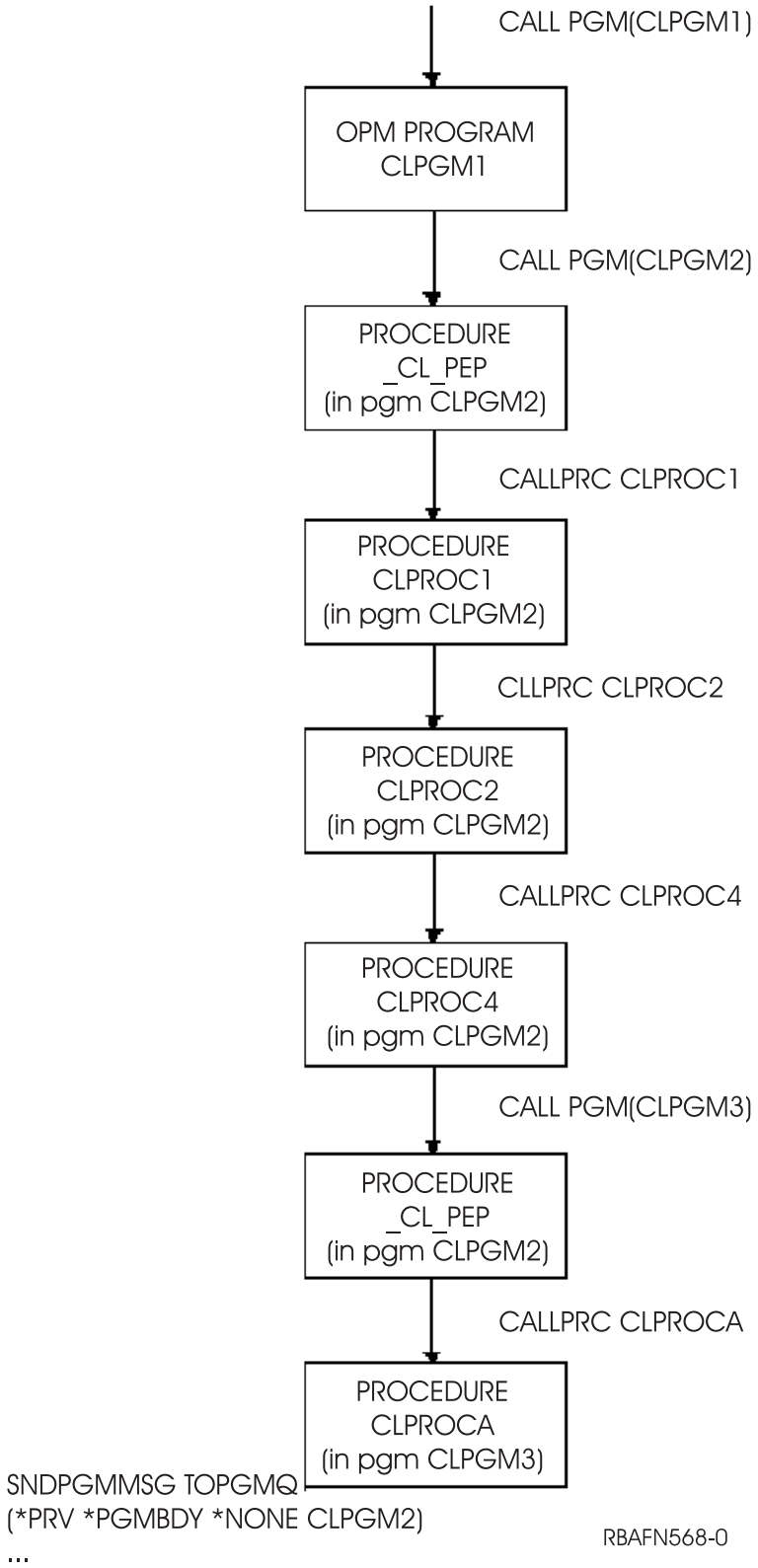
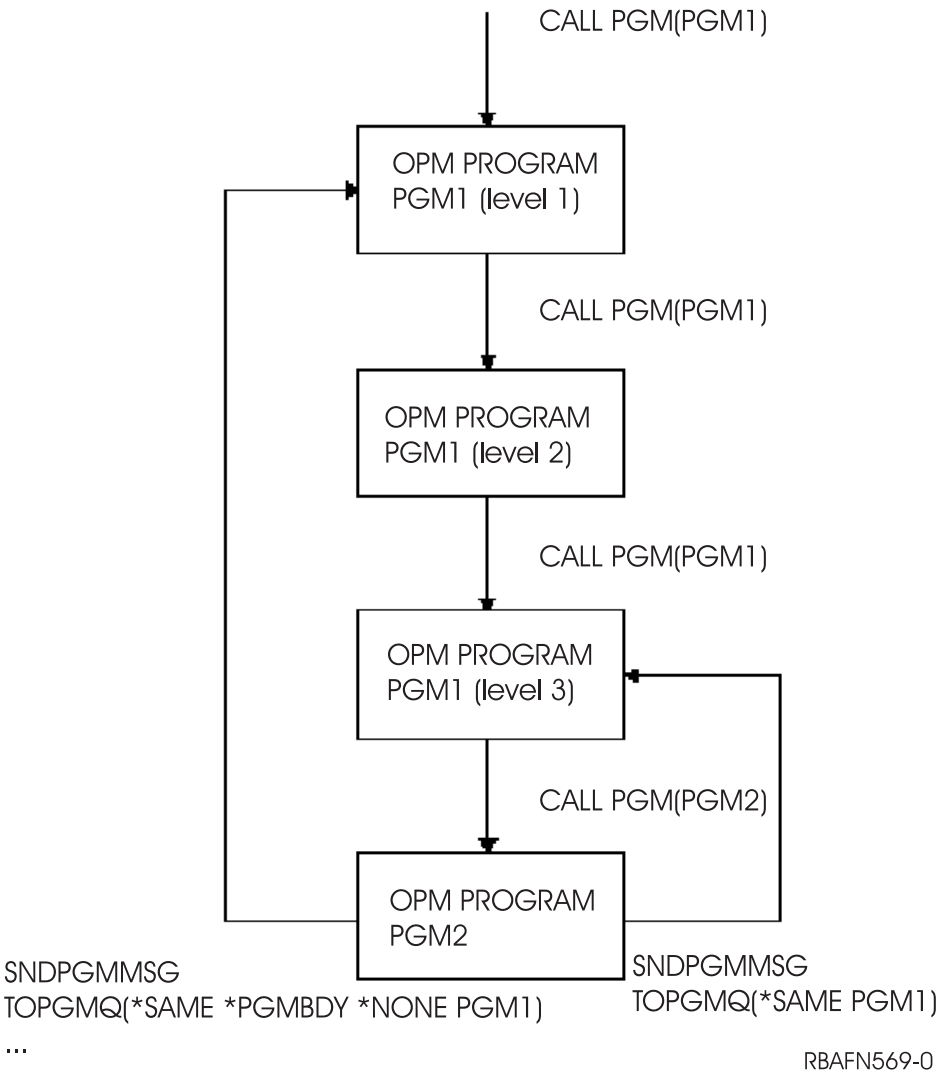
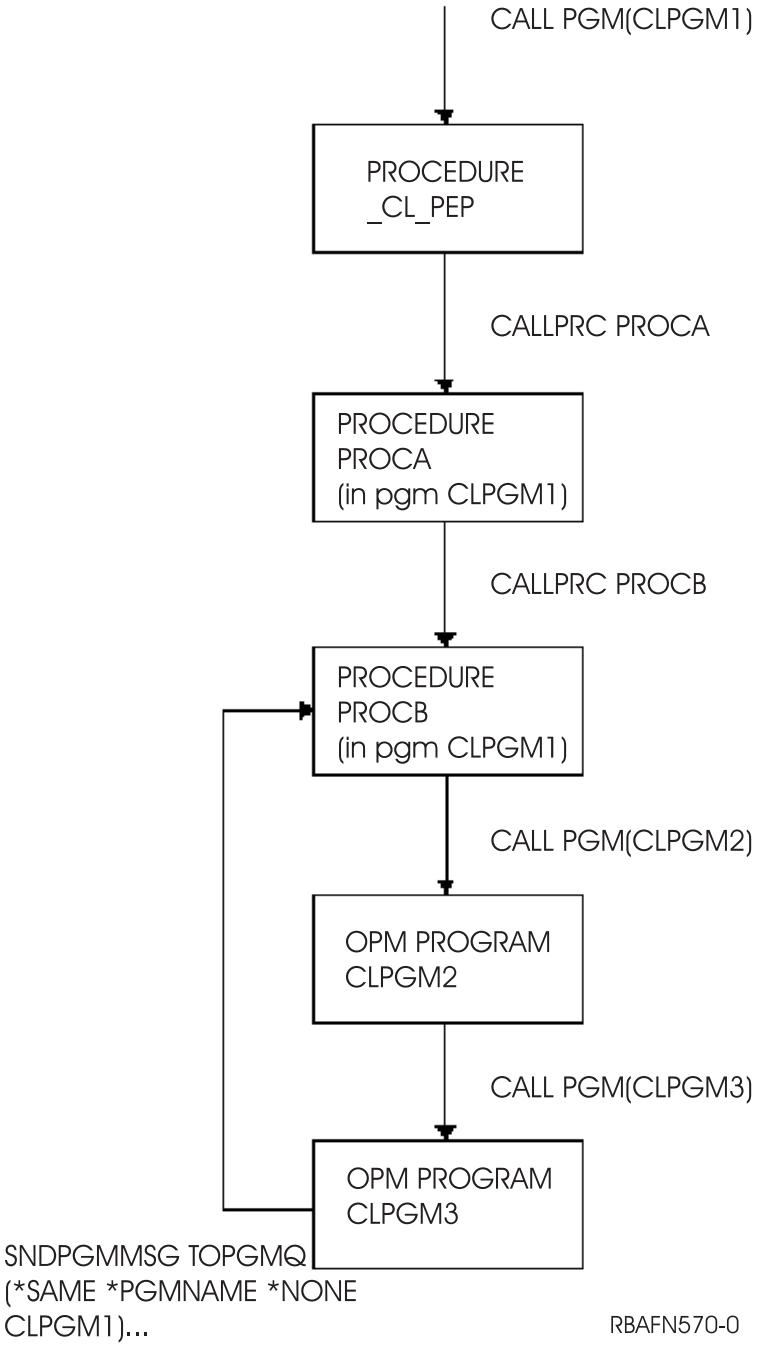
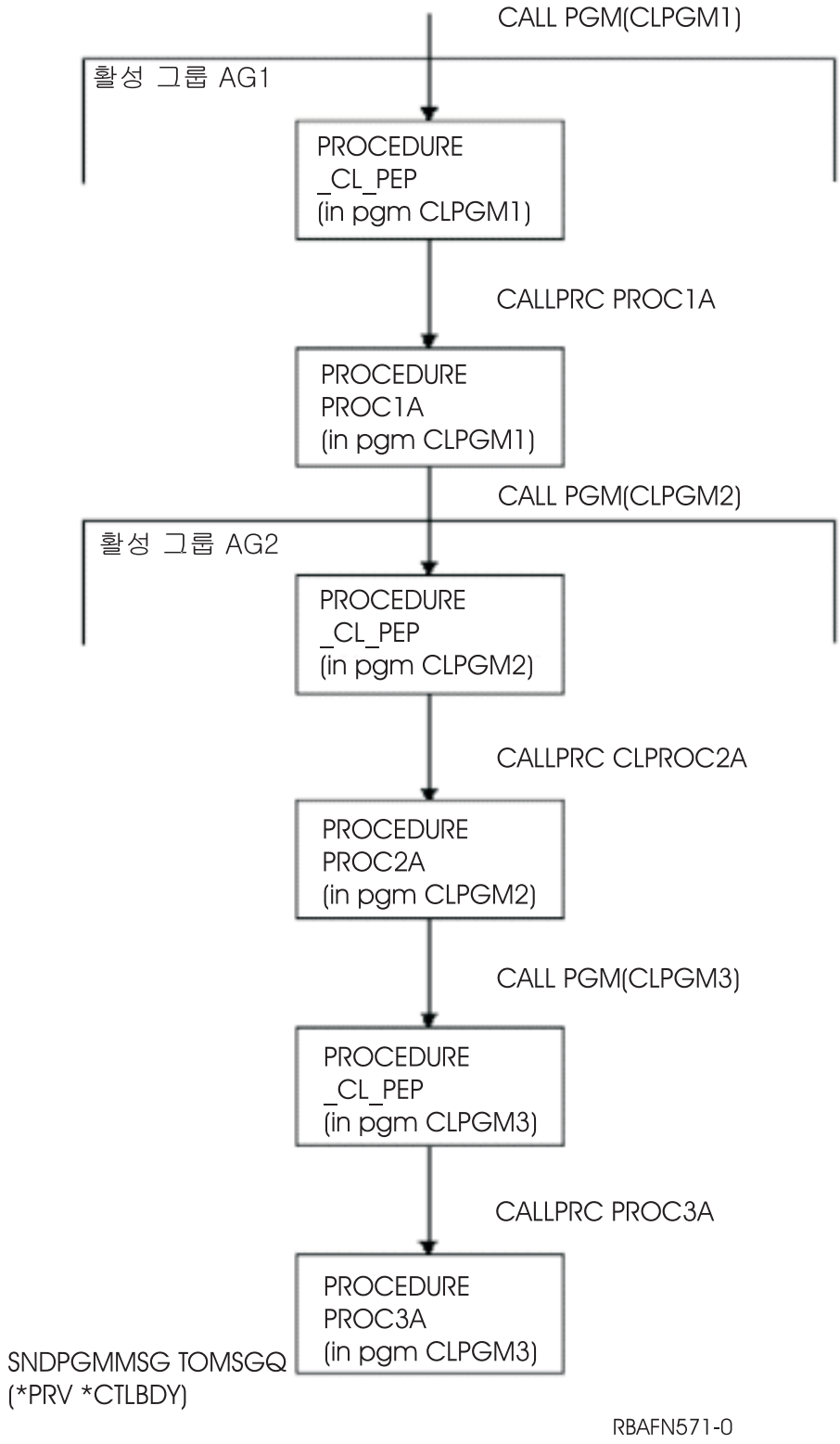
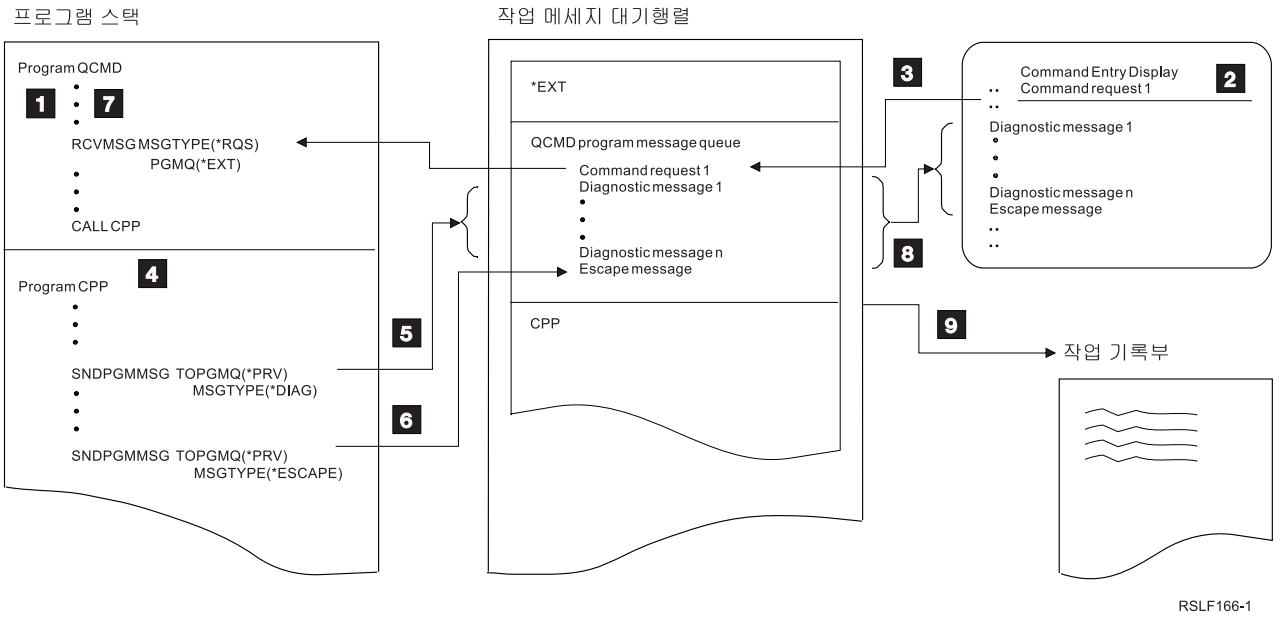
/* Can check for appropriate model number, release level, and so on */
RTVSYSVAL   SYSVAL(QMODEL) RTNVAR(&MODEL)
IF          (&MODEL *EQ 'xxxxx') THEN...

/* Install a library for new application (programs, data):         */
RSTLIB      SAVLIB(NEWAPP) DEV(&DEV) ENDOPT(*LEAVE) +
              MBROPT(*ALL)
/* Install a command to start new application:                     */
            RSTOBJ OBJ(NEWAPP) SAVLIB(QGPL) DEV(&DEV) +
              MBROPT(*ALL)

END:        ENDPGM
```

그림 3. LODRUN 명령을 사용한 어플리케이션의 예

제 7 장 메세지 정의

iSeries 서버에서는 프로시듀어 또는 프로그램 간, 작업 간, 사용자 간, 사용자와 프로시듀어 또는 프로그램 간의 통신이 메세지를 통해 이루어집니다. 메세지에는 사전정의 메세지나 즉시 메세지가 있습니다.

- 사전정의 메세지는 그것을 사용하는 프로그램의 외부에 작성되어 존재합니다. 사전정의 메세지는 메세지 파일에 저장되며 메세지 번호를 가집니다. 다음은 시스템 사전정의 메세지의 예입니다.

  ```
  CPF0006  명령에 오류가 있음.
  ```

- 즉시 메세지(immediate message)는 메세지가 송신될 때 송신자에 의해 작성됩니다. 즉시 메세지는 메세지 파일에 저장되지 않습니다. 다음은 표시장치에서 수신된 즉시 메세지의 예입니다.

  ```
  From  . . . :   QSYSOPR      06/12/94   10:50:54
  System going down at 11:00; please sign off
  ```

시스템에는 시스템 안의 프로그램 간 및 시스템과 사용자 간의 통신을 가능하게 하는 사전정의 메세지들이 많이 제공됩니다. 사용자가 주문한 사용권 프로그램에는 메세지 파일이 있으며, 이 파일은 이를 적용할 사용권 프로그램이 들어 있는 동일한 라이브러리에 저장됩니다. 예를 들면, 시스템 메세지는 라이브러리 QSYS 안의 파일 QCPFMSG에 저장됩니다.

시스템은 메세지 파일 안의 각 사전정의 메세지를 7자 코드로 고유하게 식별하며 메세지 설명을 사용하여 정의합니다. 메세지 설명에는 메세지 텍스트와 메세지 도움말 텍스트, 심각도 레벨, 유효한 응답 값 및 디폴트 응답 값, 그리고 기타 여러 속성들이 들어 있습니다. 온라인 도움말에서 ADDMSGD(메세지 설명 추가) 명령 설명을 참조하거나 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

시스템 안에서 송신되거나 수신된 모든 메세지는 메세지 대기행렬을 통해 전송됩니다. 명령과 같은 직접적인 요구에 대한 응답으로 발행된 메세지는 그것을 요구한 표시장치에 자동으로 표시됩니다. 기타 모든 메세지는 사용자, 프로그램 또는 프로시듀어가 대기행렬로부터 메세지를 수신하거나 메세지를 표시해야 합니다. 시스템에는 여러 IBM 제공 메세지 대기행렬이 있습니다. 이 메세지 대기행렬에 관해서는 이 장의 후반부에서 설명합니다. (236 페이지의 『메세지 대기행렬의 유형』 부분을 참조하십시오.)

또한 시스템은 발행된 메세지의 일부를 기록부에 기록합니다. 작업 기록부에는 작업을 위해 입력된 요구와 관련 정보가 들어 있으며, 이력 기록부에는 작업, 서브시스템 및 장치의 상태 정보가 들어 있습니다. 기록부에 대해 자세히 알려면 319 페이지의 『메세지 기록』 부분을 참조하십시오.

사용자는 사용자 고유의 메세지 파일과 메세지 설명을 작성할 수 있습니다. 사전정의된 메세지를 작성하여 여러 프로시듀어나 프로그램에서 동일한 메세지를 사용할 수 있으며, 이때 메세지는 한 번만 정의하면 됩니다. 또한 사전정의된 메세지를 사용하는 프로시듀어와 프로그램에 영향을 주지 않고 사전정의된 메세지를(메세지를 보는 사용자에 기초하여) 영어 이외의 언어로 변경하고 변환할 수 있습니다. 프로시듀어나 프로그램에 메세지가 정의된 경우 사용자가 메세지를 변경할 때 모듈이나 프로그램을 다시 컴파일해야 합니다.

사용자 고유의 메세지와 메세지 파일을 작성하는 것 이외에 시스템 메세지 처리 기능을 사용하여 다음을 수행할 수 있습니다.

- 메세지 대기행렬을 작성 및 변경합니다(CRTMSGQ[메세지 대기행렬 작성], CHGMSGQ[메세지 대기행렬 변경] 및 WRKMSGQ[메세지 대기행렬에 대한 작업] 명령).
- 메세지 파일을 작성 및 변경합니다(CRTMSGF[메세지 파일 작성], CHGMSGF[메세지 파일 변경] 명령).
- 메세지 설명을 추가합니다(ADDMSGD[메세지 설명 추가] 명령).
- 메세지 설명을 변경합니다(CHGMSGD[메세지 설명 변경] 명령).
- 메세지 설명을 제거합니다(RMVMSGD[메세지 설명 제거] 명령).
- 즉시 메세지(immediate message)를 송신합니다(SNDMSG[메세지 송신], SNDBRKMSG[일시 중단(break) 메세지 송신], SNDPGMMSG[프로그램 메세지 송신], SNDUSRMSG[사용자 메세지 송신] 명령).
- 메세지 및 메세지 설명을 표시합니다(DSPMSG[메세지 표시], DSPMSGD[메세지 설명 표시], WRKMSG[메세지에 대한 작업], WRKMSGD[메세지 설명에 대한 작업] 명령).
- 다음과 같이 CL 프로시듀어나 프로그램을 사용합니다.
  - 워크스테이션 사용자나 시스템 오퍼레이터에게 메세지를 송신합니다 (SNDUSRMSG[사용자 메세지 송신] 명령).
  - 메세지 대기행렬에 메세지를 송신합니다(SNDPGMMSG[프로그램 메세지 송신] 명령).
  - 메세지 대기행렬로부터 메세지를 수신합니다(RCVMSG[메세지 수신] 명령).
  - 메세지 대기행렬에 메세지에 대한 응답을 송신합니다(SNDRPY[응답 송신] 명령).
  - 메세지 파일에서 메세지를 검색합니다(RTVMSG[메세지 검색] 명령).
  - 메세지 대기행렬에서 메세지를 제거합니다(RMVMSG[메세지 제거] 명령).
  - 호출 메세지 대기행렬로 송신된 이탈, 통지 및 상태 메세지를 모니터합니다 (MONMSG[메세지 모니터] 명령).
- 시스템 응답 리스트를 사용하여 작업에 의해 송신된 사전정의 조회 메세지(predefined inquiry message)에 대한 응답을 지정합니다(ADDRPYLE[응답 리스트 항목 추가],

CHGRPYLE[응답 리스트 항목 변경], RMVRPYLE[응답 리스트 항목 제거], WRKRPYLE[응답 리스트 항목에 대한 작업] 명령).

메세지가 송신될 때 메세지는 다음 유형 중 하나로 정의됩니다.

- 정보(*INFO). 기능의 상태에 대한 정보를 전달하는 메세지.
- 조회(*INQ). 정보를 전달하면서 응답을 요구하는 메세지.
- 통지(*NOTIFY). 프로시듀어나 프로그램이 호출 프로시듀어나 프로그램으로부터의 정정 조치 또는 응답을 요구하는 상태를 설명하는 메세지. 프로시듀어나 프로그램은 호출한 프로그램이나 프로시듀어로부터의 통지 메세지 도착을 모니터할 수 있습니다.
- 응답(*RPY). 수신한 조회 메세지 또는 통지 메세지에 대한 응답 메세지.
- 송신자의 사본(*COPY). 송신자가 보유하고 있는 조회 메세지 또는 통지 메세지의 사본.
- 요구(*RQS). 수신 프로시듀어나 프로그램으로부터 기능을 요구하는 메세지. (예를 들면, CL 명령은 요구 메세지가 될 수 있습니다.)
- 완료(*COMP). 작업의 완료 상태를 전달하는 메세지.
- 진단(*DIAG). 시스템 처리 기능, 어플리케이션 프로그램 또는 입력 자료의 오류에 관한 메세지.
- 상태(*STATUS). 프로시듀어나 프로그램에 의해 수행된 작업의 상태를 설명하는 메세지. 프로시듀어나 프로그램은 호출한 프로그램이나 프로시듀어로부터의 상태 메세지 도착을 모니터할 수 있습니다. 외부 메세지 대기행렬(*EXT)로 송신된 상태 메세지는 표시장치에 표시되어 표시장치 사용자에게 처리 중인 작업을 알려줍니다.
- 이탈(*ESCAPE). 프로시듀어나 프로그램이 비정상 종료되어야 하는 상태를 설명하는 메세지. 프로시듀어나 프로그램은 호출한 프로그램이나 프로시듀어로부터의, 또는 기계로부터의 이탈 메세지 도착 상태를 모니터할 수 있습니다. 이탈 메세지가 송신된 후 제어는 송신 프로그램으로 리턴하지 않습니다.

다음은 이 장에서 설명될 내용입니다.

- 사용자 고유의 메세지 파일을 작성하는 방법
- 메세지 파일에 메세지 설명을 추가하는 방법
- 메세지 대기행렬 유형
- 메세지 대기행렬 작성 방법

메세지 파일 작성

사용자 고유의 사전정의 메세지를 작성하려면 먼저 메세지가 들어갈 메세지 파일을 작성해야 합니다. CRTMSGF(메세지 파일 작성) 명령을 사용하여 메세지 파일을 작성하십시오. 그러고 나서 ADDMSGD(메세지 설명 추가) 명령을 사용하여 메세지를 설명하고 그것을 메세지 파일에 넣으십시오.

CRTMSGF 명령에서 SIZE 매개변수에 K바이트 단위로 최대 *크기*를 지정할 수 있습니다. 다음 공식은 최대 *크기*를 결정하는 데 사용되는 공식입니다.

```
S + (I x N)
```

여기에서

S *기억장치의 초기 크기*

I *매번 추가되는 기억장치의 크기*

N *기억장치 추가 횟수*

S, I, N의 디폴트는 각각 10, 2, *NOMAX입니다.

예를 들어, S를 5로, I를 1 그리고 N을 2로 지정하겠습니다. 파일이 *초기 기억장치* (storage) *크기*인 5K에 이르면 시스템이 자동으로 *초기 기억장치*에 1K를 추가합니다. 추가된 양(1K)은 *기억장치*의 총 합이 최대 7K가 되도록 *기억장치*에 두 번 추가될 수 있습니다. *NOMAX를 N으로 지정하면 메세지 파일의 최대 *크기*가 16M이 됩니다.

메세지 파일에 최대 *크기*가 지정되어 있고 메세지 파일이 가득 차면 *크기*를 변경할 수 없습니다. 그러므로 새로운 메세지 파일을 작성하여 메세지를 새로운 파일에 재작성해야 합니다. MRGMSGF(메세지 파일 병합) 명령을 사용하면 메세지 파일 간에 메세지 설명을 복사할 수 있습니다. 이 단계를 수행하고 싶지 않으면 메세지 파일 작성 시 메세지 파일에 필요한 *크기*를 계산하는 것이 중요합니다. 그렇지 않으면 *NOMAX를 지정하는 것이 좋습니다.

독립 ASP의 메세지 파일

독립 보조 *기억장치* 풀(ASP)에 메세지 파일을 작성할 수 있으나 독립 ASP가 오프라인으로 바뀔 수 있으므로 권장되는 방법은 아닙니다. 독립 ASP가 오프라인이면 작업 기록부와 메세지 대기행렬의 메세지들이 올바로 표시되지 않습니다.

메세지 파일의 크기 결정

다음 공식을 사용하여 메세지의 *크기*를 결정할 수 있습니다(괄호 안의 내용은 ADDMSGD 명령 매개변수임).

- 메세지 색인 = *기본* 42바이트 + 메세지 길이
- 메세지 텍스트(MSG) = *기본* 16바이트 + 메세지 길이
- 메세지 온라인 도움말 정보(SECLVL)(있는 경우) = *기본* 16바이트 + 메세지 도움말 길이
- 형식(FMT)(있는 경우) = 14바이트 + (3 x FMTS의 수)
- 유형 및 길이(TYPE 및 LEN) = 48바이트
- 특수 값(SPCVAL) = 2 + (64 x SPCVAL의 수)
- 값(VALUES) = 32 x (VALUES의 수)

- 범위(RANGE) = 64바이트
- 관계(REL) = 관계의 길이
- 디폴트(DFT) = 디폴트 응답의 길이
- 디폴트 프로그램, 기록부 문제점 및 덤프 리스트(DFTPBM, LOGPRB, DMPLST) = 35 + (2 x DMPLST 안의 수)
- ALROPT = 12바이트

메세지 파일에서 최소 입력 길이는 59바이트이며 최대 입력 길이는 5764바이트입니다. 다음 표에 최대 입력이 표시되어 있습니다.

메세지 색인	42바이트
메세지 텍스트	148바이트
메세지 도움말 텍스트	3016바이트
99개의 형식	311바이트
유형 및 길이	48바이트
20개의 특수 값	1282바이트
20개의 값	640바이트
디폴트 응답 값	32바이트
디폴트 프로그램 및 덤프 리스트	233바이트
경고 옵션	12바이트

다음 예에서 CRTMSGF 명령은 메세지 파일 USRMSG를 작성합니다.

```
CRTMSGF   MSGF(QGPL/USRMSG) +
          TEXT('Message file for user-created messages')
```

OS/400용 RPG에서 DSPLY OP 코드와 함께 사용할 메세지 파일을 작성하려면 메세지 파일 이름을 반드시 QUSERMSG로 지정해야 합니다.

파일에 메세지 추가

ADDMSGD(메세지 설명 추가) 명령을 사용하여 사전정의 메세지를 설명하고 작성한 메세지 파일에 그것을 추가하십시오. ADDMSGD 명령에 메세지 ID, 메세지가 위치할 메세지 파일명, 메세지 설명을 지정하십시오. 메세지 설명에 다음을 지정할 수 있습니다.

- 선택적인 대체 변수가 있는 메세지 텍스트(필수적)
- 선택적인 대체 변수가 있는 메세지 도움말 텍스트
- 심각도 코드
- 대체 변수에 사용될 메세지 자료의 형식에 대한 설명
- 응답에 대한 유효성 검사 값
- 응답에 대한 디폴트 값
- 이탈 메세지에 대한 디폴트 메세지 처리 조치
- 작성 레벨
- 경고 옵션

- 오류 기록부의 항목
- 코드화 문자 세트 ID(CCSID)

메세지 설명이 들어갈 수 있는 항목에 대해서는 이후에 설명합니다

다음 명령들도 메세지 설명과 함께 사용할 수 있습니다.

CHGMSGD(메세지 설명 변경)
메세지 설명을 변경.

DSPMSGD(메세지 설명 표시)
메세지 설명을 표시합니다(이 명령으로 메세지 ID의 범위를 지정할 수 있음).

RMVMSGD(메세지 설명 제거)
메세지 파일에서 메세지 설명을 제거합니다.

RTVMSG(메세지 검색)
메세지 파일에서 메세지를 검색합니다.

MRGMSGF(메세지 파일 병합)
한 메세지 파일에서 다른 메세지 파일로 메세지를 병합합니다.

WRKMSGD(메세지 설명에 대한 작업)
파일 안의 메세지 리스트를 표시하고 메세지 설명을 추가, 변경 또는 삭제합니다.

메세지 ID의 할당

ADDMSGD 명령에 지정하는 메세지 ID는 메세지를 참조하는 데 사용되며 메세지 설명의 이름을 나타냅니다. 메세지 ID는 반드시 7자여야 합니다.

```
pppmmnn
```

여기서, ppp는 제품 또는 어플리케이션 코드, mm은 숫자로 된 그룹 코드, nn은 숫자로 된 부속 유형 코드(subtype code)입니다. mmnn으로 지정된 숫자는 제품 또는 어플리케이션 메세지를 분류하는 데 사용될 수 있습니다. 숫자 그룹과 부속 유형 코드는 0부터 9까지의 10진수와 A에서 F까지의 문자로 구성됩니다.

예를 들어, 다음과 같은 경우가 있습니다.

```
CPF1234
```

이 예는 CPF 메세지 1234를 나타냅니다.

사용자 고유의 메세지를 작성할 때 문자 U를 제품 코드의 첫 문자로 사용하는 것은 시스템 메세지와 구별할 수 있는 좋은 방법입니다. 예를 들어, 다음과 같은 경우가 있습니다.

```
USR3567
```

코드의 첫 번째 문자는 영문자여야 하지만 두 번째와 세 번째 문자는 영숫자도 가능합니다. 그룹 코드 및 부속 유형 코드는 0-9의 10진수와 A-F의 문자로 구성되어야 합니다. 이 범위는 16진수에서와 동일한 범위이기는 하지만 메세지 번호의 분류에서는 A부터 F까지를 문자로 취급된다는 점에 유의하십시오.

예를 들면, 메세지 설명의 범위를 표시할 때 CPFA000은 CPF1000보다 먼저 표시됩니다.

메세지 ID에 숫자 부속 유형 코드 00을 사용할 때는 주의해야 합니다. 이탈, 통지 또는 상태 메세지로 송신될 수 있으므로 그 결과 모니터가 가능한 메세지에 대해 숫자 부속 유형 00을 사용하는 경우 MONMSG(메세지 모니터) 명령의 부속 유형 코드 00은 숫자 그룹에 속한 모든 메세지를 모니터하게 합니다. 보다 자세히 알려면 280 페이지의 『CL 프로그램이나 프로시듀어에서의 메세지 모니터』 부분을 참조하십시오.

메세지 및 메세지 도움말 정의

ADDMSGD 명령에는 두 가지 레벨의 메세지를 정의할 수 있습니다. 메세지 텍스트는 필수 요소로서 메세지가 발행되는 상태를 식별합니다. 메세지 도움말은 선택적이며 상태를 상세히 설명하거나 수행해야 할 정정 조치를 설명합니다. 메세지 도움말을 보려면 표시장치 사용자는 메세지가 표시되었을 때 커서를 메세지 행으로 이동하여 도움말 키를 누르면 됩니다. 메세지 도움말은 세 개의 형식 제어 문자를 사용하여 표시장치에서 형식화될 수 있습니다. 이 제어 문자는 사용자가 메세지 도움말(대개의 경우 온라인 도움말 정보)을 편리하게 읽을 수 있도록 만드는 데 사용됩니다.

세 개의 형식 제어 문자는 모두 메세지 텍스트와 분리되도록 문자 다음에 공백을 사용합니다.

&Nb(b는 공백임)

텍스트를 새로운 행의 2열에서 시작합니다. 텍스트의 길이가 한 행보다 길 경우 다음 행들은 텍스트가 끝나거나 다른 형식 제어 문자가 발견될 때까지 4열부터 행 들여쓰기로 표시됩니다.

&Pb(b는 공백임)

텍스트를 새로운 행의 6열에서 시작합니다. 텍스트의 길이가 한 행보다 길 경우 다음 행들은 텍스트가 끝나거나 다른 형식 제어 문자가 발견될 때까지 4열부터 행 들여쓰기로 표시됩니다.

&Bb(b는 공백임)

텍스트를 새로운 행의 4열에서 시작합니다. 텍스트의 길이가 한 행보다 길 경우 다음 행들은 텍스트가 끝나거나 다른 형식 제어 문자가 발견될 때까지 6열부터 행 들여쓰기로 표시됩니다.

심각도 코드 할당

ADDMSGD 명령에서 메세지에 할당하는 심각도 코드는 메세지의 중요도를 나타냅니다. 심각도 코드가 높을수록 심각한 상태를 나타냅니다. 다음은 사용 가능한 심각도 코드와 그 의미입니다. (이 심각도 코드와 그 의미는 IBM 사전정의 메세지에 할당된 심각도 코드와 일치합니다.)

00: 정보. 단지 정보용임. 오류가 검출되지 않았으므로 응답이 필요없음. 이 메세지는 기능이 실행 중이거나 기능이 정상 완료되었음을 나타냄.

10: 경고. 잠재적인 오류 상태. 누락된 입력을 제공하는 등과 같이 프로시듀어나 프로그램이 디폴트를 사용할 수 있음. 조작 결과는 성공적인 것으로 가정됨.

20: 오류. 오류가 발견되었으나 그 오류에 자동 회복 프로시듀어가 적용되었음. 처리 계속됨. 잘못된 입력을 대체하기 위해 디폴트 조치를 수행할 수 있음. 조작의 결과가 유효하지 않을 수도 있음. 기능이 부분적으로만 완료될 수 있음. 예를 들면, 리스트의 일부 항목은 올바르게 처리되지만 다른 항목은 실패함.

30: 심각한 오류. 발견된 오류가 매우 심각하여 자동 회복될 수 없고 어떤 디폴트 조치도 가능하지 않음. 소스 자료에 오류가 있을 경우 입력 레코드 전체가 처리되지 않은 채 무시됨. 프로시듀어나 프로그램 처리 동안 오류가 발생하면 그 오류 때문에 프로시듀어나 프로그램이 비정상 종료됨(심각도 40). 조작 결과는 유효하지 않음.

40: 프로시듀어 또는 기능의 비정상 종료. 프로시듀어나 프로그램이 유효하지 않은 자료를 처리할 수 없거나 사용자의 취소로 인해 조작이 종료됨.

50: 작업의 비정상 종료. 작업이 종료되었거나 시작되지도 않았음. 또한 라우팅 단계가 비정상 종료되었거나 시작이 실패하였음. 작업 레벨 기능이 요구한 대로 수행되지 않거나 작업이 취소되었을 수 있음.

60: 시스템 상태. 시스템 오퍼레이터에게만 발행됨. 장치, 서브시스템 또는 시스템에 관한 상태 및 경고를 나타냄.

70: 장치 무결성. 시스템 오퍼레이터에게만 발행됨. 장치의 장애나 더 이상 작동하지 않음을 표시함. 이 경우에 사용자가 장애로부터 회복할 수 있거나 서비스 담당자의 도움이 필요할 때도 있음.

80: 시스템 경고. 심각도 코드 80의 메세지가 즉시 메세지로 발행됨. 시스템을 중단해야 할 정도로 심각하지는 않더라도 예방 조치를 수행하지 않을 경우에 더 심각한 상태가 발생할 가능성에 대한 경고.

90: 시스템 무결성. 시스템 오퍼레이터에게만 발행됨. 서브시스템이나 시스템이 작동 불가능하게 되는 상태를 설명함.

99: 조치. 응답 입력, 프린터 용지의 변경 또는 디스켓 대체 등 수동 조치가 필요함.

SEV 매개변수에 관한 자세한 설명은 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

대체 변수 정의

ADDMSGD 명령의 FMT 매개변수는 첫 번째 레벨이나 두 번째 레벨의 메세지에 대체 변수를 지정할 수 있습니다. 예를 들어, 다음과 같은 경우가 있습니다.

파일 &1이 없음

위의 예에는 대체 변수 &1이 들어 있습니다. 메세지가 표시되거나 검색될 때 변수 &1은 발견되지 않는 파일명으로 대체됩니다. 이 이름은 메세지의 송신자에 의해 제공됩니다. 예를 들어, 다음과 같은 경우가 있습니다.

파일 ORDHDRP가 없음

을

파일이 없음

과 비교해 보면, 대체 변수로 인해 보다 구체적이고 의미있는 메세지가 만들어졌음을 알 수 있습니다.

대체 변수는 &(앰퍼샌드)로 시작하고 그 뒤에 1에서 99까지의 숫자 중 하나인 n이 있어야 합니다. 한 예로 다음과 같은 메세지를 살펴보겠습니다.

파일 &1이 없음

이 경우 다음과 같이 대체 변수를 정의할 수 있습니다.

```
FMT((*CHAR 10))
```

대체 변수에 숫자를 할당할 때는 반드시 숫자 1로 시작하는 연속 숫자(예: &1, &2, &3 등)를 사용해야 합니다. 그러나 송신되는 메세지에서 메세지 설명에 대해 정의된 대체 변수를 모두 사용할 필요는 없습니다.

예를 들면,

파일 &3이 사용 가능하지 않음

메세지에서 &1과 &2가 사용되지 않더라도 메세지는 유효합니다. 이렇게 하려면 ADDMSGD 명령의 FMT 매개변수에 &1, &2 및 &3을 정의해야 합니다. 앞에 나온 메세지에 대한 FMT 매개변수는 다음과 같습니다.

```
FMT((*CHAR 10) (*CHAR 2) (*CHAR 10))
```

여기에서 첫 번째 값은 &1, 두 번째 값은 &2, 세 번째 값은 &3을 설명합니다. &3이 사용될 때는 반드시 &1과 &2에 대한 설명이 있어야 합니다. 또한 메세지가 송신될 때는 FMT 매개변수상에 설명된 모든 자료가 SNDPGMMSG(프로그램 메세지 송신) 명

령의 MSGDTA 매개변수에 들어 있어야 합니다. 앞에 나온 메세지를 송신하려면 MSGDTA 매개변수의 길이가 적어도 22자 이상이어야 합니다.

앞에 나온 메세지에 대해 FMT 매개변수를 다음과 같이 지정할 수도 있습니다.

```
FMT((*CHAR 0) (*CHAR 0) (*CHAR 10))
```

&1과 &2는 메세지에 사용되지 않으므로 길이 0으로 서술할 수 있습니다. 따라서 메세지 자료를 송신할 필요가 없습니다. (이 경우 SNDPGMMSG 명령의 MSGDTA 매개변수의 길이는 10자 이상이면 됩니다.)

메세지에 &3을 사용하고 FMT 매개변수에 &1과 &2가 들어 있는 예는 DMPLST 매개변수상에 &1과 &2가 지정되어 있을 경우입니다. (DMPLST 매개변수는 메세지가 이를 모니터하고 있지 않은 프로그램에 이탈 메세지로서 송신될 때 자료가 덤프됨을 지정합니다.)

대체 변수는 FMT 매개변수에 정의된 순서대로 메세지에 지정되지 않아도 됩니다. 예를 들어, FMT 매개변수 안의 세 개의 값을 다음과 같이 정의할 경우가 있습니다.

```
FMT((*CHAR 10) (*CHAR 10) (*CHAR 7))
```

이 경우 메세지에 대체 변수들이 다음과 같이 사용될 수 있습니다.

```
라이브러리 &2 안의 유형 &3의 오브젝트 &1이 사용 가능하지 않음
```

이 메세지가 CL 프로시듀어나 프로그램에서 송신된 경우 다음과 같이 메세지 자료에 사용된 값을 연결할 수 있습니다.

```
SNDPGMMSG .....MSGDTA(&OBJ *CAT &LIB *CAT &OBJTYPE)
```

ADDMSGD 명령에 자료 유형 및 메세지의 길이(선택적)를 지정하여 대체 변수에 대한 메세지 자료 필드의 형식을 지정해야 합니다. 다음은 메세지 자료 필드에 사용 가능한 자료 유형입니다.

- 인용 문자 스트링(*QTDCHAR). 어포스트로피로 묶인 문자 자료 스트링. 선행 및 후미 공백은 삭제되지 않습니다. 메세지 설명에 길이가 지정되지 않으면 송신자가 필드의 길이를 결정합니다.
- 문자 스트링(*CHAR). 어포스트로피로 묶이지 않은 문자 자료 스트링. 후미 공백이 삭제됩니다. 메세지 설명에 길이가 지정되지 않으면 송신자가 필드의 길이를 결정합니다.
- 변환 가능한 문자 스트링(*CCHAR). 어포스트로피로 묶이지 않은 문자 자료 스트링. 후미 공백이 삭제됩니다. 길이는 항상 송신자가 결정합니다. 이런 자료 유형이 65535 또는 65534 이외의 CCSID 태그를 가진 메세지 대기행렬로 송신되면 자료가 메세지 자료의 CCSID에서 메세지 대기행렬의 CCSID로 변환됩니다. 수신 또는 표시 기능을 사용하여 자료가 메세지 대기행렬로부터 확보될 경우에도 이런 자료 유

형이 변환될 수 있습니다. CCSID를 사용하는 메세지 핸들러 사용법에 관해서는 iSeries Information Center의 **프로그래밍** 범주에서 *국제화* 주제를 참조하십시오.

- 16진(*HEX). 앞에 문자 X가 있고 어포스트로피로 묶인 스트링. 스트링의 각 바이트는 두 개의 16진 문자(0-9 및 A-F)로 변환됩니다. 메세지 설명에 길이가 지정되지 않으면 송신자가 필드의 길이를 결정합니다.
- 2진(*BIN). 부호화 10진 정수로 형식화된 2진 정수(2, 4 또는 8바이트 길이). 지정된 길이를 제공하지 않으면 시스템이 2진 정수를 2로 가정합니다.
- 비부호화 2진(*UBIN). 비부호화 10진 정수로 형식화된 비부호화 2진 정수(2, 4 또는 8바이트 길이). 지정된 길이를 제공하지 않으면 시스템이 2진 정수를 2로 가정합니다.
- 10진(*DEC). 소수점이 있고 부호화 10진수로 형식화된 팩 10진수. 길이가 지정되어야 합니다. 소수 자릿수는 0이 디폴트입니다.
- 시스템 포인터(*SYP). 시스템 오브젝트에 대한 16바이트 포인터. 메세지 또는 메세지 도움말에서 10자의 오브젝트명은 *CHAR 유형 자료처럼 형식화됩니다.
- 공간 포인터(*SPP). 프로그램 오브젝트에 대한 16바이트 포인터. 덤프에서 오브젝트 안의 자료는 *HEX 유형의 자료처럼 형식화됩니다. 메세지에서는 *SPP를 대체 텍스트로 사용할 수 없습니다. 이것은 ADDMSGD 명령에서 DMPLST 매개변수의 일부로만 사용할 수 있습니다.

다음은 IBM 제공 메세지 설명에서만 유효한 자료 유형으로 다른 메세지에서는 사용할 수 없습니다.

- 시간 간격(*ITV). 다양한 대기 시간종료 상태에 대한 초 단위의 시간이 들어 있는 8바이트 시간 간격.
- 날짜 및 시간소인(*DTS). 8바이트의 시스템 날짜 및 시간소인으로, 날짜는 QDATFMT 및 QDATSEP 시스템 값에 지정된 대로 형식화되고, 시간은 hh:mm:ss로 형식화됩니다.

응답에 대한 유효성 검사 지정

ADDMSGD 명령에는 조회나 통지 메세지에 유효한 응답 유형을 지정할 수 있습니다. 다음 괄호 안에 주어진 매개변수를 지정할 수 있습니다.

- 응답 유형(TYPE)
  - 10진(*DEC)
  - 문자(*CHAR)
  - 영문자(*ALPHA)
  - 이름(*NAME)
- 응답의 최대 길이(LEN)
  - 10진수일 때는 15자리(9자리의 소수 자릿수)

- 문자 및 영문자일 때는 32자
- 이름일 때는 10자

주: 유효성 검사를 지정하지 않은 경우(VALUES, RANGE, REL, SPCVAL, DFT), 유형이 *CHAR 및 *ALPHA인 응답의 최대 길이는 132자입니다.

- 응답용으로 사용 가능한 값
  - 값의 리스트(VALUES)
  - 특수 값의 리스트(SPCVAL)
  - 값의 범위(RANGE)
  - 응답 값이 반드시 만족시켜야 하는 간단한 관계식

주: 특수 값이란 허용될 수는 있으나 다른 유효성 검사 값을 만족시키지 못하는 값을 의미합니다.

표시장치 사용자가 메세지에 대한 응답을 입력할 때 응답이 소문자로 입력되도록 키보드는 하단 지정 상태에 놓입니다. 사용자의 프로그램에 대문자로 응답할 필요가 있으면 다음 중 하나를 수행할 수 있습니다.

- 디폴트 소문자에서 대문자로 변환하는 변환 표 옵션을 지원하는 SNDUSRMSG 명령을 사용하십시오.
- 표시장치 사용자가 대문자만 입력하도록 VALUES 매개변수에 대문자만 지정하십시오.
- 해당 소문자를 대문자로 변환시키려면 VALUES 매개변수를 대문자로 지정하고 SPCVAL 매개변수를 사용하십시오.
- 입력되는 문자가 모두 A부터 Z까지의 문자일 때는 TYPE(*NAME)을 사용하십시오. 이때 문자는 검사되기 전에 대문자로 변환됩니다.

즉시 메세지 송신과 응답 처리

이 예에서는 프로시듀어가 다음을 수행합니다.

- 즉시 조회 메세지를 QSYSOPR에 송신함
- 예 또는 아니오(Y 또는 N) 응답을 요구함
- 유효한 응답이 입력되었는지 확인함
- 오퍼레이터가 120초 이내에 응답하지 않을 경우 시간종료가 수행됨.

```
                PGM
           DCL         &MSGKEY *CHAR LEN(4)
           DCL         &MSGRPY *CHAR LEN(1)
SNDMSG:    SNDPGMMSG   MSG('.... Reply Y or N') TOMSGQ(QSYSOPR) +
                         MSGTYPE(*INQ) KEYVAR(&MSGKEY)
           RCVMSG      MSGTYPE(*RPY) MSGKEY(&MSGKEY) WAIT(120) +
                         MSG(&MSGRPY)
           IF          ((&MSGRPY *EQ 'Y') *OR (&MSGRPY *EQ 'y')) DO
           .
```

```
            .
          GOTO END
            ENDDO      /* Reply of Y */
            IF         ((&MSGRPY *EQ 'N') *OR (&MSGRPY *EQ 'n')) DO
            .
            .
          GOTO END
            ENDDO      /* Reply of N */
            IF         (&MSGRPY *NE ' ') DO
            SNDPGMMSG  MSG('Reply was not Y or N, try again') +
                           TOMSGQ(QSYSOPR)
            GOTO       SNDMSG
            ENDDO      /* Reply not valid */
      /* Timeout occurred */
            SNDPGMMSG  MSG('No reply from the previous message +
                         was received in 120 seconds and a 'Y'' +
                         value was assumed') TOMSGQ(QSYSOPR)
            .
            .
END:        ENDPGM
```

SNDUSRMSG 명령이 시간종료 옵션을 지원하지 않기 때문에 이 프로시듀어에서는 대신 사용할 수 없습니다. (SNDUSRMSG가 응답을 수신하거나 작업이 취소될 때까지 대기합니다.)

SNDPGMMSG 명령은 메세지를 송신하고 KEYVAR 매개변수를 지정합니다. 이것은 이 메세지를 고유하게 식별하는 메세지 참조 키를 리턴시켜, 응답이 RCVMSG 명령과 적절하게 대응될 수 있도록 합니다. KEYVAR 값은 반드시 길이가 4인 문자 필드로 정의되어야 합니다.

RCVMSG 명령은 특정 메세지를 수신하기 위해 MSGKEY 매개변수에 SNDPGMMSG 명령의 메세지 참조 키 값을 지정합니다. 응답이 MSG 매개변수에 다시 전달됩니다. WAIT 매개변수는 시간종료 전에 응답을 대기하는 시간을 지정합니다.

응답이 수신되며 프로시듀어 논리가 Y 또는 N의 대문자나 소문자 값을 검사합니다. 일반적으로 오퍼레이터는 이 값을 소문자로 입력합니다. 오퍼레이터가 Y나 N 이외의 공백이 아닌 값을 입력하면 프로시듀어가 다른 메세지를 송신하고 나서 조회 메세지를 반복합니다.

오퍼레이터가 공백을 입력하면 프로시듀어로 응답을 송신하지 않습니다. 공백이 프로시듀어로 리턴되면 시간종료가 발생합니다(오퍼레이터가 응답하지 않음). 프로시듀어는 시스템 오퍼레이터에게 메세지를 송신하여 응답을 수신하지 못했으며 디폴트 값이 가정되었음을 알립니다. (‘Y’ 값은 메세지 대기행렬에서 ‘Y’로 나타납니다.) 가정 값 ‘Y’가 응답으로 표시되지 않으므로 사용자는 메세지 대기행렬을 볼 때 메세지가 응답되었는지 시간종료가 발생하였는지의 여부를 판별할 수 없습니다. 프로시듀어는 일단 메세지가 송신되면 메세지 대기행렬에서 메세지를 제거하지 않습니다. 두 번째 메세지는 이러한 문제들을 최소화시켜 무엇이 발생하였는지에 대한 감사 추적을 제공합니다.

시간종료 후 이루어진 메세지에 대한 오퍼레이터의 응답은 무시됩니다. 오퍼레이터는 자신의 응답이 무시되었다는 표시를 받지 못합니다.

2바이트 문자로 된 즉시 메세지 송신

2바이트 텍스트로 즉시 메세지를 송신하려면 텍스트를 37자의 2바이트 문자에 시프트 제어 문자를 더한 길이로 제한하십시오. 메세지의 *크기*를 제한함으로써 메세지를 보다 정확히 표시할 수 있습니다.

응답의 디폴트 값 정의

ADDMSGD 명령을 사용하면 메세지에 응답할 디폴트 값을 지정할 수 있습니다. 디폴트 응답의 유효성 검사 값은 그 메세지의 다른 응답의 검사 값과 동일해야 하며, 메세지 설명에 특수 값으로 지정되어야 합니다. 디폴트 값은 사용자의 메세지 대기행렬에 송신된 모든 조회 메세지에 대해 디폴트 응답이 발행됨을 사용자가 표시했을 때 (CHGMSGQ 명령을 사용하여) 사용됩니다. 또한 디폴트 응답은 응답되지 않은 조회 메세지가 삭제될 때도 송신됩니다. 예를 들어, 워크스테이션 사용자가 DSPMSG 명령을 사용하여 메세지를 표시하고 F13 키를 눌러 모든 메세지를 삭제하거나 F11 키를 눌러 특정 메세지를 삭제하는 등 응답되지 않은 조회 메세지를 제거할 수 있습니다.

INQMSGRPY의 작업 속성이 *DFT로 설정되거나 *SYSRPYL 옵션으로 설정될 때 디폴트 응답을 사용할 수 있습니다. 디폴트 응답을 변경하기 위해 시스템 응답 리스트를 사용할 수도 있습니다.

디폴트 응답은 프로그램 메세지 표시 화면에서도 사용됩니다(*EXT로 송신된 메세지를 표시). 디폴트 응답을 송신하는 것은 다음의 두 조건에서 발생합니다.

- 프로그램 메세지 표시 화면에 응답이 없는 조회 메세지가 표시되고 사용자는 응답을 입력하지 않고 Enter 키를 눌러 작업을 계속합니다.
- 사용자는 F3 키를 눌러 **프로그램 메세지 표시** 화면을 나갑니다.

이탈 메세지에 대한 디폴트 메세지 처리 지정

이탈 메세지로서 송신 가능한, 각 사용자 작성 메세지에 대해서는 메세지가 송신될 때 다른 방법으로 처리되지 않을 경우에 사용할 디폴트 메세지 처리 조치를 설정할 수 있습니다.

디폴트 메세지 조치는 다음으로 구성됩니다.

- 디폴트 프로그램명. 메세지를 처리하기 위해 호출되어 디폴트 조치를 수행하는 프로그램. 다음 매개변수들은 디폴트 프로그램으로 전달됩니다.
  - 메세지 대기행렬명 호출. 이 매개변수는 시스템이 메세지를 송신한 곳을 식별하는 많은 필드로 이루어진 구조입니다. 디폴트 메세지 처리 종료 프로그램과 매개변수의 필드에 대한 자세한 정보는 **iSeries Information Center** 프로그래밍 범주의 *API* 섹션에 나와 있는 메세지 처리 API를 참조하십시오.

  - 메세지 참조 키(4자). 호출 메세지 대기행렬상의 이탈 메세지의 메세지 참조 키

- 덤프 리스트. 덤프될 오브젝트를 표시하는 메세지 자료 필드 번호(대체 변수와 동일한 번호) 리스트.

  이외에도 다음 사항을 덤프할 수 있습니다.

  - 작업의 자료 영역
  - 작업의 내부 기계 자료 구조
  - 작업

    작업에 대해 덤프 리스트를 지정하려면 매개변수 JOB(*) OUTPUT(*PRINT)으로 DSPJOB(작업 표시) 명령을 지정하면 됩니다.

메세지 설명에 디폴트 조치를 지정하지 않으면 (DSPJOB JOB(*) OUTPUT(*PRINT)이 지정된 것처럼) 작업을 덤프할 수 있습니다.

메세지에서 지정된 디폴트 조치는 이탈 메세지를 처리하지 않고 메세지 여과 조치가 완료된 이후에만 수행됩니다. 디폴트 처리에 대해 자세히 알려면 284 페이지의 『디폴트 처리』를 참조하십시오.

디폴트 프로그램의 예

다음 프로그램은 진단 메세지와 이탈 메세지를 차례로 송신할 때 사용이 가능한 샘플 디폴트 프로그램입니다. 이 프로그램은 이러한 단일 CL 프로시듀어를 가진 OPM CL 프로그램이나 ILE 프로그램이 될 수 있습니다.

```
             PGM        PARM(&MSGQ &MRK)
DCL VAR(&MRK) TYPE(*CHAR) LEN(4)
             DCL        VAR(&MSGQ) TYPE(*CHAR) LEN(6381)
             DCL        VAR(&QNAME) TYPE(*CHAR) LEN(4096)
             DCL        VAR(&MODNAME) TYPE(*CHAR) LEN(10)
             DCL        VAR(&BPGMNAME) TYPE(*CHAR) LEN(10)
             DCL        VAR(&BLANKMRK) TYPE(*CHAR) LEN(4) VALUE(' ')
             DCL        VAR(&DIAGMRK) TYPE(*CHAR) LEN(4) VALUE(' ')
             DCL        VAR(&SAVEMRK) TYPE(*CHAR) LEN(4)
             DCL        VAR(&MSGID) TYPE(*CHAR) LEN(7)
             DCL        VAR(&MSGDTA) TYPE(*CHAR) LEN(100)
             DCL        VAR(&MSGF) TYPE(*CHAR) LEN(10)
      DCL VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
             DCL        VAR(&OFFSET) TYPE(*DEC)
             DCL        VAR(&LENGTH) TYPE(*DEC)

             /* Check for OPM program type                          */

             IF         (%SST(&MSGQ 277 1) *EQ '0') THEN(DO)
               CHGVAR     VAR(&QNAME) VALUE(%SST(&MSGQ 1 10))
               CHGVAR     VAR(&MODNAME) VALUE('*NONE')
               CHGVAR     VAR(&BPGMNAME) VALUE('*NONE')
 ENDDO
       ELSE DO
             /* Not an OPM program; always use the long procedure name */
               CHGVAR     VAR(&OFFSET) VALUE(%BIN(&MSGQ 281 4))
               CHGVAR     VAR(&LENGTH) VALUE(%BIN(&MSGQ 285 4))
               CHGVAR     VAR(&QNAME) VALUE(%SST(&MSGQ &OFFSET &LENGTH))
               CHGVAR     VAR(&MODNAME) VALUE(%SST(&MSGQ 11 10))
               CHGVAR     VAR(&BPGMNAME) VALUE(%SST(&MSGQ 1 10))
 ENDDO
 GETNEXTMSG: CHGVAR     VAR(&SAVEMRK) VALUE(&DIAGMRK)
                 RCVMSG     PGMQ(*SAME (&QNAME &MODNAME &BPGMNAME)) +
                            MSGTYPE(*DIAG) RMV(*NO) KEYVAR(&DIAGMRK)
```

```
            IF         (&DIAGMRK *NE &BLANKMRK) THEN(GOTO GETNEXTMSG)
            ELSE IF (&SAVEMRK *NE ' ') THEN(DO)
            /* If no diag message is sent, no message is sent to the previous program */
                RCVMSG     PGMQ(*SAME (&QNAME &MODNAME &BPGMNAME)) +
                           MSGKEY(&SAVEMRK) RMV(*NO) MSGDTA(&MSGDTA) +
                           MSGID(&MSGID) MSGF(&MSGF) MSGFLIB(&MSGLIB)
                SNDPGMMSG  MSGID(&MSGID) MSGF(&MSGLIB/&MSGF) +
                           MSGDTA(&MSGDTA) TOPGMQ(*PRV (&QNAME +
                           &MODNAME &BPGMNAME))
            MSGTYPE(*ESCAPE)
 ENDDO
     ENDPGM
```

프로그램은 FIFO순으로 모든 진단 메세지를 수신합니다. 그런 후 프로그램은 이전 프로그램이 그것을 모니터할 수 있도록 최종 진단 메세지를 이탈 메세지로 송신합니다.

경고 옵션 지정

ADDMSGD 명령에 경고 옵션을 지정하여 메세지에 대한 경고를 작성할 수 있습니다. 경고가 작성될 수 있는 메세지는 SNA(Sytem Network Architecture) 경고가 작성되도록 하여 문제 관리 중재점으로 송신합니다. 메세지에 대해 작성된 경고는 ADDALRD(경고 설명 추가) 명령을 사용하여 정의됩니다. OS/400에 대한 자세한 내용은 DSNX Support  책을 참조하십시오.

메세지 설명의 예

다음 예에서는 ADDMSGD 명령이 주문 입력과 같은 어플리케이션에 사용될 메세지를 작성합니다. 다음은 입력된 고객 번호를 화면에서 찾을 수 없을 때 나타나는 메세지입니다. 메세지는 다음과 같습니다.

```
고객 번호 &1이 없음
```

다음은 이 메세지에 대한 ADDMSGD 명령입니다.

```
ADDMSGD   MSGID(USR4310) +
          MSGF(QGPL/USRMSG)  +
          MSG('Customer number &1 not found') +
          SECLVL('Change customer number') +
             SEV(40) +
          FMT((*CHAR 8))
```

메세지가 라이브러리 QGPL 안의 USRMSG 파일에 추가됩니다.

DSPMSGD 또는 WRKMSGD 명령을 사용하면 메세지 설명을 인쇄하거나 표시할 수 있습니다.

SECLVL 매개변수는 아주 간단한 텍스트를 제공합니다. 이것을 '추가 메세지 정보' 화면에 표시하려면 SECLVL('메세지 텍스트')을 지정하십시오. 커서를 이 메세지에 위치시킨 후 도움말 키를 누르면 이 매개변수에 지정한 텍스트가 '추가 메세지 정보' 화면에 나타납니다.

2바이트 메세지 정의

2바이트 텍스트로 메세지를 정의하려면 ADDMSGD 명령을 사용하여 CL 프로시듀어나 프로그램을 작성하십시오. 정의된 메세지는 메세지 파일에 저장된 뒤 정상적으로 송신됩니다. 프로그램을 작성할 때에는 다음과 같이 하십시오.

1. 프로그램이 들어 있는 소스 파일이 2바이트 파일인지 확인하십시오. CRTSRCPF(소스 실제 파일 작성) 명령에 IGCDTA(*YES)를 지정하십시오.
2. 소스 입력 유틸리티(SEU)를 사용하여 프로그램을 입력하십시오. 2바이트 문자를 사용하는 CL 명령은 SEU로만 입력될 수 있습니다. 따라서 2바이트 메세지는 CL 프로그램으로 작성되어야 합니다.
3. 메세지 전체가 표시되거나 인쇄될 수 있도록 메세지의 길이를 37자 이내의 2바이트 문자로 제한하십시오.

   MONMSG 명령을 사용할 때는 CMPDATA(자료 비교) 매개변수를 6자 이내의 2바이트 문자로 제한하십시오.
4. 2바이트 메세지 파일이 영숫자 메세지 파일(예: 2바이트 표시장치로만 송신되는 변환된 메세지 파일)을 대체할 때는 다음과 유사한 명령을 입력하여 영숫자 메세지 파일을 대체하십시오.

   ```
   OVRMSGF MSGF(QCPFMSG) TOMSGF(DBCSLIB/QCPFMSG)
   ```

   2바이트 메세지는 2바이트 표시장치에서만 표시될 수 있습니다.

시스템 메세지 파일 탐색

탐색이 이루어질 때 시스템은 다음 두 단계를 사용하여 메세지 파일로부터 메세지를 검색합니다.

1. 시스템이 메세지 파일명에 유효한 대체를 처리합니다.

   보다 자세히 알려면 232 페이지의 『메세지 파일 대체』를 참조하십시오.
2. 메세지 파일명이 대체되지 않으면 메세지가 사용될 때 지정된 메세지 파일명과 라이브러리에 근거하여 시스템이 메세지 파일을 탐색합니다.

   자세히 알려면 『메세지 파일에 대한 탐색』을 참조하십시오.

메세지 파일에 대한 탐색

메세지 파일이 대체되지 않으면 (메세지 파일이 송신될 때) 지정된 메세지 파일명과 라이브러리를 사용하여 메세지 설명이 검색되는 메세지 파일을 탐색합니다.

메세지 파일명이 대체되었지만 메세지 ID가 대체된 파일에 들어 있지 않은 경우에도 지정된 메세지 파일명과 라이브러리를 사용하여 메세지 파일을 탐색합니다.

시스템 탐색은 사용자가 메세지 파일 라이브러리를 *CURLIB로 지정했는지 아니면 *LIBL로 지정했는지에 따라 달라집니다. 다음은 *CURLIB와 *LIBL의 탐색 경로에 대한 설명입니다.

- *CURLIB로 지정하거나 명시적으로 메세지 파일 라이브러리를 지정하십시오.

  시스템이 지정된 라이브러리나 작업의 현재 라이브러리(*CURLIB)에서 명명된 메세지 파일을 탐색합니다.
- 메세지 파일 라이브러리를 *LIBL로 지정하십시오.

  시스템이 작업의 라이브러리 리스트(*LIBL)에서 명명된 메세지 파일을 탐색합니다. 지정된 이름이 있는 첫 번째 메세지 파일을 찾으면 탐색이 중단됩니다.

메세지 파일이 발견되었지만 메세지 ID에 대한 설명이 없으면, QCPFMSG에 있는 메세지 CPF2457의 메세지 속성과 텍스트가 누락 메세지 설명 대신 사용됩니다.

메세지 파일이 발견되지 않으면 시스템은 그 메세지가 송신된 시간에 사용되었던 메세지 파일로부터 메세지를 검색합니다.

주: 메세지 파일이 있더라도 손상 또는 권한 부여상의 문제로 인해 액세스하지 못할 수 없습니다.

메세지 파일 대체

프로시듀어나 프로그램에서 사용된 메세지 파일은 대체시킬 수 있습니다. 메세지 파일 대체 작성(메세지 파일 대체 명령), 삭제(대체 삭제 명령) 및 표시(대체 표시 명령)는 기타 유형의 대체와 유사합니다. 그러나 여기에서는 속성이 아닌 메세지 파일명만이 대체되며 대체를 적용하는 규칙도 약간씩 다릅니다.

메세지 파일을 대체하려면 OVRMSGF(메세지 파일 대체) 명령을 사용하십시오. 대체된 파일은 MSGF 매개변수에 지정되고, 그것을 대체하는 파일은 TOMSGF 매개변수에 지정됩니다.

예를 들어, 이름이 USRMSGF인 사용자 메세지 파일을 QCPFMSG로 대체하려는 경우 다음 명령을 사용할 수 있습니다.

```
OVRMSGF MSGF(QCPFMSG) TOMSGF(USRMSGF)
```

사전정의 메세지가 검색되거나 표시될 때 메세지 설명을 찾기 위해 대체 파일이 탐색됩니다. 이 파일에서 메세지 설명이 없으면 대체되는 파일이 탐색됩니다.

메세지 파일을 대체하는 이유는 다음과 같습니다.

- 변경된 디폴트 응답이나 덤프 리스트를 제공하기 위해. 메세지 파일은 변경된 디폴트 응답 또는 덤프 리스트를 가진 메세지 설명을 사용하여 작성되는데 이것은 원래의 메세지 설명에 있는 디폴트 응답이나 덤프 리스트가 만족스럽지 않기 때문입니다. 각각 다른 디폴트 응답을 사용하는 여러 운영 환경을 설정할 수 있습니다.

- 메세지의 심각도 레벨을 변경하기 위해.
- 디폴트 프로그램을 제공하기 위해.
- 메세지 텍스트를 변경하기 위해. 텍스트가 비어 있을 경우에는 메세지가 송신되지 않은 것처럼 사용자에게 표시됩니다. 예를 들면, CPYF(파일 복사) 명령이 송신하는 상태 메세지를 표시하지 않는 것을 원할 경우입니다.
- 메세지를 자국어로 번역하기 위해. 영어로 기록된 메세지 파일을 다른 언어로 작성된 메세지 파일로 대체시킬 수 있습니다. (모든 메세지를 변경할 경우 작업에 대한 라이브러리 리스트를 사용하여 메세지 파일을 대체하는 대신 메세지 파일의 순서를 변경할 수 있습니다.)

메세지에서 메세지 파일을 선택하는 또 다른 방법은 작업에 대한 라이브러리 리스트의 파일 순서를 변경해서 검색하는 것입니다. 그러나 이 접근 방법을 사용하면 지정된 이름을 발견한 첫 번째 메세지 파일에서 메세지 탐색이 중단됩니다. 메세지가 그 파일에 없으면 탐색이 중단됩니다.

예를 들어, USRMSG로 명명된 메세지 파일이 라이브러리 USRLIB1에 있고 USRMSG로 명명된 다른 메세지 파일이 라이브러리 USRLIB2에 있다고 가정하겠습니다. 이때 USRLIB1에 있는 메세지 파일을 사용하기 위해서는 이 라이브러리 리스트에서 USRLIB1이 USRLIB2보다 앞에 있어야 합니다.



시스템은 그 이름을 가진 첫 번째 메세지 파일을 탐색합니다. 해당 파일에 메세지가 없으면, 탐색이 중단됩니다. 그러나 OVRMSGF 명령을 사용하는 경우에는 시스템이 대체하는 파일을 탐색하고, 메세지가 없으면, 대체된 파일을 탐색합니다.

메세지 파일 대체의 예

어떤 작업에서 사용하기 위해 IBM 제공 메세지를 변경할 경우가 있습니다. 예를 들면,

```
YYY 안의 유형 *ZZZ인 오브젝트 XXX가 삭제됨
```

위의 메세지 CPC2191을 아래와 같이 변경하는 것으로 가정하겠습니다.

```
YYY 안의 오브젝트 XXX가 삭제됨
```

CPC2191에 대한 상세한 설명을 표시하여 FMT 매개변수를 설명하는 방법을 지정하십시오.

먼저, 메세지 파일을 작성하십시오.

```
CRTMSGF MSGF(USRMSG/OVRCPF)
```

그런 후 사용자 메세지를 기초로 하여 메세지 CPC2191을 메세지 파일에 추가하십시오.

```
ADDMSGD  MSGID(CPC2191) MSGF(USRMSG/OVRCPF) +
         MSG('Object &1 in &2 deleted') +
         SEV(00) FMT((*CHAR 10) (*CHAR 10))
```

다음 작업을 수행할 때 메세지 파일을 대체하려면 OVRMSGF 명령을 사용하십시오.

```
OVRMSGF MSGF(QCPFMSG) TOMSGF(USRMSG/OVRCPF)
```



모든 작업에 사용하기 위해 이 메세지를 변경하려는 경우 CHGMSGD(메세지 설명 변경) 명령을 사용하여 메세지를 변경할 수 있습니다. 그러면 시스템 메세지 파일을 대체하지 않아도 됩니다.

CHGMSGD 명령을 사용하여 IBM 제공 메세지를 변경할 경우 시스템의 새로운 릴리스가 설치될 때 메세지를 다시 변경해야 합니다. 메세지를 다시 변경하려는 경우 언제든지 실행시킬 수 있는 입력 스트림이나 프로그램에 변경 내용을 넣을 수 있습니다.

또한 대체 파일도 대체할 수 있습니다. 예를 들면, 작업 중에 다음과 같은 OVRMSGF 명령을 지정할 수 있습니다.

```
OVRMSGF  MSGF(MSGFILE1)  TOMSGF(MSGFILE2)
OVRMSGF  MSGF(MSGFILE2)  TOMSGF(MSGFILE3)
```

맨 먼저 파일 MSGFILE1이 MSGFILE2로 대체됩니다. 두 번째로 MSGFILE2가 MSGFILE3으로 대체됩니다. 메세지가 송신될 때 파일들은 다음 순서로 탐색됩니다.

1. MSGFILE3
2. MSGFILE2
3. MSGFILE1

메세지 파일이 대체되지 않도록 할 수 있습니다. 이렇게 하려면 OVRMSGF 명령에 SECURE 매개변수를 지정해야 합니다.

메세지 대기행렬의 유형

시스템상의 모든 메세지는 메세지 대기행렬로 송신됩니다. 메세지 대기행렬과 관련된 시스템 사용자나 프로그램은 대기행렬로부터 메세지를 수신합니다. 이와 마찬가지로, 메세지에 대한 응답은 사용자 또는 응답을 요구하는 프로그램의 메세지 대기행렬로 다시 송신됩니다.

다음 다이어그램은 IBM에서 제공하는 메세지 대기행렬을 보여줍니다. 메세지 대기행렬은 각각의 표시장치(DSP01과 DSP02는 표시장치명임) 및 각 사용자 프로파일(BOB와 RAY는 사용자 프로파일명)별로 제공됩니다.



RBAFN560-0

시스템상에서 수행 중인 각 작업에 대해 작업 메세지 대기행렬이 제공됩니다. 각 작업에 외부 메세지 대기행렬(*EXT)이 제공되어, 작업 안에서 OPM 프로그램이나 ILE 프

로시듀어를 호출할 때마다 자체의 호출 메세지 대기행렬을 가집니다.



시스템 이력 기록부(QHST)와 시스템 오퍼레이터(QSYSOPR)에 대해서도 메세지 대기행렬이 제공됩니다.



메세지 대기행렬은 다음과 같이 사용됩니다.

- 워크스테이션 메세지 대기행렬은 워크스테이션 사용자 간 및 워크스테이션 사용자와 시스템 오퍼레이터 간에 메세지를 송수신하는 데에 사용됩니다. 대기행렬명은 워크스테이션명과 동일합니다. 워크스테이션이 시스템에 설명되면 시스템에서 대기행렬을 작성합니다.
- 사용자 프로파일 메세지 대기행렬은 사용자 간의 통신에 사용될 수 있습니다. 사용자 프로파일이 작성될 때 사용자 프로파일 메세지 대기행렬이 라이브러리 QUSRSYS에 자동으로 작성됩니다.
- 작업 메세지 대기행렬은 처리될 요구(명령과 같은)를 수신하고 요구를 처리한 결과로 얻는 메세지의 송신에 사용됩니다. 메세지는 작업의 리퀘스터에게 송신됩니다. 작업 메세지 대기행렬은 각 작업마다 존재하며, 작업이 존재하는 동안에만 존재합니다. 작업 메세지 대기행렬은 외부 메세지 대기행렬(*EXT)과 일련의 호출 스택 입력 메세지 대기행렬로 이루어집니다. 자세히 알려면 242 페이지의 『작업 메세지 대기행렬』을 참조하십시오.
- 시스템 오퍼레이터 메세지 대기행렬(QSYSOPR)은 시스템, 표시장치 사용자 및 어플리케이션 프로그램으로부터 메세지를 수신하고 이 메세지에 응답하기 위해 사용됩니다.
- 이력 기록부 메세지 대기행렬은 시스템 안의 작업으로부터 이력 기록부(QHST)로 정보를 송신하는 데 사용됩니다.

이 메세지 대기행렬 외에도 메세지를 시스템 사용자와 어플리케이션 프로그램 간에 송신하기 위한 사용자 고유의 사용자 메세지 대기행렬을 작성할 수 있습니다.

메세지 대기행렬의 작성 또는 변경

사용자 자신의 사용자 메세지 대기행렬을 작성하려면 CRTMSGQ(메세지 대기행렬 작성) 명령을 사용하십시오. 또한 CHGMSGQ(메세지 대기행렬 변경) 명령을 사용하여 메세지 대기행렬의 다음과 같은 속성을 변경할 수 있습니다.

다음은 메세지 대기행렬의 속성입니다.

- 메세지 대기행렬에 대한 변경 내용이 디스크에 즉시 기록되어야 하는지의 여부. 변경 내용을 디스크에 즉시 기록하면 시스템 장애 시에도 메세지가 유실되지 않습니다. 그러나 시스템 성능이 저하됨에 유의하십시오.
- 메세지 대기행렬에 도착한 메세지의 전달 방법. 메세지 대기행렬이 작성될 때 전달 방법은 보류 전달로 정의됩니다. 표시장치가 사인 온될 때 사용자의 메세지 대기행렬은 사용자 프로파일에 지정된 모드로 설정됩니다. 다음은 사용자가 CHGMSGQ 명령에 지정할 수 있는 전달 유형입니다.
  - 일시 중단 전달. 작업이 인터럽트된 후 메세지를 전달하는 프로그램이 호출됩니다. 사용자 프로그램이 일시 중단 전달을 요구하는 CHGMSGQ 명령에 지정되지 않았거나 *SAME이 지정되었을 때는 DSPMSG(메세지 표시) 명령이 자동으로 메세지를 표시합니다. 작업에 대한 일시 중단(break) 메세지는 CHGJOB 명령의 BRKMSG 매개변수로 제어할 수 있습니다.
  - 통지 전달. 주의 표시등이나 경보(또는 둘 다)를 사용하여 메세지가 대기행렬에 있음을 표시장치 사용자에게 알립니다. 표시장치 사용자는 DSPMSG 명령을 사용하여 메세지를 볼 수 있습니다.
  - 보류 전달. 표시장치 사용자가 DSPMSG 명령으로 메세지를 요구할 때까지 메세지 대기행렬에서 메세지를 보류합니다.
  - 디폴트 전달. 모든 메세지가 무시되며, 응답을 요구하는 메세지가 메세지의 디폴트 응답으로 송신됩니다.
- 일시 중단 전달 시의 메세지 처리 방법.
  - 자동으로 DSPMSG 명령을 수행함. 대화식 작업인 경우 심각도 코드가 상당이 높으면 표시장치에 메세지가 표시됩니다. 일괄처리 작업인 경우 심각도 코드가 상당히 높으면 스풀 프린터 파일에 메세지가 나열됩니다.
  - 메세지를 처리하기 위해 일시 중단 처리 프로그램을 호출함. CHGMSGQ 명령을 사용하여 호출된 프로그램을 지정하고 전달 방법을 일시 중단(break) 모드로 설정해야 합니다. 일시 중단 처리 프로그램을 사용하여 일시 중단 모드로 만든 경우 다른 작업들이 대기행렬에 있는 메세지를 조회하기 위해 응답할 수 있는지를 지정할 수 있습니다.

- 일시 중단 및 통지 전달 메세지를 선택하기 위한 심각도 코드. 지정된 최소 심각도 코드 이상의 심각도 코드를 가진 메세지가 표시됩니다. 대기행렬이 작성되면 최소 심각도 코드는 00으로 설정됩니다. 최소 심각도 코드를 변경하려면 CHGMSGQ 명령을 사용하십시오.

  DSPMSG 명령을 사용하여 메세지 대기행렬에 있는 메세지를 표시할 경우 표시되는 메세지를 선택하기 위해 심각도 코드 필터(SEV) 매개변수가 사용됩니다. 이 필터는 작성 시 메세지 대기행렬에 대해 지정된 심각도 필터보다 우선적으로 사용됩니다. 이 필터를 사용하려면 DSPMSG SEV(*MSGQ)를 지정하십시오. DSPMSG 명령을 사용하여 일시 중단(break) 및 통지 메세지를 선택하는 데 사용되는 현재 심각도 코드를 판별할 수 있습니다. 이 코드는 메세지 화면의 머리말 행에 표시됩니다.
- 메세지 대기행렬과 연관된 코드화 문자 세트 ID(CCSID). 이 대기행렬로 송신된 메세지는 이 CCSID로 변환됩니다. 메세지 대기행렬 CCSID가 65534나 65535이면, 변환되지 않습니다. 메세지 대기행렬 CCSID가 65534이면, 각 메세지에는 송신자가 설정한 자체 CCSID가 들어 있습니다.
- 표준 메세지 대기행렬에 경고 허용. 경고 허용은 작성 중인 대기행렬이 그곳으로 송신된 경고 메세지로부터 경고가 생성되는 것을 허용하는지 여부를 지정합니다.
- 메세지 대기행렬이 가득 찼을 때 수행할 수 있는 조치. 메세지 대기행렬 QHST에 대해 이 속성을 변경할 수 없으나 메세지 대기행렬이 가득 찼을 경우 QHST가 CPF2460을 송신합니다. IBM은 원래 설정된 이 속성과 함께 QSYSOPR을 제공합니다.
  - 가득 찬 대기행렬로 메세지를 송신하는 프로그램이나 사용자에게 CPF2460(메세지 대기행렬을 확장할 수 없음)을 송신합니다.
  - 대기행렬을 랩시킵니다. 랩이 이루어질 경우 대기행렬에 있는 메세지를 제거하여 대기행렬로 송신되는 신규 메세지를 위한 공간이 만들어집니다.

주: 워크스테이션 장치 설명이 작성되면 시스템은 해당 장치가 모든 조치 메세지를 수신하도록 메세지 대기행렬을 설정합니다. 워크스테이션 프린터, 테이프 드라이브 및 APPC 장치의 경우 장치 설명을 작성할 때 MSGQ 매개변수를 사용하여 메세지 대기행렬을 지정할 수 있습니다. 이들 장치에 대해 메세지 대기행렬이 지정되지 않으면 QSYSOPR이 디폴트 메세지 대기행렬로 사용됩니다. 다른 모든 장치는 작성 시, QSYSOPR 메세지 대기행렬에 할당됩니다.

사용자 프로파일에 정의된 메세지 대기행렬을 사용자 메세지 대기행렬이라고 합니다. 사용자 프로파일을 사용하여 시스템을 사인 온하면 사용자 메세지 대기행렬은 사용자 프로파일에 지정된 전달 모드로 됩니다.

하나의 표시장치를 사인 온한 후 다른 표시장치를 사인 온하는 동안에 사용자 메세지 대기행렬이 일시 중단 또는 통지 전달 모드이면, 사용자 메세지 대기행렬이 새로운 사인 온을 위해 전달 모드를 변경할 수 없습니다. 메세지 대기행렬이 다른 작업에 대해

일시 중단 또는 통지 전달 모드에 있는 경우 사용자 메세지 대기행렬(워크스테이션 메세지 대기행렬 및 QSYSOPR 메세지 대기행렬과 함께)의 전달 모드는 작업에 의해 변경할 수 없습니다.

표시장치를 사인 오프하거나 작업이 예기치 못하게 종료될 때 이 작업에 대한 사용자 메세지 대기행렬 전달 모드가 일시 중단 또는 통지 모드이면, 사용자 메세지 대기행렬 전달 모드는 보류 모드로 변경됩니다. 또한 사용자가 대체 작업으로 전송할 때에도 사용자 메세지 대기행렬 전달 모드는 일시 중단 또는 통지 모드에서 보류 모드로 변경됩니다. TFRSECJOB(2차 작업 전송) 명령을 사용하거나 시스템 요구 키를 누른 후 시스템 요구 메뉴상에서 옵션 1을 지정하여 이를 수행할 수 있습니다.

대체 작업으로 전송한 후 사용자 프로파일을 사용하여 사인 온할 수 있습니다. 사용자 메세지 대기행렬 전달 모드는 사용자 프로파일에 지정된 전달 모드에 놓입니다. 따라서 사용자 메세지 대기행렬은 대체 작업으로 전송될 수 있습니다. 그런 후 사용자는 이 두 작업 사이에서 상호 전송할 수 있고 사용자 메세지 대기행렬을 선택할 수 있습니다.

그러나 대체 작업으로 전송한 후 사용자가 자신의 것이 아닌 사용자 프로파일을 사용하여 사인 온하게 되면 전송된 작업에 대한 사용자 메세지 대기행렬이 보류 전달 모드가 됩니다. 사용자가 사인 온한 사용자 프로파일에 대한 사용자 메세지 대기행렬은 사용자 프로파일에 지정된 전달 모드가 됩니다. 따라서 사용자 메세지 대기행렬은 다른 사용자에 의해 일시 중단 또는 통지 전달 모드로 변경될 수 있습니다. 사용자가 첫 번째 작업으로 되돌아왔을 때 다른 사용자가 자신의 사용자 메세지 대기행렬을 계속 그 전달 모드에 두고 있는 경우 사용자의 메세지 대기행렬 전달 모드는 원래의 전달 모드로 변경될 수 없습니다.

QSYSOPR 메세지 대기행렬은 시스템 오퍼레이터를 위한 메세지 대기행렬입니다(변경되지 않는 한). 위와 같은 상태는 시스템 오퍼레이터에게도 발생할 수 있습니다.

독립 ASP의 메세지 대기행렬

독립 ASP에 관한 자세한 정보는 독립 ASP 주제를 참조하십시오.

독립 ASP의 메세지 대기행렬을 중단 모드로 만들어서는 안됩니다. 메세지 대기행렬이 중단 모드이면 메세지 대기행렬로 메세지를 송신할 때 메세지 대기행렬이 스레드의 라이브러리 이름 공간에 없을 경우 중단 프로그램이 호출되지 않습니다. 스레드의 ASP 그룹 가운데 독립 ASP에 있는 라이브러리 그리고 시스템 ASP(ASP 번호 1)와 기본 사용자 ASP(ASP 번호 2-32)에 있는 라이브러리들이 모두 함께 스레드를 위한 라이브러리 이름 공간을 구성합니다.

메세지 대기행렬에 조회 메세지를 송신할 때 송신 메세지 대기행렬과 응답 메세지 대기행렬 모두 시스템 ASP에 있거나 같은 독립 ASP에 있어야 합니다. 그렇지 않으면 두 개의 메세지 대기행렬 중 어느 하나가 오프라인일 때 응답 메세지 대기행렬로 응답이 송신되지 않습니다.

다음 상황에서는 메세지를 수신할 수 없습니다.

- 대기 상태의 메세지 대기행렬로부터 단절변환 상태의 독립 ASP에서
- 메세지 대기행렬로 송신된 조회 메세지에 대한 응답으로서 단절변환 상태의 독립 ASP에서

일시 중단 처리 프로그램은 스레드를 위한 라이브러리 이름 공간을 변경할 수 없습니다.

일시 중단(break) 처리 프로그램

심각도 코드 필터보다 높은 심각도를 가진 메세지가 일시 중단 전달(break delivery) 모드에 있는 메세지 대기행렬에 도착할 때마다 일시 중단 처리 프로그램이 호출됩니다. 일시 중단 처리 프로그램을 요구하려면 CHGMSGQ 명령에 프로그램명과 일시 중단 전달 이름을 동시에 지정해야 합니다. 메세지 처리 프로그램은 RCVMSG(메세지 수신) 명령으로 메세지를 수신하여 메세지를 처리된 것으로 표시하고, 프로그램이 다시 호출되지 않도록 해야 합니다. 메세지 수신과 일시 중단(break) 처리 프로그램에 대해 자세히 알려면 제 8 장『메세지에 대한 작업』부분을 참조하십시오.

주: 인터럽트된 프로그램이 장치 화면으로부터 입력 자료를 기다리고 있으면 이 프로그램이 화면 파일을 열 수 없습니다.

사용자는 시스템 응답 리스트를 사용하여 지정된 사전정의 조회 메세지에 시스템이 응답하도록 지정할 수 있습니다. 그러면 표시장치 사용자는 응답을 하지 않아도 됩니다. 자세히 알려면 315 페이지의『시스템 응답 리스트의 사용』부분을 참조하십시오.

전달 모드 변경의 예

시스템이 시작되면 제어 서브시스템이 시작될 때 시스템이 QSYSOPR 메세지 대기행렬을 일시 중단 전달로 만듭니다. 그러나 시스템 오퍼레이터가 사인 오프하면 메세지 대기행렬은 보류 전달 모드로 됩니다. 시스템 오퍼레이터가 다시 사인 온하면 QSYSOPR은 QSYSOPR 사용자 프로파일에 지정된 모드로 됩니다.

CL 초기 프로그램의 다음 프로시듀어를 사용하여 QSYSOPR 메세지 대기행렬을 구분 모드에 놓을 수 있습니다. 초기 프로그램은 유사한 프로시듀어를 사용하여 사용자의 자체 사용자 프로파일에서 지정된 하나의 메세지 대기행렬 이외의 메세지 대기행렬을 모니터할 수 있습니다.

```
PGM /* Procedure to place a msg queue in break mode */
CHGMSGQ  QSYSOPR DLVRY(*BREAK) SEV(50)
MONNMSG   MSGID(CPF0000) EXEC(SNDPGMMSG MSG('Unable to put QSYSOPR +
          message queue in *BREAK mode') TOPGMQ(*EXT))
     ENDPGM
```

프로시듀어는 일시 중단 전달에 대한 QSYSOPR 메세지 대기행렬을 심각도 50으로 설정하려고 시도합니다. 이러한 시도가 실패할 경우 메세지는 외부 작업 메세지 대기행렬(*EXT)로 송신됩니다. 이 프로시듀어가 들어 있는 프로그램이 종료되면 초기 메뉴가

표시됩니다. 심각도 레벨 50은 워크스테이션 사용자를 인터럽트하는 일시 중단 메세지의 수를 줄이는 데 사용됩니다. 일반적인 실패 원인은 다른 사용자가 이미 QSYSOPR을 구분 모드로 만들었기 때문입니다.

작업 메세지 대기행렬

시스템의 각 작업에 대해서는 작업 메세지 대기행렬이 작성되어 작업의 메세지 요구사항을 모두 처리합니다. 단일 작업에 대한 작업 메세지 대기행렬은 하나의 외부 메세지 대기행렬(*EXT)과 일련의 호출 메세지 대기행렬로 구성됩니다. 호출 메세지 대기행렬은 작업 안에서 호출된 각 ILE 프로시듀어와 OPM 프로그램에 할당됩니다. 그 외에도, 각 작업에 대해 작업 기록부가 작성됩니다. 작업 기록부는 작업 안에서 송신된 모든 메세지가 시간 순서로 유지보수되는 논리 대기행렬입니다. 메세지를 *EXT 대기행렬이나 호출 메세지 대기행렬로 송신할 수 있습니다. 메세지를 작업 기록부에는 송신하지 않습니다. 오히려 시스템이 *EXT나 호출 메세지 대기행렬로 송신된 메세지도 작업 기록부에 논리적으로 추가합니다.

외부 메세지 대기행렬(*EXT)은 작업의 외부 리퀘스터(표시장치 사용자와 같은)와 통신을 하는 데 사용됩니다. 작업의 외부 메세지 대기행렬로 송신된 상태 메세지를 제외한 메세지도 작업 기록부에 기록됩니다. (자세히 알려면 319 페이지의 『작업 기록부』 부분을 참조하십시오.)

정보 메세지, 조회 메세지 또는 통지 메세지가 대화식 작업에서 외부 메세지 대기행렬로 송신되면 메세지가 '프로그램 메세지 표시' 화면에 표시되고 프로시듀어는 표시장치 사용자로부터 조회 또는 통지 메세지에 대한 응답을 기다립니다. 사용자가 응답을 입력하지 않고 Enter 키나 F3(나감) 키를 누르면, 디폴트 메세지 응답이 메세지 송신자에게 리턴됩니다. 디폴트 메세지 응답이 없을 경우에는 *N이 송신됩니다. 일괄처리 작업에 대한 외부 메세지 대기행렬로 조회 또는 통지 메세지를 송신하면 시스템이 디폴트 응답을 사용자에게 송신합니다. 디폴트 메세지 응답이 없을 경우에는 *N이 응답합니다. 시스템 응답 리스트에서 조회의 표시 또는 조회에 대한 디폴트 응답 송신이 *EXT로 대체될 수도 있습니다.

상태 메세지가 대화식 작업의 외부 메세지 대기행렬로 송신되면 그 메세지는 표시장치의 메세지 행에 표시됩니다. 사용자는 이와 같은 상태 메세지를 사용하여 표시장치 사용자에게 수행 시간이 긴 조작의 진행 상황을 알려줍니다. 예를 들면, 사용자가 여러 개의 멤버를 가진 파일을 복사하는 경우 CPYF 명령을 수행할 때 시스템은 상태 메세지를 송신합니다.

주: 어플리케이션이 장시간 수행 조작을 완료하고 나면 사용자는 화면에 나타난 메세지 행을 지우기 위해 다른 메세지를 송신해야 합니다. 이를 위해 사용자는 다음과 같이 공백 메세지인 CPI9801을 사용할 수 있습니다. 예를 들어, 다음과 같은 경우가 있습니다.

```
         PGM
.
.
.
SNDPGMMSG  MSGID(CPF9898)  MSGF(QCPFMSG)  MSGDTA('Status 1')  +
           TOPGMQ(*EXT) MSGTYPE(*STATUS)
.
.
.
SNDPGMMSG  MSGID(CPF9898)  MSGF(QCPFMSG)  MSGDTA('Status 2')  +
           TOPGMQ(*EXT) MSGTYPE(*STATUS)
.
.
.
SNDPGMMSG  MSGID(CPI9801)  MSGF(QCPFMSG)  TOPGMQ(*EXT)  +
            MSGTYPE(*STATUS)
.
.
.
     ENDPGM
```

호출 메세지 대기행렬은 어떤 프로그램이나 프로시듀어와 또 다른 프로그램이나 프로시듀어 사이에서 메세지를 송신하는 데 사용됩니다. 프로그램이나 프로시듀어가 호출 스택(아직 리턴되지 않은)에 있는 동안에는 *그것*의 호출 메세지 대기행렬이 활동하며 메세지가 프로그램이나 프로시듀어로 송신될 수 있습니다. 일단 프로그램이나 프로시듀어가 리턴되면 *그것*의 호출 메세지 대기행렬은 더 이상 존재하지 않게 되며 메세지도 더 이상 송신될 수 없습니다. 호출 메세지 대기행렬로 송신될 수 있는 메세지 유형에는 정보, 요구, 완료, 진단, 상태, 이탈 및 통지 메세지 등이 있습니다.

OPM 프로그램이나 ILE 프로시듀어의 호출 메세지 대기행렬은 해당 프로그램이나 프로시듀어가 호출될 때 작성됩니다. 호출 메세지 대기행렬은 프로그램이나 프로시듀어가 실행 중인 호출 스택 항목에만 배타적으로 연관됩니다. 호출 스택 항목을 식별하여 호출 메세지 대기행렬을 간접적으로 식별합니다. 호출 스택 항목은 해당 호출 스택 항목에서 실행 중인 프로그램이나 프로시듀어의 이름으로 식별됩니다.

OPM 프로그램의 경우 연관된 호출 스택 항목이 10자(까지)의 프로그램명으로 식별됩니다. ILE 프로시듀어의 경우 연관된 호출 스택 항목이 세 부분, 즉 256자(까지)의 프로시듀어명, 10자(까지)의 모듈명 및 10자(까지)의 프로그램명으로 구성되어 식별됩니다. 모듈명은 프로시듀어가 컴파일되었던 모듈의 이름입니다. ILE 프로그램명은 모듈이 바인드된 ILE 프로그램의 이름입니다.

ILE 프로시듀어에 대한 호출 스택 항목을 식별할 때에는 프로시듀어명만 지정해도 됩니다. 프로시듀어명 자체가 호출 스택 항목을 고유하게 식별하지 않는 경우 모듈명이나 ILE 프로그램명도 지정될 수 있습니다. 메세지가 송신될 때 프로그램이나 프로시듀어가 호출 스택에 두 개 이상 있는 경우 지정된 이름이 가장 최근에 호출된 프로그램이나 프로시듀어의 발생을 식별합니다.

호출 스택 항목을 식별하는 방법에는 이 외에도 또 다른 방법이 있습니다. 이러한 방법들은 255 페이지의『SNDPGMMSG상의 호출 스택 항목 식별』부분에서 자세히 설명됩니다.

OPM 또는 ILE 프로그램이 컴파일된 후 호출 스택에 있는 동안 대체된 경우 호출 스택 항목을 참조하는 데 프로그램명이 사용될 때 주의해야 합니다. 대체 조작이 수행된 시점보다 먼저 스택상에 있던 호출 스택 항목의 경우 현재 QRPLOBJ에 상주하는 대체된 오브젝트로 이름 참조가 해결됩니다. 이들 이름 참조는 대체된 오브젝트가 QRPLOBJ 라이브러리에 상주하는 한 유효합니다. 스택상의 항목이 대체 조작이 수행된 시점보다 최근의 것이면, 이름 참조는 프로그램의 새로운 버전에 대한 것입니다. 사용될 버전이 결정되는 방식 때문에 프로그램을 라이브러리 QRPLOBJ에 직접 놓아서는 안됩니다. 이 라이브러리는 프로그램의 대체된 버전에 배타적으로 사용되어야 합니다. 직접 QRPLOBJ에 놓인 프로그램에 대한 이름 참조는 실패합니다.

프로그램 오브젝트가 호출 스택에 상주하는 동안 제거되거나 이름이 변경된 경우 제거된 프로그램에 대한 이름 참조나 기존 이름을 사용한 이름 참조는 실패합니다. ILE 프로시듀어의 경우 참조할 프로시듀어와 모듈명만 사용하면 프로그램의 이름 변경이 이름 참조에 영향을 주지 않습니다. ILE 프로그램명을 사용할 경우에도, 이름 참조가 실패합니다.

프로그램 또는 프로시듀어의 호출 스택 항목에 대한 메세지 대기행렬은 프로그램 또는 프로시듀어가 종료하면 더 이상 사용 가능하지 않습니다. 연관된 호출 메세지 대기행렬상의 메세지는 메세지의 메세지 참조 키를 사용하여 해당 시점에서만 참조할 수 있습니다.

예를 들어, 프로시듀어 A가 프로시듀어 B를 호출하는 프로시듀어 C를 호출한다고 가정해 보십시오. 프로시듀어 C는 프로시듀어 B로 메세지를 송신한 후 종료합니다. 메세지가 프로시듀어 B를 사용할 수 있습니다. 그러나 프로시듀어 B가 종료하면 그 호출 메세지 대기행렬은 더 이상 사용할 수 없습니다. 그 결과 작업 기록부에 메세지가 나오더라도 프로시듀어 A를 사용해서는 프로시듀어 B에 액세스할 수 없습니다. 프로시듀어 A에 해당 메세지에 대한 메세지 참조 키가 없는 한 프로시듀어 A가 프로시듀어 B로 송신된 메세지에 액세스할 수 없습니다.



프로시듀어 A가 특정 메세지를 삭제해야 하는 경우 사용자는 다음을 수행할 수 있습니다.

- 프로시듀어 C가 프로시듀어 A로 특정 메세지를 송신함
- 프로시듀어 B가 프로시듀어 A로 메세지를 재송신함

다음 그림은 프로시듀어 호출, 작업 메세지 대기행렬 및 호출 스택 항목 대기행렬 간의 관계입니다. 연결선(-----)은 어떤 메세지 대기행렬이 어떤 프로시듀어 호출과 연관되는지를 나타냅니다.



앞의 그림에서 프로시듀어 B는 각 프로시듀어 호출에 대해 하나씩 두 개의 호출 스택 항목 대기행렬을 가집니다. 프로시듀어 C로는 메세지가 송신되지 않았으므로 프로시듀어 C의 메세지 대기행렬은 없습니다. 프로시듀어 C가 프로시듀어 B로 메세지를 송신하면 이 메세지는 프로시듀어 B의 최종 호출을 위해 호출 스택 항목 대기행렬로 갑니다.

주: 명령 입력 화면을 사용 중일 때 F10(상세 메세지 포함) 키를 눌러 작업 메세지 대기행렬로 송신되는 모든 메세지를 표시할 수 있습니다. 메세지가 표시되면 사용자는 화면이동 키 중 하나를 사용하여 이들을 차례로 볼 수 있습니다.

또한 DSPJOBLOG(작업 기록부 표시) 명령을 사용하여 작업에 대한 메세지를 표시할 수도 있습니다.

제 8 장 메세지에 대한 작업

이 장에서는 사용자와 프로그램 간의 통신에서 메세지가 사용되는 방법을 설명합니다. 메세지는 다음과 같이 송신될 수 있습니다.

- 메세지의 리시버가 현재 시스템을 사용하지 않더라도 하나의 시스템 사용자로부터 다른 시스템 사용자로
- 하나의 OPM 프로그램이나 ILE 프로시듀어로부터 다른 OPM 프로그램이나 ILE 프로시듀어로
- 프로그램 또는 프로시듀어로부터 시스템 사용자에게로(메세지의 리시버가 현재 시스템을 사용하지 않더라도)

대화식 시스템 사용자는 즉시 메세지 및 응답만 송신할 수 있습니다.

OPM 프로그램이나 ILE 프로시듀어는 사용자 정의 자료가 있는 즉시 메세지나 사전정의 메세지를 송신할 수 있습니다. 또한 프로그램이나 프로시듀어는 다음을 수행할 수 있습니다.

- 메세지 수신
- 메세지 파일에서 메세지 설명을 검색하고 그 메세지를 프로그램 변수에 배치
- 메세지 대기행렬에서 메세지 제거
- 메세지 모니터

시스템 사용자에게 메세지 송신

다음은 메세지를 시스템 사용자에게 송신하는 명령입니다.

- 메세지 송신(SNDMSG)
- 일시 중단 메세지 송신(SNDBRKMSG)
- 프로그램 메세지 송신(SNDPGMMSG)
- 사용자 메세지 송신(SNDUSRMSG)

SNDPGMMSG와 SNDUSRMSG는 일괄처리 또는 대화식 OPM 프로그램이나 ILE 프로시듀어에서만 사용할 수 있습니다. 이 명령은 명령 행에서 입력될 수 없습니다. SNDMSG 명령은 정보용 메세지나 조회 메세지를 시스템 오퍼레이터 메세지 대기행렬(QSYSOPR), 표시장치 메세지 대기행렬 또는 사용자 메세지 대기행렬로 송신합니다. 정보용 메세지를 한번에 두 개 이상의 메세지 대기행렬로 송신할 수 있습니다. 그러나 조회 메세지는 한번에 하나의 메세지 대기행렬로만 송신할 수 있습니다. 메세지는 메세지 대기행렬에 지정된 전달 유형으로 전달됩니다. 메세지 대기행렬이 일시 중단 모드가 아니면 메세지가 사용자를 인터럽트시킬 수 없습니다.

다음의 SNDMSG 명령은 표시장치 사용자가 시스템 오퍼레이터에게 송신한 것입니다.

```
SNDMSG   MSG('Mount tape on device TAP1') TOUSR(*SYSOPR)
```

SNDBRKMSG 명령은 워크스테이션, 프로그램 또는 작업에서 리시버의 메세지 대기행렬이 어떤 전달 모드로 설정되든 그와 관계 없이 중단 모드에서 전달되는 하나 이상의 표시장치로 즉시 메세지를 전달합니다. 이 명령은 메세지를 표시장치 메세지 대기행렬에만 송신하는 데 사용할 수 있습니다. 표시장치 사용자의 즉각적인 조치가 필요한 메세지를 송신할 때는 반드시 SNDBRKMSG 명령을 사용해야 합니다. CHGJOB(작업 변경) 명령의 BRKMSG 매개변수를 사용하여 각 작업을 제어하기 때문에 그 메세지로 인해 일시 중단이 발생할 수도 있습니다.

조회 메세지를 송신하는 경우 사용자 표시장치의 메세지 대기행렬이 아닌 다른 메세지 대기행렬로 응답이 송신되도록 지정할 수 있습니다.

다음의 SNDBRKMSG 명령은 시스템 오퍼레이터가 모든 표시장치 메세지 대기행렬로 송신한 것입니다.

```
SNDBRKMSG MSG('System going down in 15 minutes')
          TOMSGQ(*ALLWS)
```

이 메세지를 송신할 때의 단점은 메세지가 송신될 때 활동 중이던 사용자에게만 송신되는 게 아니라 모든 사용자에게 송신된다는 점입니다.

CL 프로그램으로부터 메세지 송신

CL 프로시듀어나 프로그램으로부터 메세지를 송신하려면 SNDPGMMSG(프로그램 메세지 송신) 명령이나 SNDUSRMSG(사용자 메세지 송신) 명령을 사용하십시오.

SNDPGMMSG 명령으로는 다음과 같은 유형의 메세지를 송신할 수 있습니다.

- 정보
- 조회
- 완료
- 진단
- 요구
- 이탈
- 상태
- 통지

CL 프로시듀어나 프로그램의 메세지를 다음 유형의 대기행렬로 송신할 수 있습니다.

- 작업 리퀘스터의 외부 메세지 대기행렬(242 페이지의 『작업 메세지 대기행렬』 부분 참조)

- 작업이 호출한 프로그램 또는 프로시듀어의 호출 메세지 대기행렬(242 페이지의 『작업 메세지 대기행렬』 부분 참조)
- 시스템 오퍼레이터 메세지 대기행렬
- 워크스테이션 메세지 대기행렬
- 사용자 메세지 대기행렬

프로시듀어나 프로그램으로부터 메세지를 송신하기 위해 SNDPGMMSG 명령에서 다음을 지정할 수 있습니다.

- 메세지 ID 또는 즉시 메세지. 메세지 ID는 사전정의 메세지의 메세지 설명 이름입니다.
- 메세지 파일. 사전정의 메세지가 송신될 때 메세지 설명이 들어 있는 메세지 파일명.
- 메세지 자료 필드. 사전정의 메세지가 송신될 경우 이 필드에는 메세지의 대체 변수에 대한 값이 들어 있습니다. 각 필드의 형식은 메세지 설명에 서술되어 있어야 합니다. 즉시 메세지가 송신될 때에는 메세지 자료 필드가 없습니다.
- 메세지를 수신할 메세지 대기행렬 또는 사용자.
- 메세지 유형. 다음 표에는 대기행렬의 유형에 따라 송신될 수 있는 각 메세지의 유형이 표시되어 있습니다(V=유효).

표 8. 메세지 대기행렬 유형에 따른 유효한 메세지 유형

메세지 유형	메세지 대기행렬 유형				
	외부	호출	QSYSOPR	워크스테이션	사용자
정보	V	V	V	V	V
조회	V		V	V	V
완료	V	V	V	V	V
진단	V	V	V	V	V
요구	V	V			
이탈		V			
상태	V	V			
통지	V	V			

- 코드화 문자 세트 ID(CCSID). 제공된 메세지 또는 메세지 자료가 들어 있는 코드화 문자 세트 ID(CCSID)를 지정합니다.
- 응답 메세지 대기행렬. 조회 메세지에 대한 응답을 수신하는 메세지 대기행렬명. 디폴트로, 응답이 조회 메세지를 송신한 프로시듀어나 프로그램의 호출 메세지 대기행렬로 송신됩니다.
- 키 변수명. 메세지에 대한 메세지 참조 키가 들어 있는 CL 변수명.

230 페이지의 『메세지 설명의 예』에서 작성된 메세지를 송신하려면 다음 명령을 사용하십시오.

```
SNDPGMMSG   MSGID(USR4310) MSGF(QGPL/USRMSG) +
            MSGDTA(&CUSNO) TOPGMQ(*EXT) +
            MSGTYPE(*INFO)
```

메세지의 대체 변수는 고객 번호입니다. 고객 번호가 변하므로 사용자는 메세지상에 정확한 고객 번호를 지정할 수 없습니다. 대신, CL 프로시듀어나 프로그램에서 고객 번호의 CL 변수(&CUSNO)를 선언하십시오. 그런 다음 이 변수를 메세지 자료 필드로 지정하십시오. 메세지가 송신되면 변수의 현재 값이 메세지에 전달됩니다.

고객 번호 35500이 없음

그 외에도, 어떤 표시장치가 프로시듀어나 프로그램을 사용하는지를 항상 알고 있는 것은 아니므로 메세지가(TOPGMQ 매개변수) 송신되어야 하는 정확한 표시장치 메세지 대기행렬을 지정할 수 없습니다. 따라서 외부 메세지 대기행렬 *EXT를 지정하십시오.

메세지

조회 및 정보용 메세지

SNDUSRMSG 명령을 사용하면 조회 메세지나 정보용 메세지를 표시장치 사용자, 시스템 오퍼레이터 또는 사용자 정의 메세지 대기행렬로 송신할 수 있습니다.

SNDUSRMSG 명령을 사용하여 사용자에게 조회 메세지를 송신하는 경우 프로시듀어나 프로그램이 사용자로부터의 응답을 기다립니다. 메세지는 즉시 메세지이거나 사전정의 메세지일 수 있습니다. 대화식 작업에서는 메세지가 디폴트 표시장치 오퍼레이터에게 송신되고, 일괄처리 작업에서는 메세지가 디폴트 시스템 오퍼레이터에게 송신됩니다.

SNDUSRMSG 명령을 사용하여 프로시듀어나 프로그램으로부터 메세지를 송신하려면 SNDUSRMSG 명령에서 다음을 지정할 수 있습니다.

- 메세지 ID 또는 즉시 메세지. 메세지 ID는 사전정의 메세지의 메세지 설명 이름입니다.
- 메세지 파일. 사전정의 메세지가 송신될 때 메세지 설명이 들어 있는 메세지 파일명.
- 메세지 자료 필드. 사전정의 메세지가 송신될 경우 이 필드에는 메세지의 대체 변수에 대한 값이 들어 있습니다. 각 필드의 형식은 메세지 설명에 서술되어 있어야 합니다. 즉시 메세지가 송신될 때에는 메세지 자료 필드가 없습니다.
- 조회 메세지에 대해 유효한 응답.
- 조회 메세지에 대한 디폴트 응답 값.
- 메세지 유형.
- 메세지가 송신될 메세지 대기행렬.
- 메세지 응답. 조회 메세지에 대한 응답이 들어갈 CL 변수가 있는 경우.
- 변환 표. 응답 값을 변환하는 데 사용될 변환 표가 있는 경우. 이것은 일반적으로 소문자를 대문자로 변환시키는 데 사용됩니다.
- 코드화 문자 세트 ID(CCSID). 제공된 메세지 또는 메세지 자료가 들어 있는 코드화 문자 세트 ID(CCSID)를 지정합니다.

완료 및 진단 메세지

SNDPGMMSG 명령을 사용하면 진단 메세지와 완료 메세지를 송신할 수 있습니다. 사용자의 CL 프로시듀어나 프로그램으로부터 모든 메세지 대기행렬로 이들 메세지 유형을 송신할 수 있습니다. 진단 메세지는 CL 프로시듀어나 프로그램이 검출한 오류에 관한 호출을 프로그램이나 프로시듀어에 알려줍니다. 완료 메세지는 CL 프로시듀어나 프로그램이 수행한 작업 결과를 알려줍니다.

보통 이탈 메세지는 호출 프로그램이나 프로시듀어의 메세지 대기행렬로 송신되어 문제가 무었이었는지 또는 진단 메세지도 송신되었는지를 호출자에게 알려줍니다. 완료 메세지의 경우에는 요구된 기능이 수행되었기 때문에 일반적으로 이탈 메세지가 송신되지 않습니다.

완료 메세지를 송신하는 예로서, 시스템 오퍼레이터가 시스템 오퍼레이터 메뉴를 사용하여 어떤 오브젝트를 저장하기 위해 CL 프로그램 SAVPAY를 호출하는 것으로 가정하십시오. CL 프로그램에는 오브젝트를 저장한 후 다음과 같은 완료 메세지를 발행하는 다음의 프로시듀어만 포함되어 있습니다.

```
   PGM
SAVOBJ OBJ(PAY1 PAY2) LIB(PAYROLL) CLEAR(*YES)
SNDPGMMSG MSG('Payroll objects have been saved') MSGTYPE(*COMP)
    ENDPGM
```

SAVOBJ 명령이 실패하면 CL 프로시듀어 기능이 검사하고 시스템 오퍼레이터가 상세 메세지를 표시하여 이 장의 뒷부분에서 설명된 대로 실패 이유를 설명하는 특정 이탈 메세지를 찾습니다. SAVOBJ 명령이 정상적으로 완료되면 시스템 오퍼레이터 메뉴를 표시하는 프로그램과 연관된 호출 메세지 대기행렬로 완료 메세지가 송신됩니다.

완료 메세지의 장점 중 하나는 IBM 제공 명령과 일치한다는 점입니다. 많은 IBM 명령들은 성공적인 완료를 나타내는 완료 메세지를 송신합니다. 작업 기록부로 송신된 메세지의 유형을 보면 문제 분석에 도움이 될 수 있습니다.

상태 메세지

SNDPGMMSG 명령을 사용하여 CL 프로시듀어나 프로그램으로부터 작업에 대한 호출 메세지 대기행렬이나 외부 메세지 대기행렬(*EXT)로 상태 메세지를 송신할 수 있습니다. 상태 메세지가 호출 메세지 대기행렬로 송신되면 수신 프로그램 또는 프로시듀어가 상태 메세지의 도착 상황을 모니터하여 그 메세지에 설명된 상태를 처리할 수 있습니다. 수신 프로그램 또는 프로시듀어가 메세지를 모니터하지 못하는 경우에는 제어가 송신자에게로 리턴되어 처리가 재개됩니다. 280 페이지의 『CL 프로그램이나 프로시듀어에서의 메세지 모니터』를 참조하십시오.

이탈 및 통지 메세지

SNDPGMMSG 명령을 사용하여 사용자의 CL 프로시듀어나 프로그램의 이탈 메세지를 호출 프로그램이나 프로시듀어의 호출 메세지 대기행렬로 송신할 수 있습니다. 이탈 메세지는 호출자에게 프로시듀어나 프로그램이 비정상 종료되었다는 것과 그 이유를 알

려줍니다. 호출자는 이탈 메세지의 도착 상황을 모니터하여 그 메세지에 설명된 상태를 처리할 수 있습니다. 호출자가 상태를 처리할 때는 제어가 이탈 메세지의 송신자에게 리턴되지 않습니다.

호출자가 동일한 프로그램 안의 다른 프로시듀어인 경우 프로그램 자체는 종료되지 않습니다. 이탈 메세지가 송신된 프로시듀어는 계속될 수 있습니다. 이탈 메세지가 프로그램 자체 호출자에게 송신되면 프로그램 안의 모든 활동 프로시듀어가 즉시 종료됩니다. 결과적으로, 프로그램이 계속 실행될 수 없습니다. 호출자가 이탈 메세지를 모니터하지 않으면 디폴트 시스템 조치가 수행됩니다.

통지 메세지를 CL 프로시듀어나 프로그램으로부터 호출 프로그램이나 프로시듀어의 메세지 대기행렬 또는 외부 메세지 대기행렬로 송신할 수 있습니다. 통지 메세지는 호출자에게 처리를 계속할 수 있는 상태에 대해 알려줍니다. 호출 프로그램이나 프로시듀어가 통지 메세지의 도착 상황을 모니터하여 그 메세지에 설명된 상태를 처리할 수 있습니다. 호출자가 통합 언어 환경 프로시듀어인 경우 다음 기능을 수행할 수 있습니다.

- 조건을 처리할 수 있습니다.
- 호출자에게 거꾸로 응답을 송신할 수 있습니다.
- 송신 프로시듀어가 계속해서 처리하도록 허용할 수 있습니다.

호출자가 OPM 프로그램이고 메세지를 모니터하고 있지 않으면 송신자가 디폴트 응답을 수신합니다. 호출자가 ILE 프로시듀어이면, 메세지가 제어 경계로 여과됩니다. 모니터를 찾지 못한 경우에는 시스템이 송신자에게 디폴트 응답을 리턴합니다. 그러면 송신자가 처리를 재개합니다. 280 페이지의 『CL 프로그램이나 프로시듀어에서의 메세지 모니터』를 참조하십시오.

즉시 메세지는 이탈 메세지나 통지 메세지로 허용될 수 없습니다. 시스템은 메세지 CPF9898을 어플리케이션 프로그램에서 즉시 이탈 및 통지 메세지로 사용될 수 있도록 정의하였습니다. 예를 들어, 다음과 같은 경우가 있습니다.

```
SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) MSGDTA('Error condition') +
            MSGTYPE(*ESCAPE)
```

메세지 송신의 예

예 1: 다음 CL 프로시듀어를 사용하면 표시장치 사용자가 SBMJOB(작업 제출) 명령을 입력하는 대신이 프로시듀어를 포함한 CL 프로그램을 호출함으로써 작업을 제출할 수 있습니다. 작업이 제출된 경우 프로시듀어가 완료 메세지를 송신합니다.

```
   PGM
SBMJOB JOB(WKLYPAY) JOBD(USERA) RQSDTA('CALL WKLY PARM(PAY1)')
SNDPGMMSG MSG('WKLYPAY job submitted') MSGTYPE(*COMP)
    ENDPGM
```

예 2: 다음 CL 프로시듀어는 이 프로시듀어 안에서 호출된 프로그램으로부터 수신한 매개변수에 기초하여 메세지를 변경합니다. 그러면 CL 프로시듀어가 메세지를 완료 메세지로서 송신합니다. (RCDCNT 필드는 PGMA에서 문자로 정의됩니다.)

```
   PGM
DCL &RCDCNT TYPE(*CHAR) LEN(3)
CALL PGMA PARM(&RCDCNT)
SNDPGMMSG MSG('PGMA completed' *BCAT &RCDCNT *BCAT +
          'records processed') MSGTYPE(*COMP)
    ENDPGM
```

예 3: 다음 프로시듀어는 특별한 양식을 로드하도록 시스템 오퍼레이터에게 요구하는 메세지를 송신합니다. RCVMSG(메세지 수신) 명령은 응답을 기다립니다. 시스템 오퍼레이터는 조회 메세지에 대한 응답으로 적어도 하나의 문자를 입력해야 하지만, 프로시듀어는 응답 값을 사용하지 않습니다.

```
   PGM
DCL &MSGKEY TYPE(*CHAR) LEN(4)
SNDPGMMSG   MSG('Load special form') TOUSR(*SYSOPR) +
            KEYVAR(&MSGKEY) MSGTYPE(*INQ)
RCVMSG MSGTYPE(*RPY) MSGKEY(&MSGKEY) WAIT(120)
.
.
.
    ENDPGM
```

WAIT 매개변수를 RCVMSG 명령에 지정하여 프로시듀어가 응답을 기다리도록 해야 합니다. WAIT 매개변수가 지정되지 않으면 응답을 수신하지 않고 RCVMSG 명령 다음의 명령어(instruction)로 프로시듀어가 계속됩니다. 프로시듀어가 특정 메세지에 대한 응답을 수신할 수 있도록 MSGKEY 매개변수가 RCVMSG 명령에 사용됩니다. SNDPGMMSG 명령의 변수 &MSGKEY는 RCVMSG 명령에 사용되도록 프로시듀어로 리턴됩니다.

예 4: 다음 프로시듀어가 일괄처리 모드에서 수행될 경우에는 시스템 오퍼레이터에게 메세지가 송신되고, 표시장치에서 수행될 경우에는 표시장치 오퍼레이터에게 메세지가 송신됩니다. 프로시듀어는 Y 또는 N의 대소문자를 모두 허용합니다. (프로그램 논리를 보다 쉽게하려면 변환 표(TRNTBL 매개변수)를 사용하여 소문자는 대문자로 변환됩니다.) 입력된 값이 네 가지 값 중 하나가 아니면, 응답이 유효하지 않음을 알리는 메세지가 오퍼레이터에게 발행됩니다.

```
   PGM
DCL &REPLY *CHAR LEN(1)
.
.
SNDUSRMSG MSG('Update YTD Information Y or N') VALUES(Y N) +
     MSGRPY(&REPLY)
IF (&REPLY *EQ Y)
                  DO
 .
 .
 .
```

```
 ENDDO
ELSE
                  DO
 .
 .
 ENDDO
 .
 .
 .
    ENDPGM
```

예 5: 다음 프로시듀어는 메세지 CPF9898을 사용하여 이탈 메세지를 송신합니다. 메세지의 텍스트는 'Procedure detected failure'입니다. 즉시 메세지를 이탈 메세지로 사용할 수는 없으므로 이 메세지와 함께 메세지 CPF9898이 메세지 자료로서 사용될 수 있습니다.

```
      PGM
   .
   .
   .
SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)
     MSGDTA('Procedure detected failure')
   .
   .
    ENDPGM
```

예 6: 다음 프로시듀어는 시스템 오퍼레이터가 메세지를 여러 표시장치로 송신할 수 있도록 합니다. 시스템 오퍼레이터가 프로그램을 호출하면 호출된 프로그램 내에 들어 있는 이 프로시듀어는 시스템 오퍼레이터가 송신될 메세지 유형과 메세지에 대한 텍스트를 입력할 수 있는 프롬트를 표시합니다. 이 프로시듀어는 메세지의 날짜, 시간 및 텍스트를 연결합니다.

```
           PGM
          DCLF WSMSGD
          DCL &MSG TYPE(*CHAR) LEN(150)
          DCL &HOUR TYPE(*CHAR) LEN(2)
          DCL &MINUTE TYPE(*CHAR) LEN(2)
          DCL &MONTH TYPE(*CHAR) LEN(2)
          DCL &DAY TYPE(*CHAR) LEN(2)
          DCL &WORKHR TYPE(*DEC) LEN(2 0)
          SNDRCVF RCDFMT(PROMPT)
          IF &IN91 RETURN /* Request was ended  */
          RTVSYSVAL QMONTH RTNVAR(&MONTH)
          RTVSYSVAL QDAY RTNVAR(&DAY)
          RTVSYSVAL QHOUR RTNVAR(&HOUR)
          IF (&HOUR *GT '12') DO
          CHGVAR &WORKHR &HOUR
          CHGVAR &WORKHR (&WORKHR - 12)
          CHGVAR &HOUR &WORKHR /* Change from military time */
 ENDDO
          RTVSYSVAL QMINUTE RTNVAR(&MINUTE)
          CHGVAR    &MSG ('From Sys Opr ' *CAT &MONTH *CAT '/' +
                    *CAT &DAY +
                    *BCAT &HOUR *CAT ':' *CAT &MINUTE +
                    *BCAT &TEXT)
```

```
            IF (&TYPE *EQ 'B') GOTO BREAK
NORMAL:     SNDPGMMSG MSG(&MSG) TOMSGQ(WS1 WS2 WS3)
            GOTO ENDMSG
BREAK:      SNDBRKMSG MSG(&MSG) TOMSGQ(WS1 WS2 WS3)
ENDMSG:     SNDPGMMSG MSG('Message sent to display stations') +
                      MSGTYPE(*COMP)
    ENDPGM
```

다음은 이 프로그램에서 사용된 화면 파일 WSMSGD의 DDS입니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                      DSPSIZ(24 80)
     A          R PROMPT                    TEXT('Prompt')
     A                                      BLINK
     A                                      CA03(91 'Return')
     A                                   1  2'Send Messages To Workstations'
     A                                       DSPATR(HI)
     A                                   3  2'TYPE'
     A            TYPE           1   1    +2VALUES('N' 'B')
     A                                      CHECK(ME)
     A                                      DSPATR(MDT)
     A                                     +3'(N = No breaks  B = Break)'
     A                                   5  2'Text'
     A            TEXT         100   1    +2LOWER
     A
     A
```

시스템 오퍼레이터가 프롬트에 다음을 입력하면

```
B

Please sign off by 3:30 today
```

다음과 같은 일시 중단 메세지가 송신됩니다.

```
From Sys Opr 10/30 02:00  Please sign off by 3:30 today
```

SNDPGMMSG상의 호출 스택 항목 식별

CL 프로시듀어가 OPM 프로그램이나 다른 ILE 프로시듀어로 메세지를 송신할 경우에는 메세지가 송신되는 호출 스택 항목을 식별해야 합니다. 식별된 호출 스택 항목의 호출 메세지 대기행렬로 메세지가 송신됩니다.

SNDPGMMSG 명령의 TOPGMQ 매개변수를 사용하여 메세지가 송신될 호출 스택 항목을 식별할 수 있습니다. 호출 스택 항목 ID는 다음 두 부분으로 구성됩니다.

- 기본 항목의 스펙

  스펙 TOPGMQ(*PRV *)는 기본 항목을 SNDPGMMSG 명령을 사용하는 프로시듀어가 실행 중인 항목으로 식별합니다. 오프셋은 그 기준 이전의 항목으로서 지정됩니다. 이 스펙은 명령을 사용 중인 프로시듀어의 호출자를 식별합니다.

- 기본 항목의 오프셋 스펙

  오프셋 스펙(TOPGMQ의 요소 1)은 사용자가 기본(*SAME)으로 메세지를 송신하는지 아니면 기본 호출자(*PRV)에게 메세지를 송신하는지를 식별합니다.

TOPGMQ의 요소 2인 기본 항목을 식별하는 방법을 이해하려면 ILE 프로그램이 실행 중인 호출 스택도 함께 이해해야 합니다. 두 개의 프로그램이 이것을 설명하는 데 사용됩니다. 프로그램 CLPGM1은 OPM CL 프로그램이고, 프로그램 CLPGM2는 ILE 프로그램입니다. 프로그램 CLPGM2가 ILE이므로 CLPROC1, CLPROC2, CLPROC3 및 CLPROC4와 같은 몇 개의 프로시듀어로 구성될 수 있습니다. 수행 시에는 다음 순서로 호출됩니다.

- CLPGM1이 가장 먼저 호출됩니다.
- CLPGM1이 CLPGM2를 호출합니다.
- CLPGM2가 CLPROC1을 호출합니다.
- CLPROC1이 CLPROC2를 호출합니다.
- CLPROC2가 CLPROC3이나 CLPROC4를 호출합니다.

CLPROC2가 CLPROC4를 호출할 때의 호출 스택 구조를 이해하려면 258 페이지의 그림 4를 보십시오. 이 그림은 다음 고려사항을 설명합니다.

- 호출 스택 항목과 OPM 프로그램은 1 대 1로 대응됩니다. OPM 프로그램이 호출될 때마다, 하나의 새로운 항목이 호출 스택에 추가됩니다.
- ILE 프로그램은 하나의 단위로 스택에서 표시되지 않습니다. 대신에 ILE 프로그램이 호출될 때 프로그램에서 호출된 각 프로시듀어에 대한 스택에 하나의 항목이 추가됩니다. 결과적으로, 사용자는 ILE 프로그램이 아니라 ILE 프로시듀어로 메세지를 송신합니다.

주: ILE 프로그램이 호출될 때 실행될 첫 번째 프로시듀어는 프로그램에 대한 프로그램 입력 프로시듀어(PEP)입니다. CL에서는 이 프로시듀어(_CL_PEP)가 시스템에 의해 생성되어 사용자가 제공한 첫 번째 프로시듀어를 호출합니다. 이 예에서 PEP에 대한 항목은 OPM 프로그램 CLPGM1에 대한 항목과 프로시듀어 CLPROC1에 대한 항목 사이에 있습니다.

다음은 기본 호출 스택 항목을 지정하는 또 다른 방법입니다.

명령을 기본으로 사용하는 프로시듀어

TOPGMQ 매개변수가 TOPGMQ(*SAME *) 또는 TOPGMQ(*PRV *)를 지정하면 SNDPGMMSG 명령을 사용하는 프로시듀어에 대한 항목이 기본으로 사용됩니다. TOPGMQ(*SAME *)가 지정되면 프로시듀어가 그 자신에게 메세지를 송신합니다. TOPGMQ(*PRV *)가 지정되면 프로그램이 호출자에게 메세지를 송신합니다.

주: 프로시듀어가 TOPGMQ(*PRV *)를 지정하여 호출자에게 메세지를 송신할 때 사용자는 다음 정보를 반드시 알아야 합니다.

- CLPROC4와 CLPROC2가 호출자에게 메세지를 되돌려 송신하면 메세지를 포함하는 프로그램에 메세지가 남지 않습니다. 메세지는 동일한 프로그램 안에 있

는 프로시듀어 사이에서 송신됩니다. 오브젝트가 프로그램 호출자(이 예에서 CLPGM1)에게로 메세지를 송신해야 하는 경우 TOPGMQ(*PRV *) 지정은 적절한 선택이 아닙니다.

- CLPROC1이 호출자에게로 메세지를 되돌려 송신하는 경우 프로그램 입력 프로시듀어를 건너뜁니다. 호출자가 PEP인 경우에는 메세지는 CLPGM1로 송신됩니다. TOPGMQ(*PRV *)가 지정되면 PEP 항목을 *볼 수 없으며* 송신 조작에 포함시킬 수도 없습니다. TOPGMQ가 다른 방법으로 지정되면 송신자가 PEP를 *볼 수 있습니다.*

259 페이지의 그림 5는 CLPROC1, CLPROC2 및 CLPROC4가 각 메세지를 각 프로시듀어의 호출자에게로 돌려 보낼 때의 결과를 설명합니다.



그림 4. 수행 시 호출 스택의 예



*그림 5. TOPGMQ(*PRV *)의 예*

기본 항목을 이름으로 식별

해당 항목에서 실행 중인 OPM 프로그램이나 ILE 프로시듀어의 이름을 제공하여 기본 항목을 식별할 수 있습니다. 제공된 이름은 단순명(한 부분) 아니면 복합명(둘 또는 세 부분)입니다. 다음은 단순명 및 복합명에 대한 설명입니다.

- 단순명

  단순명은 OPM 프로그램이나 ILE 프로시듀어를 식별하는 데 사용됩니다. 사용자가 제공하는 단순명의 길이가 10자 이하인 경우에는 이름이 OPM 프로그램인지 아니

면 ILE 프로시듀어인지를 시스템이 판별합니다. 해당 이름을 사용하여 기본을 가장 최근에 호출된 OPM 프로그램이나 ILE 프로시듀어로 식별할 수 있습니다.

이름의 길이가 10자보다 긴 경우 ILE 프로시듀어에 대한 이름으로 시스템이 판별합니다. (OPM 프로그램명은 10자 이하입니다.) 해당 이름을 사용하여 기본을 가장 최근에 호출된 프로시듀어에 대한 입력으로 식별합니다. OPM 프로그램을 실행 중인 항목은 고려되지 않습니다.

단순명을 사용하여 메세지를 송신하는 예에 대해서는 261 페이지의 그림 6을 보십시오. 이 예에서는 CLPROC4가 메세지를 CLPROC2로 송신하고 CLPROC2는 메세지를 CLPGM1로 송신합니다.

- 복합명

  복합명은 둘 또는 세 부분으로 구성됩니다. 복합명은 다음과 같습니다.

  – 모듈명

    모듈명은 프로시듀어가 컴파일되었던 모듈의 이름입니다.

  – 프로그램명

    프로그램명은 프로시듀어가 바인드된 프로그램의 이름입니다.

  – 프로시듀어명

  메세지를 송신하려는 프로시듀어를 고유하게 식별하고자 할 때는 다음 조합 중 하나에 복합명을 사용할 수 있습니다.

  – 프로시듀어명, 모듈명, 프로그램명

  – 프로시듀어명과 모듈명

  – 프로시듀어명과 프로그램명

    모듈명은 반드시 *NONE으로 지정해야 합니다.

  복합명을 사용하면 식별되고 있는 기본이 OPM 프로그램을 실행할 수 없습니다.

복합명을 사용하여 메세지를 송신하는 예에 대해서는 262 페이지의 그림 7을 보십시오. 이 예에서는 CLPROC4가 (프로시듀어명, 프로그램명)으로 구성된 두 부분의 이름을 사용하여 CLPROC1로 메세지를 송신중입니다.

전체 OPM 프로그램명이나 전체 ILE 프로시듀어명을 사용하기보다는 부분명을 사용할 수 있습니다. IBM에서는 부분 호출 스택 항목 이름을 지정하는 것과 관련된 온라인 정보를 제공합니다. 이 정보에 대해서는 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.


CALL PGM(CLPGM1)
OPM PROGRAM
CLPGM1
CALL PGM(CLPGM2)
PROCEDURE
_CL_PEP
(in pgm CLPGM2)
CALLPRC CLPROC1
PROCEDURE
CLPROC1
(in pgm CLPGM2)
CALLPRC CLPROC2
PROCEDURE
CLPROC2
(in pgm CLPGM2)
SNDPGMMSG
TOPGMQ(*SAME
CLPGM1)...
CALLPRC CLPROC4
PROCEDURE
CLPROC4
(in pgm CLPGM2)
SNDPGMMSG TOPGMQ
(*SAME CLPROC2)
RBAFN565-0

그림 6. 단순명 사용의 예



그림 7. 복합명(complex name) 사용의 예

프로그램 경계를 기본으로 사용

특수 값 *PGMBDY가 자체 사용되거나 프로그램명과 함께 사용되어 CL 프로그램의 PEP를 식별합니다. 그러면 식별된 CL 프로그램의 PEP에 대한 항목은 기본 항목이 됩니다. 이 옵션은 프로시듀어가 들어 있는 프로그램의 경계 외부에 있는 CL 프로시듀어 안에서 메세지를 송신하려는 경우에 유용합니다.

특수 값 *PGMBDY를 사용한 메세지 송신의 예에 대해서는 264 페이지의 그림 8을 참조하십시오. 이 예에서는 CLPROC4가 포함하는 프로그램 CLPGM2의 호출자인

CLPGM1로 메세지를 직접 송신하고 있습니다. CLPROC4는 어떤 프로그램이 CLPGM2를 호출하는지 또는 메세지를 송신하는 프로시듀어와 비교된 PEP의 위치를 모르고도 이것을 수행할 수 있습니다. 이 예에서는 동반하는 프로그램명이 지정되지 않고 *PGMBDY가 사용됩니다. 이는 경계가 식별되어야 하는 프로그램이 메세지를 송신중인 프로시듀어가 들어 있는 프로그램임을 의미합니다.

특수 값 *PGMBDY와 프로그램명을 사용한 메세지 송신의 예에 대해서는 265 페이지의 그림 9를 보십시오. 다음 프로그램과 프로시듀어가 265 페이지의 그림 9에 사용됩니다.

- CLPGM1과 CLPGM2. 이것은 앞의 예에서처럼 정의됩니다.
- CLPGM3. 이는 다른 ILE 프로그램입니다.
- CLPGM3의 CLPROCA. 메세지가 CLPROCA에서 CLPGM2의 호출자로 송신됩니다.

특수 값 *PGMBDY와 프로그램명 CLPGM2를 사용하여 메세지가 CLPROCA에서 CLPGM2의 호출자로 송신됩니다.

이 예에서는 TOPGMQ 매개변수가 TOPGMQ(*PRV_CL_PEP)로 지정되면 메세지가 CLPGM2의 호출자보다는 CLPGM3의 호출자에게 송신됩니다. 이것은 해당 이름으로 가장 최근에 호출된 프로시듀어가 CLPGM3에 대한 PEP이기 때문에 발생합니다.



*그림 8. *PGMBDY 사용 예 1*



*그림 9. *PGMBDY 사용 예 2*

또한 특수 값 *PGMBDY가 OPM 프로그램과 함께 사용됩니다. *PGMBDY와 함께 OPM 프로그램명을 지정하면 OPM 프로그램명만 사용되었을 때와 결과가 동일합니다.

예를 들면, TOPGMQ(*SAME *PGMBDY *NONE opmname)는 TOPGMQ(*SAME opmname)와 동일한 위치로 메세지를 송신합니다.

이에 대한 예외는 순환하여 스스로를 호출한 OPM 프로그램으로 메세지가 송신될 때입니다. TOPGMQ(*SAME pgmname)는 메세지를 가장 최근의 순환 레벨로 송신합니다. 그러나 TOPGMQ(*SAME *PGMBDY *NONE pgmname)는 메세지를 첫 번째 순환 레벨로 송신합니다. 267 페이지의 그림 10은 PGM1이 호출되어 자기 자신을 두 번 더 순환적으로 호출하는 방법을 보여줍니다. 세 번째 순환 레벨에서 PGM1이 PGM2를 호출합니다. 그러면 PGM2가 메세지를 PGM1로 되돌려 송신합니다. 프로그램이 이름 PGM1만 사용하여 송신되면 메세지가 PGM1의 세 번째 순환 레벨로 갑니다. 프로그램이 특수 값 *PGMBDY와 함께 이름 PGM1을 사용하여 송신되면 메세지가 PGM1의 첫 번째 순환 레벨로 보내집니다.



*그림 10. *PGMBDY 사용 예 3*

가장 최근에 호출된 프로시듀어를 기본으로 사용

프로시듀어의 이름을 알지 못해도 ILE 프로그램의 가장 최근에 호출된 프로시듀어로 메세지를 거꾸로 송신하려는 경우가 있습니다. 이 경우 특수 값 *PGMNAME이 ILE 프로그램명과 함께 사용되어 기본 항목명을 식별된 프로그램의 가장 최근에 호출된 프로시듀어의 이름으로서 사용할 수 있습니다. 이 예에서 사용되는 세 개의 프로그램은 다음과 같습니다.

- CLPGM1은 프로시듀어 PROCA와 PROCB를 가진 ILE 프로그램입니다.
- CLPGM2와 CLPGM3은 둘 다 OPM 프로그램입니다.
- CLPGM3은 CLPGM1로 메세지를 송신해야 하지만, 어느 프로시듀어가 가장 최근에 호출된 것인지 모릅니다.

송신은 특수 값 *PGMNAME과 프로그램명 CLPGM1을 사용하여 이루어집니다.

269 페이지의 그림 11을 참조하십시오. 메세지를 사용하여 송신하는 방법의 예에 대해서는 269 페이지의 그림 11을 보십시오.

사용자가 모든 CL 프로그램이 아닌 일부 CL 프로그램만 ILE 프로그램으로 변환할 경우에는 특수 값 *PGMNAME이 유용합니다. 예를 들면, CLPGM1은 OPM CL 프로그램입니다. CLPGM3이 CLPGM1로 메세지를 송신하고 TOPGMQ(*SAME CLPGM1)를 지정합니다. CLPGM1이 ILE로 변환되면 CLPGM3(OPM) 안의 SNDPGMMSG 명령만 작동합니다. CLPGM1은 CLPGM1을 위한 호출 스택에 항목이 없으므로 작동하지 않습니다. 만약 사용자가 명령을 TOPGMQ(*SAME *PGMNAME *NONE CLPGM1)로 변경하면 CLPGM3은 프로시듀어명에 대하여 사용했던 이름에 관계 없이 CLPGM1에 성공적으로 메세지를 송신합니다.

특수 값 *PGMNAME은 OPM 프로그램명에서도 사용할 수 있습니다. 이 경우에도 효과는 사용자가 이름을 올바르게 사용한 것과 같습니다. 예를 들면, TOPGMQ(*SAME *PGMNAME *NONE opmpgm)는 메세지를 TOPGMQ(*SAME opmpgm)와 같은 장소로 송신합니다. 메세지가 OPM 프로그램명이나 ILE 프로그램명으로 송신되어야 하는지를 판별할 수 없을 때는 *PGMNAME 사용을 고려해야 합니다.

제어 경계를 기본으로 사용

특수 값 *CTLBDY를 사용하여 기본 항목을 가장 가까운 제어 경계에 있는 것으로 식별할 수 있습니다. 두 개의 항목이 두 개의 서로 다른 활성 그룹에서 실행 중인 경우 제어 경계는 두 개의 호출 스택 항목 사이에 존재합니다. 이 특수 값을 사용하여 식별된 항목은 메세지를 송신중인 항목과 동일한 활성 그룹에서 실행됩니다.

특수 값 *CTLBDY를 사용한 메세지 송신 예에 대해서는 270 페이지의 그림 12를 보십시오. 이 예에서는 세 개의 프로그램(CLPGM1, CLPGM2 및 CLPGM3)이 모두 ILE 프로그램입니다. CLPGM2와 CLPGM3이 활성 그룹 AG2에서 실행되는 동안 CLPGM1은 활성 그룹 AG1에서 실행됩니다. 이 예에서 PROC3A는 AG2에 대한 경계 바로 앞에 있는 항목으로 메세지를 되돌려 송신합니다.



그림 11. 수행 시 호출 스택의 예



*그림 12. *CTLBDY 사용 예*

서비스 프로그램의 고려사항

앞의 논의사항은 ILE 프로그램과 ILE 서비스 프로그램에 둘 다 적용됩니다. ILE 프로그램과 ILE 서비스 프로그램 간의 가장 중요한 차이점은 메세지 처리에 관련된 것입니다. 서비스 프로그램은 PEP를 갖고 있지 않습니다.

기본 항목을 식별하는 데 사용되는 옵션에는 PEP가 전혀 필요없습니다. 예외는 이름 _CL_PEP가 명시적으로 사용될 때입니다. 예를 들어, TOPGMQ(*PRV *PGMBDY)는 항상 ILE 프로그램이나 서비스 프로그램의 호출자에게 메세지를 송신합니다. 이것이 ILE 프로그램인 경우 *PGMBDY 값이 PEP를 기본으로서 식별합니다. 이것이 ILE 서비스 프로그램인 경우 *PGMBDY 값이 서비스 프로그램에서 호출된 첫 번째 프로시듀어에 대한 항목을 식별합니다.

CL 프로시듀어나 프로그램에서 메세지 수신

프로시듀어나 프로그램에 대한 메세지 대기행렬로부터 메세지를 수신하려면 RCVMSG(메세지 수신) 명령을 사용하십시오. 다음과 같은 방법으로 메세지를 수신할 수 있습니다.

- 메세지 유형에 의해. 모든 유형 또는 특정 유형만 수신되도록 지정할 수 있습니다(MSGTYPE 매개변수). 새 메세지(프로시듀어나 프로그램에서 아직 수신되지 않은)는 선입선출(FIFO)순으로 수신됩니다. 그러나 ESCAPE 유형의 메세지는 후입선출(LIFO)순으로 수신됩니다.
- 메세지 참조 키에 의해. 다음 중 하나를 수행할 수 있습니다.
  - 메세지 참조 키를 사용하여 메세지를 수신함. 시스템은 메세지 대기행렬에 있는 각 메세지에 대해 메세지 참조 키를 할당하며, 이 키는 인쇄 가능하지 않으므로 이 키를 변수 자료로 전달합니다. 사용자는 이 변수를 CL 프로시듀어나 프로그램에 반드시 선언해야 합니다. 또한 키를 전달할 CL 변수를 RCVMSG 명령에 반드시 지정해야 합니다(MSGKEY 매개변수).
  - 메세지 대기행렬상에 있는 메세지 중 지정된 메세지 참조 키를 가진 메세지의 그 다음 메세지를 수신함. MSGKEY 매개변수뿐 아니라 MSGTYPE(*NEXT)도 지정해야 합니다.
  - 메세지 대기행렬상에 있는 메세지 중 지정된 메세지 참조 키를 가진 메세지의 이전 메세지를 수신함. MSGKEY 매개변수뿐 아니라 MSGTYPE(*PRV)도 지정해야 합니다.
- 메세지 대기행렬에서의 위치에 의해. 메세지 대기행렬상의 첫 번째 메세지에 MSGTYPE(*FIRST)을 지정해야 합니다. 최종 메세지는 MSGTYPE(*LAST)을 지정하십시오.
- 메세지 유형과 메세지 참조 키 모두에 의해(MSGTYPE과 MSGKEY 매개변수).

메세지를 수신하기 위해서는 다음을 지정할 수 있습니다.

- 메세지 대기행렬. 수신될 메세지가 있는 대기행렬입니다.
- 메세지 유형. 특정한 메세지 유형 또는 모든 유형을 지정할 수 있습니다.

- 메세지 도착을 기다릴 것인지의 여부. 대기 시간이 종료된 후에도 메세지가 수신되지 않으면 리턴이 요구된 CL 변수는 공백(숫자인 경우 0)으로 채워지고 RCVMSG 명령을 실행 중인 프로시듀어나 프로그램으로 제어가 리턴됩니다.
- 메세지가 수신된 후 이 메세지를 메세지 대기행렬로부터 제거할 것인지의 여부. 메세지가 제거되지 않으면 이 메세지는 메세지 대기행렬상의 구 메세지가 되어 메세지 참조 키를 통해서만 (프로시듀어가) 다시 수신할 수 있습니다. 그러나 메세지 대기행렬상의 메세지가 CHGMSGQ 명령에 의해 새로운 메세지로 재설정되면 이 메세지를 수신하기 위해 메세지 참조 키를 사용하지 않아도 됩니다. 이미 응답한 조회 메세지는 새로운 상태로 재설정되지 않음에 유의하십시오. (보다 자세히 알려면 279 페이지의 『메세지 대기행렬로부터 메세지 제거』를 참조하십시오.)
- 변환될 CCSID. 메세지 텍스트를 리턴시키려는 CCSID를 지정합니다.
- 다음 정보가 들어갈 CL 변수의 그룹(각각의 정보는 하나의 변수와 대응됨)
  - 메세지 대기행렬에 있는 메세지의 메세지 참조 키(문자 변수, 4자)
  - 메세지(문자 변수, 가변 길이)
  - 대체 변수 자료의 길이를 포함한 메세지 길이(10진 변수, 소수 5자릿수)
  - 메세지 온라인 도움말 정보(문자 변수, 가변 길이)
  - 대체 변수 자료의 길이를 포함한 메세지 도움말의 길이(10진 변수, 소수 5자릿수)
  - 메세지의 송신자가 제공한 대체 변수용 메세지 자료(문자 변수, 가변 길이)
  - 메세지 자료의 길이(10진 변수, 소수 5자릿수)
  - 메세지 ID(문자 변수, 7자)
  - 심각도 코드(10진 변수, 길이 2)
  - 메세지 송신자(문자 변수는 최소한 80자여야 함)
  - 수신되는 메세지의 유형(문자 변수, 길이 2자)
  - 수신되는 메세지의 경고 옵션(문자 변수, 9자)
  - 사전정의 메세지가 들어 있는 메세지 파일(문자 변수, 10자)
  - 메세지를 수신할 때 사용되는 메세지 파일이 들어 있는 메세지 파일 라이브러리 명(문자 변수, 10자)
  - 메세지를 송신할 때 사용되는 메세지 파일이 들어 있는 메세지 파일 라이브러리 명(문자 변수, 10자)
  - 메세지 자료 CCSID(리턴된 대체 자료(10진 변수, 5자릿수)와 연관된 코드화 문자 세트 ID)
  - 텍스트 자료 CCSID(메세지 및 메세지 도움말 매개변수(10진 변수, 소수 5자릿수)가 리턴시킨 텍스트와 연관된 코드화 문자 세트 ID)

```
RCVMSG MSGQ(QGPL/INVN) MSGTYPE(*ANY) MSG(&MSG)
```

수신된 메세지는 변수 &MSG에 놓입니다. *ANY는 MSGTYPE 매개변수에 대한 디폴트 값입니다.

CL 이외의 언어로 작성된 ILE 프로시듀어의 호출 스택 항목 메세지 대기행렬에 대해 작업할 때 예외를 아직 처리하지 않았을 경우에는 예외 메세지(이탈 또는 통지)를 수신할 수 있습니다. RCVMSG 명령을 사용하여 메세지를 수신하고 시스템에 예외가 처리됨을 동시에 나타낼 수 있습니다.

RMV 키워드를 사용하여 이를 제어할 수 있습니다. 이 키워드에 *NO가 지정되면 예외가 처리되고, 메세지는 메세지 대기행렬에 구 메세지로 남습니다. *KEEPEXCP가 지정되는 경우 예외가 처리되지 않고 메세지는 새로운 메세지로서 메세지 대기행렬상에 보존됩니다. *YES가 지정되는 경우 예외 메세지가 처리되며 메세지가 메세지 대기행렬에서 제거됩니다.

RTNTYPE 키워드는 수신된 메세지가 예외 메세지인지 판별하기 위해 사용할 수 있습니다. 그 결과 예외 메세지인 경우에는 예외가 처리되었는지 여부를 판별하는 데도 사용됩니다.

요구 메세지

요구 메세지를 수신하는 것은 CL 프로시듀어나 프로그램이 CL 명령을 처리하는 방법입니다. 예를 들면, 사용자의 프로시듀어나 프로그램은 표시장치로부터 입력을 받아서 프로그램의 분석 및 처리 결과인 메세지를 처리할 수 있습니다. 일반적으로 요구 메세지는 작업의 외부 메세지 대기행렬(*EXT)로부터 수신됩니다. 일괄처리 작업인 경우 수신된 요구는 입력 스트림으로부터 읽혀진 것입니다. 대화식 작업인 경우 수신된 요구는 표시장치 사용자가 명령 입력 화면상에서 한번에 하나씩 입력한 것입니다. 예를 들면, CL 명령은 IBM 제공 CL 프로세서에 의해 수신되는 요구입니다.

사용자의 프로시듀어나 프로그램은 요구 메세지의 자료 구문의 정의, 요구 해석 및 오류 진단을 수행해야 합니다. 요구를 분석하고 있거나 요구 기능을 수행하고 있는 동안 오류가 검출될 수 있습니다. 이들 오류의 결과로, 메세지가 프로시듀어나 프로그램의 호출 메세지 대기행렬로 송신됩니다. 프로시듀어나 프로그램은 메세지를 처리하고 나서 다음 요구 메세지를 수신합니다. 따라서 요구 처리 주기가 정의됩니다. 요구 메세지가 수신되고 사용자의 프로시듀어나 프로그램이 요구를 분석하고 실행하여 결과적으로 메세지가 표시되고 다음 요구가 수신됩니다. 일괄처리 작업에 수신되어야 하는 요구 메세지가 더 이상 없으면, 사용자의 프로시듀어나 프로그램으로 이러한 상태를 나타내는 이탈 메세지가 송신됩니다.

한 작업에서 두 개 이상의 OPM 프로그램이나 ILE 프로시듀어가 처리할 요구 메세지를 수신할 수 있습니다. 이 경우 가장 최근의 프로그램 호출에 의해 수신된 요구는 보다 높은 레벨의 프로그램 호출에 의해 수신된 요구 안에 내포된 것으로 여겨집니다. 각 내포 수준에서의 요구 처리 주기는 서로 독립적입니다. ILE 프로그램 안에서 그 프로

그램에 있는 하나 이상의 프로시듀어가 요구 메세지를 수신할 수 있습니다. 두 개 이상의 프로시듀어가 요구를 처리하면 동일한 ILE 프로그램 안에서 내포가 발생하며 내포 레벨은 독립적으로 남습니다.

다음은 요구 메세지가 QCMD에 의해 어떻게 처리되는지를 나타내는 다이어그램입니다.



1 CL 프로세서 QCMD는 *EXT로부터 요구 메세지를 수신합니다.

2 *EXT에 요구 메세지가 없으면 '명령 입력' 화면이 표시됩니다. 표시장치 사용자는 이 화면에 명령을 입력합니다. 명령이 입력되면 요구 메세지로서 *EXT에 위치합니다.

3 명령이 QCMD 호출 메세지 대기행렬의 끝으로 이동하고 거기에서 QCMD로 전달됩니다.

4 명령이 분석되며 명령 처리 프로그램(CPP)이 호출됩니다.

5 명령 처리 프로그램이 진단 메세지를 QCMD의 호출 메세지 대기행렬로 송신합니다.

6 그러면 명령 처리 프로그램이 이탈 메세지를 QCMD의 호출 메세지 대기행렬로 송신합니다. 이 이탈 메세지는 QCMD에게 진단 메세지가 대기행렬에 있으며 QCMD가 CPP 처리를 종료해야 함을 통지합니다.

7 QCMD는 요구 검사(CPF9901) 또는 기능 검사(CPF9999) 이탈 메세지의 도착을 모니터합니다. QCMD는 다음 번 요구 메세지의 수신을 시도합니다. 요구 프로세서가 메세지 CPF9901 또는 CPF9999를 수신하면 RCLRSC(자원 재

생) 명령을 수행해야 합니다. 또한 요구 프로세서는 메세지 CPF1907(요구 종료)과 메세지 CPF2415(사용자가 '명령 입력' 화면에서 F3 키 또는 F12 키를 눌렀음을 표시)를 모니터해야 합니다.

8 요구 메세지가 처리 중이므로 QCMD에 대한 호출 메세지 대기행렬상의 모든 메세지가 '명령 입력' 화면에 기록됩니다. 그러면 이 화면이 표시장치 사용자에게 다른 명령을 프롬트합니다.

9 앞의 요구 메세지(명령) 및 그것과 연관된 메세지는 작업에 지정된 메세지 기록 레벨에 따라 작업 기록부에 포함됩니다. 더 자세한 내용은 319 페이지의 『메세지 기록』 부분을 참조하십시오.

요구 처리 프로시듀어와 프로그램 작성

CL 프로시듀어를 프로그램 안의 요구 프로세서로 지정하면 많은 장점이 있습니다. 다음 리스트는 세 가지 장점을 요약한 것입니다.

- 요구 메세지를 273 페이지의 『요구 메세지』에 설명된 대로 처리함.
- ENDRQS(요구 종료) 명령 사용이 가능함. 이 명령은 시스템 요구 메뉴로부터 사용하거나 작업 단절 기능의 일부로 사용할 수 있습니다.
- 메세지를 필터링할 수 있음.

요구 프로세서 프로시듀어나 프로그램이 되려면 사용자의 프로시듀어나 프로그램이 SNDPGMMSG(프로그램 메세지 송신) 및 RCVMSG(메세지 수신) 명령을 포함해야 합니다. 예를 들면, 다음 명령은 프로시듀어나 프로그램이 요구 프로세서가 될 수 있게 합니다.

```
    SNDPGMMSG  MSG('Request Message') TOPGMQ(*EXT) MSGTYPE(*RQS)
RCVMSG     PGMQ(*EXT) MSGTYPE(*RQS) RMV(*NO)
```

요구 메세지가 PGMQ *EXT로부터 수신됩니다. 요구 메세지가 수신되면 RCVMSG 명령을 지정한 프로시듀어나 프로그램의 호출 메세지 대기행렬로 이동합니다(실제로 제거 및 재전송됨). 따라서 메세지를 제거할 때는 올바른 호출 메세지 대기행렬을 사용하여야 합니다.

메세지 참조 키(MRK)를 사용하여 요구 메세지가 제거되면 SNDPGMMSG 명령이 아닌 RCVMSG 명령의 KEYVAR 키워드로부터 MRK를 확보해야 합니다. (MRK는 요구 메세지의 수신 시기를 변경합니다.) 요구 메세지가 호출 메세지 대기행렬로부터 제거되면 프로시듀어나 프로그램이 요구 프로세서가 될 수 없으므로 RCVMSG 명령에 RMV(*NO)를 반드시 지정해야 합니다.

요구 메세지가 수신될 때 프로시듀어나 프로그램은 요구 프로세서로 식별됩니다. 프로시듀어나 프로그램이 요구 프로세서인 동안에는 호출된 다른 프로시듀어나 프로그램이 시스템 요구 메뉴상의 옵션 2(종료 요구)를 사용하여 종료될 수 있습니다. 요구 프로세서 프로시듀어나 프로그램은 메세지 CPF1907(MONMSG 명령)에 대한 모니터를 포함

해야 합니다. (시스템 요구 메뉴상의 옵션 2 또는 종료 요구 명령의) 종료 요구 기능이 요구 프로세서로 메세지를 송신하기 때문에 이는 반드시 필요한 사항입니다.

프로시듀어가 종료하거나(이상 또는 정상적으로) RMVMSG 명령이 수행될 때까지 프로시듀어나 프로그램이 요구 프로세서에 남아서 요구 프로세서의 호출 메세지 대기행렬로부터 모든 요구 메세지를 제거합니다. 예를 들면, 다음은 메세지 대기행렬에서 모든 요구 메세지를 제거하여 요구 처리를 종료시키는 명령입니다.

```
RMVMSG CLEAR(*ALL)
```

QCAPCMD API를 호출하고, OS/400 명령 분석기가 OS/400 명령에 대한 요구 메세지를 처리하도록 메세지 검색 키를 지정하십시오. 요구 메세지를 수신할 때 메세지 검색 키를 얻을 수 있습니다. QCAPCMD(명령 프로세스)는 작업 기록부에 있는 요구 메세지를 갱신하고 제공되는 새로운 값을 추가합니다. QCAPCMD는 또한 작업 기록부 안에 감추어진 암호와 같은 매개변수 값을 숨깁니다. 시스템은 다음 두 가지 조건 중 하나가 존재하면 작업 기록부에 있는 요구 메세지를 갱신하지 않습니다.

- 실행 명령(QCMDEXC 또는 QCAEXEC) API를 사용하는 경우
- 메세지 검색 키를 QCAPCMD에 제공하지 못하는 경우

요구 프로세서의 존재 여부 판별

작업이 요구 프로세서를 가지고 있는지 판별하려면 작업의 호출 스택을 표시하십시오. DSPJOB(작업 표시) 또는 WRKJOB(작업에 대한 작업) 명령에서 옵션 11을 사용하거나 WRKACTJOB 화면에 나열된 작업에 대해 옵션 10을 선택하십시오. 작업의 호출 스택에 있는 화면의 요구 레벨 열에 숫자가 표시되어 있으면, 그 숫자와 연관된 프로그램이나 ILE 프로시듀어가 요구 프로세서입니다. 다음 예에서는 QCMD와 QTEVIREF 둘 다 요구 프로세서입니다.

```
                          호출 스택 표시
                                                        시스템:   S0000000
작업:  WS31          사용자: QSECOFR       번호:     000173

옵션을 입력한 후 Enter 키를 누르십시오.
  5=상세 표시

     요구     프로그램 또는
Opt  레벨     프로시듀어     라이브러리     명령문        명령어
              QCMD           QSYS                       01DC
       1      QCMD           QSYS                       016B
              QTECADTR       QSYS                       0001
       2      QTEVIREF       QSYS                       02BA

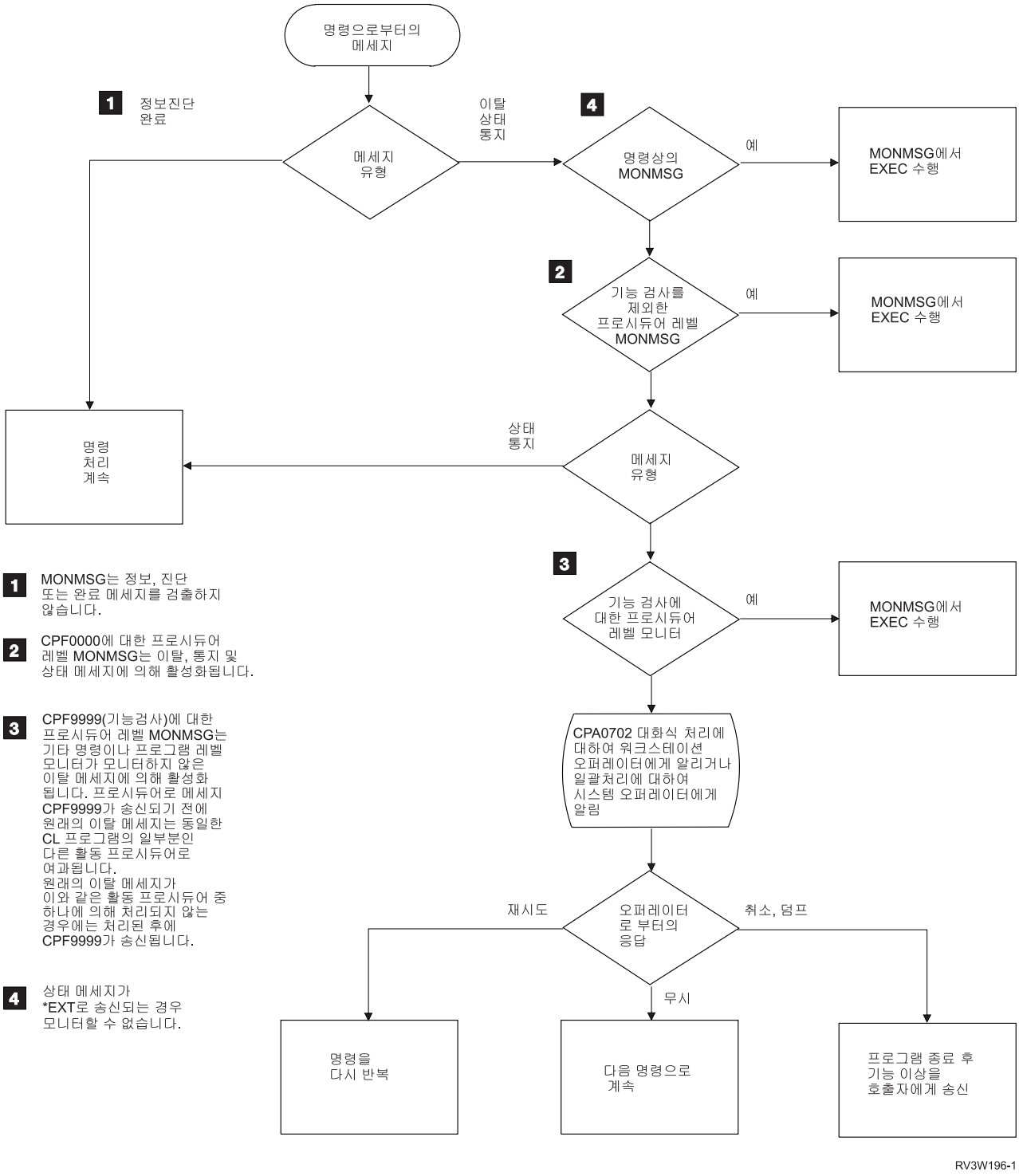








                                                                맨 아래

F3=나감      F10=스택 갱신      F11=활성 그룹 표시      F12=취소
F17=맨 위    F18=맨 아래
```

다음은 요구 처리 프로시듀어의 예입니다.

```
   PGM
    SNDPGMMSG  MSG('Request Message') TOPGMQ(*EXT) MSGTYPE(*RQS)
    RCVMSG     PGMQ(*EXT) MSGTYPE(*RQS) RMV(*NO)
      .
      .
      .
    CALL   PGM(PGMONE)
    MONMSG MSGID(CPF1907)
      .
      .
      .
RMVMSG CLEAR(*ALL)
    CALL   PGM(PGMTWO)
      .
      .
      .
         ENDPGM
```

이 프로시듀어의 첫 번째 두 명령이 이를 요구 프로세서로 만듭니다. 프로시듀어는 RMVMSG 명령이 실행될 때까지는 요구 프로세서로 남아 있습니다. PGMONE로부터 요구 프로세서로 종료 요구가 송신될 수 있기 때문에 프로그램 PGMONE의 호출 다음에 메세지 모니터 명령이 놓입니다. 모니터링이 사용되지 않는 경우에는 종료 요구에 대한 기능 검사가 발생하게 됩니다. RMVMSG 명령이 요구 처리를 종료시키기 때문에 PGMTWO를 호출한 뒤로는 메세지 모니터가 지정되지 않습니다.

요구 처리 프로시듀어나 프로그램이 호출되지 않았는데 종료 요구가 시도되면 오류 메세지가 발행되고 종료 조작이 수행되지 않습니다.

주: 샘플 프로그램에서 RCVMSG 명령은 요구 프로세서가 되는 데 필요한 최소한의 매개변수를 사용합니다. 사용자가 요구 메세지의 수신은 원하지만 그것을 제거하는 것은 원하지 않음을 알릴 필요가 있습니다. 또한 메세지 요구가 시작된 특정 호출 대기행렬을 식별해야 합니다. 다른 매개변수는 필요에 따라 추가될 수 있습니다.

CL 프로시듀어에서 메세지 검색

RTVMSG(메세지 검색) 명령을 사용하면 메세지 파일에서 메세지의 텍스트를 검색하여 변수에 넣을 수 있습니다. RTVMSG는 사전정의 메세지 설명에 대해 작업합니다. 메세지 ID와 메세지 파일명을 다음에 추가하여 지정할 수 있습니다.

- 변환될 CCSID. 메세지 텍스트와 자료를 리턴시키려는 코드화 문자 세트 ID
- 메세지 자료 필드. 대체 변수에 대한 메세지 자료.
- 메세지 자료 CCSID. 제공된 메세지 자료가 고려되어야 하는 코드화 문자 세트 ID를 지정합니다.
- 다음 정보가 들어갈 CL 변수의 그룹(각각의 정보는 하나의 변수와 대응됨)
  - 메세지(문자 변수, 가변 길이)
  - 대체 변수 자료의 길이를 포함한 메세지 길이(10진 변수, 소수 5자릿수)

- 메세지 온라인 도움말 정보(문자 변수, 가변 길이)
- 대체 변수 자료의 길이를 포함한 메세지 도움말의 길이(10진 변수, 소수 5자릿수)
- 심각도 코드(10진 변수, 소수 2자릿수)
- 경고 옵션(문자 변수, 9자)
- 서비스 활동 기록부 안의 문제점 기록(문자 변수, 1자)
- 메세지 자료 CCSID(리턴된 대체 자료(10진 변수, 5자릿수)와 연관된 코드화 문자 세트 ID)
- 텍스트 자료 CCSID(메세지 및 메세지 도움말 매개변수(10진 변수, 소수 5자릿수)가 리턴시킨 텍스트와 연관된 코드화 문자 세트 ID)

예를 들면, 다음의 명령은 메세지 USR1001에 대한 메세지 설명을 메세지 파일 USRMSG에 추가합니다.

```
ADDMSGD   MSGID(USR1001) MSGF(QGPL/USRMSG) +
          MSG('File &1 not found in library &2') +
          SECLVL('Change file name or library name') +
          SEV(40) FMT((*CHAR 10) (*CHAR 10))
```

다음 명령은 검색된 메세지 USR1001에서 10자 변수 &FILE에 들어 있는 파일명 INVENT와 10자 변수 &LIB에 들어 있는 라이브러리명 QGPL을 대체하는 결과를 가져옵니다.

```
DCL &FILE TYPE(*CHAR) LEN(10) VALUE(INVENT)
DCL &LIB TYPE(*CHAR) LEN(10) VALUE(QGPL)
DCL &A TYPE(*CHAR) LEN(20)
DCL &MSG TYPE(*CHAR) LEN(50)
CHGVAR VAR(&A) VALUE(&FILE||&LIB)
RTVMSG   MSGID(USR1001) MSGF(QGPL/USRMSG) +
         MSGDTA(&A) MSG(&MSG)
```

&1과 &2의 자료는 프로시듀어 변수 &A에 들어 있습니다. 이 변수에 프로시듀어 변수 &FILE과 &LIB의 값이 연결되어 있습니다. CL 변수 &MSG에는 다음 메세지가 놓입니다.

```
파일 INVENT가 라이브러리 QGPL에 없음.
```

RTVMSG 명령에 MSGDTA 매개변수가 사용되지 않으면 다음 메세지가 CL 변수 &MSG에 놓입니다.

```
파일이 라이브러리에 없음.
```

메세지가 변수 &MSG에 놓인 후에는 다음 작업을 할 수 있습니다.

- SNDPGMMSG 명령을 사용한 메세지 송신
- 변수를 DDS 안의 메세지 행에 대한 텍스트로 사용(38열의 M)
- 메세지 서브파일 사용
- 메세지 인쇄 또는 표시

주: 텍스트에 들어 있는 변수명으로는 메세지 텍스트를 검색할 수 없습니다. 시스템은 RTVMSGD에 대해 송신 가능한 메세지를 리턴합니다.

메세지 대기행렬로부터 메세지 제거

메세지는 RMVMSG(메세지 제거) 명령, CLRMSGQ(메세지 대기행렬 지우기) 명령, RCVMSG(메세지 수신)과 SNDRPY(응답 송신) 명령의 RMV 매개변수, 메세지 표시 화면의 제거 기능 키 또는 메세지 대기행렬에 대한 작업 화면의 메세지 대기행렬 지우기 옵션을 사용하여 제거될 때까지 메세지 대기행렬에 남아 있습니다. 제거할 수 있는 메세지는 다음과 같습니다.

- 단일 메세지
- 모든 메세지
- 응답되지 않은 메세지를 제외한 모든 메세지
- 모든 기존 메세지
- 모든 새로운 메세지
- 모든 비활동 프로그램으로부터의 모든 메세지

RMVMSG 명령을 사용하여 단일 메세지를 제거하거나 RCVMSG 명령을 사용하여 단일 기존 메세지를 제거할 때에는 제거할 메세지의 메세지 참조 키를 지정해야 합니다.

주: 메세지 참조 키는 메세지 수신 및 메세지 응답에도 사용할 수 있습니다.

응답되지 않은 조회 메세지를 제거하면 디폴트 응답이 메세지의 송신자에게 송신되며 조회 메세지와 디폴트 응답이 제거됩니다. 이미 응답한 조회 메세지를 제거하면 메세지와 응답 모두 제거됩니다.

사용자의 작업 메세지 대기행렬로부터 모든 비활동 프로그램과 프로시듀어에 대한 메세지를 모두 제거하려면 RMVMSG 명령상의 PGMQ 매개변수에는 *ALLINACT를 지정하고 CLEAR 매개변수에는 *ALL을 지정하십시오. 모든 비활동 메세지를 제거하기 전에 작업 기록부를 인쇄하려면 DSPJOBLOG(작업 기록부 표시) 명령을 사용하고 OUTPUT 매개변수에 *PRINT를 지정하십시오.

ILE 프로시듀어의 호출 메세지 대기행렬에 대해 작업중인 경우 처리되지 않은 예외에 대한 예외 메세지가 RMVMSG 명령이 수행되는 시점에 대기행렬상에 있을 수 있습니다. 이 명령의 RMVEXCP 키워드가 이러한 유형의 메세지에 대한 조치를 제어하는 데 사용됩니다. 이 키워드에 *YES를 지정하면 RMVMSG 명령으로 인해 예외가 처리되고 메세지가 제거됩니다. *NO를 지정하면 메세지가 제거되지 않습니다. 결과적으로 예외는 처리되지 않습니다.

다음의 RMVMSG 명령은 사용자 메세지 대기행렬 JONES로부터 메세지를 제거합니다. 메세지 참조 키는 CL 변수 &MRKEY에 있습니다.

```
DCL &MRKEY TYPE(*CHAR) LEN(4)
RCVMSG MSGQ(JONES) RMV(*NO) KEYVAR(&MRKEY)
RMVMSG MSGQ(JONES) MSGKEY(&MRKEY)
```

다음의 RMVMSG 명령은 메세지 대기행렬로부터 모든 메세지를 제거합니다.

```
RMVMSG CLEAR(*ALL)
```

주: 사용자 메세지 대기행렬 또는 워크스테이션 메세지 대기행렬의 최대 메세지 수는 송신된 메세지의 유형별로 65,535개입니다. 예를 들어, 65 535 진단 메세지는 대기행렬상에 있을 수 있습니다. 65 535 완료 메세지도 대기행렬상에 있을 수 있습니다. 호출 메세지 대기행렬 또는 *EXT 대기행렬의 경우에는 유형당 최대 메세지 수에 제한이 없습니다.

CL 프로그램이나 프로시듀어에서의 메세지 모니터

사용자의 CL 프로시듀어나 프로그램의 호출 메세지 대기행렬로 송신된 이탈, 통지 및 상태 메세지의 경우 이 프로시듀어나 프로그램에 있는 명령 또는 다른 프로시듀어나 프로그램에 있는 명령으로 모니터할 수 있습니다. MONMSG(메세지 모니터) 명령은 명령에서 지정한 조건에 대해 호출 메세지 대기행렬로 송신되는 메세지를 모니터합니다. 조건이 존재하면 MONMSG 명령에 지정된 CL 명령이 수행됩니다. 다음은 MONMSG 명령에 관련된 논리입니다.

이탈 메세지: 이탈 메세지는 송신자를 강제로 종료하게 만든 오류 상태를 프로시듀어나 프로그램에 알리기 위해 송신됩니다. 이탈 메세지를 모니터하여 정정 조치를 수행하거나 프로시듀어 또는 프로그램을 지우고 종료할 수 있습니다.

상태 또는 통지 메세지: 상태 메세지와 통지 메세지는 송신자를 강제로 종료하게 만들 만큼 심각하지 않은 비정상 상태를 프로시듀어나 프로그램에 알리기 위해 송신됩니다. 상태 또는 통지 메세지를 모니터하여 사용자의 프로시듀어나 프로그램이 이 상태를 검출할 수 있고 기능이 계속되지 않게 할 수 있습니다.

다음 두 가지 레벨의 MONMSG 명령을 사용하여 메세지를 모니터할 수 있습니다.

- 프로시듀어 레벨: 사용자의 CL 프로시듀어나 프로그램에서 최종 선언 명령 바로 다음에 MONMSG 명령을 지정하여 프로시듀어 안의 명령이 송신한 이탈, 통지 또는 상태 메세지를 모니터할 수 있습니다. 이를 프로시듀어 레벨 MONMSG 명령이라고 합니다. 한 프로시듀어나 OPM 프로그램에서 100개까지의 프로시듀어 레벨 MONMSG 명령을 사용할 수 있습니다. (CL 프로시듀어나 OPM 프로그램에는 총 1000개의 MONMSG 명령이 들어 있을 수 있습니다.) 따라서 동일한 이탈 메세지는 모든 명령에 대해 동일한 방법으로 처리할 수 있습니다. EXEC 매개변수는 선택적이며 이 EXEC 매개변수에는 GOTO 명령만 지정할 수 있습니다.



- 특정 명령 레벨: 프로시듀어나 프로그램에 있는 특정 명령 바로 다음에 MONMSG 명령을 지정하면 특정 명령에 의해 송신된 이탈, 통지 또는 상태 메세지에 대해 모니터할 수 있습니다. 이를 명령 레벨 MONMSG 명령이라고 합니다. 사용자는 하나의 단일 명령에 최대 100개의 명령 레벨 MONMSG 명령을 사용할 수 있습니다. 따라서 서로 다른 이탈 메세지들을 다른 방법으로 처리할 수 있습니다.

이탈, 상태 또는 통지 메세지를 모니터하려면 MONMSG 명령에 다음 중 한 가지 방법으로 메세지에 대한 총칭 메세지 ID를 반드시 지정해야 합니다.

- pppmmnn

  특정 메세지에 대해 모니터합니다. 예를 들면, MCH1211은 ″0으로 나눔″이라는 이탈 메세지에 대한 메세지 ID입니다.

- pppmm00

  특정 사용권 프로그램(ppp)과 숫자 mm으로 시작하는 총칭 메세지 ID를 가진 모든 메세지를 모니터합니다. 예를 들면, CPF5100은 CPF51로 시작하는 모든 통지, 상태 및 이탈 메세지가 모니터됨을 의미합니다.

- ppp0000

  특정 사용권 프로그램(ppp)으로 시작하는 총칭 메세지 ID를 가진 모든 메세지를 모니터합니다. 예를 들면, CPF0000은 CPF로 시작하는 모든 통지, 상태 및 이탈 메세지가 모니터됨을 의미합니다.

  주: 전체 시스템의 설치, 저장 또는 복원과 같은 시스템 기능을 수행할 때에는 중요한 정보를 상실할 수도 있으므로 MONMSG CPF0000을 사용하지 마십시오.

- CPF9999

  모든 총칭 메세지 ID에 대한 기능 검사 메세지를 모니터합니다. 오류 메세지가 모니터되지 않으면 이는 CPF9999(기능 검사)가 됩니다.

주: 일반적으로 모니터중에 통지 및 상태 메세지가 송신되면 사용자의 모니터가 제어권을 갖게 됩니다.

메세지 ID를 이용하여 이탈 메세지를 모니터하는 것 외에도 MONMSG 명령에 지정한 문자 스트링을 메세지 안에 송신한 자료와 비교할 수 있습니다. 예를 들면, 다음 명령은 파일 MYFILE에 대한 이탈 메세지(CPF5101)를 모니터합니다. 파일명이 메세지 자료로서 송신됩니다.

```
MONMSG MSGID(CPF5101) CMPDTA(MYFILE) EXEC(GOTO EOJ)
```

이 비교 자료의 길이는 28자까지 가능합니다. 메세지 자료의 첫 번째 필드의 첫 문자에서부터 비교가 시작됩니다. 비교 자료가 메세지 자료와 일치하면 EXEC 매개변수에 지정된 조치가 수행됩니다.

MONMSG 명령의 EXEC 매개변수는 이탈 메세지가 처리되는 방법을 지정합니다. PGM, ENDPGM, IF, ELSE, DCL, DCLF, ENDDO 및 MONMSG 이외의 명령을 EXEC 매개변수에 지정할 수 있습니다. EXEC 매개변수에 DO 명령을 지정할 수도 있는데 이 경우는 DO 그룹 안의 명령이 수행됩니다. 명령이나 DO 그룹이 (EXEC 매개변수에서) 실행되면 이탈 메세지를 송신한 명령 다음에 있는 사용자의 프로시듀어나 프로그램의

명령으로 제어가 리턴됩니다. 그러나 GOTO나 RETURN 명령을 지정했을 때는 제어가 리턴되지 않습니다. EXEC 매개변수를 지정하지 않으면 이탈 메세지는 무시되고 사용자의 프로시듀어가 계속됩니다.

다음은 ″0으로 나눔″ 이탈 메세지인 메세지 ID MCH1211을 모니터하는 CHGVAR(변수 변경) 명령입니다.

```
CHGVAR VAR(&A) VALUE(&A / &B)
MONMSG MSGID(MCH1211) EXEC(CHGVAR VAR(&A) VALUE(1))
```

변수 &A의 값이 &A를 &B로 나눈 값으로 변경됩니다. &B가 0이면 나눗셈 연산을 수행할 수 없으며 0으로 나눗셈을 시도한다는 이탈 메세지가 프로시듀어로 송신됩니다. 이 경우 &A의 값이 (EXEC 매개변수에 지정된) 1로 변경됩니다. &B가 0인지도 테스트할 수 있으며 0이 아닌 경우에만 나눗셈을 수행할 수 있습니다. 이것은 이탈 메세지에 대한 조작 및 모니터를 시도하는 것보다 더 효율적입니다.

다음 예에서는 프로시듀어가 CHKOBJ(오브젝트 검사) 명령상의 이탈 메세지 CPF9801(오브젝트가 메세지를 발견하지 못함)을 모니터합니다.

```
   PGM
          CHKOBJ LIB1/PGMA *PGM
          MONMSG MSGID(CPF9801) EXEC(GOTO NOTFOUND)
          CALL LIB1/PGMA
          RETURN
NOTFOUND: CALL FIX001 /* PGMA Not Found Routine */
           ENDPGM
```

다음 CL 프로시듀어에는 두 개의 CALL 명령과 CPF0001에 대한 한 개의 프로시듀어 레벨 MONMSG 명령이 들어 있습니다. (이 이탈 메세지는 CALL 명령이 정상적으로 완료될 수 없을 때 발생합니다.) CALL 명령이 실패하면 프로시듀어가 완료 메세지를 송신하고 종료됩니다.

```
           PGM
        MONMSG MSGID(CPF0001) EXEC(GOTO ERROR)
        CALL PROGA
        CALL PROGB
          RETURN
ERROR:  SNDPGMMSG MSG('A CALL command failed') MSGTYPE(*COMP)
           ENDPGM
```

EXEC 매개변수가 프로시듀어 레벨 MONMSG 명령에 코드화되지 않으면 MONMSG 명령이 처리하는 모든 이탈 메세지가 무시됩니다. IF 명령 조건을 제외한 모든 명령에서 이탈 메세지가 발생하면 프로시듀어나 프로그램은 이탈 메세지가 발생하지 않을 경우 다음에 실행될 명령으로 처리를 계속합니다. 이탈 메세지가 IF 명령 조건에서 발생하면 IF 명령상의 조건이 거짓인 것으로 간주하고 처리를 계속합니다. 다음 예는 이탈 메세지가 프로시듀어 안의 서로 다른 지점에서 발생한 경우를 설명합니다.

```
   PGM
DCL &A TYPE(*DEC) LEN(5 0)
DCL &B TYPE(*DEC) LEN(5 0)
```

```
MONMSG MSGID(CPF0001 MCH1211)
CALL PGMA PARM(&A &B)
IF (&A/&B *EQ 5) THEN(CALL PGMB)
ELSE CALL PGMC
CALL PGMD
           ENDPGM
```

이탈 메세지가 발생하는 위치에 따라 다음 사항이 발생합니다.

- PGMA에 대한 호출에서 CPF0001이 발생하면 프로시듀어가 IF 명령에서 처리를 재개합니다.
- IF 명령에서 MCH1211(0으로 나눔)이 발생하면 IF 조건이 거짓으로 간주되고 프로시듀어가 PGMC를 호출하여 처리를 재개합니다.
- CPF0001이 PGMB 또는 PGMC에 대한 호출에서 발생하면 프로시듀어가 PGMD를 호출하여 처리를 재개합니다.
- CPF0001이 PGMD에 대한 호출에서 발생하면 프로시듀어가 ENDPGM 명령에 대한 처리를 재개하여 호출 프로시듀어로 리턴시킵니다.

프로시듀어나 프로그램에 있는 특정 명령 및 다른 명령에 의해 송신되는 동일한 이탈 메세지에 대해서도 모니터가 가능합니다. 이를 위해서는 두 개의 MONMSG 명령이 필요합니다. 하나의 MONMSG 명령은 이탈 메세지에 대한 특수 처리를 필요로 하는 명령 다음에 나옵니다. 그 명령의 경우 이 MONMSG 명령은 이탈 메세지가 송신될 때 사용됩니다. 또 다른 MONMSG 명령은 최종 선언 명령 다음에 나오며 다른 모든 명령에 대해 사용됩니다.

MONMSG 명령은 명령이 코드화된 CL 프로시듀어나 OPM 프로그램에만 적용됩니다. 하나의 프로시듀어와 다른 프로시듀어 둘 다 동일한 프로그램의 일부일지라도, 한 프로시듀어의 명령이 다른 프로시듀어에 적용되지 않습니다. IBM에서는 CL 명령을 위해 발행되는 이탈, 통지, 상태 메세지의 리스트가 나오는 온라인 도움말을 제공합니다. 이 정보에 대해서는 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오. 사용자는 자신이 정의한 모든 메세지의 리스트를 보관해야 합니다.

주: 위의 설명은 메세지의 여과 방식으로 인해 ILE 프로시듀어에는 해당되지 않습니다. 프로시듀어로 송신된 이탈 메세지를 처리하기 위해서는 시스템에 MONMSG가 필요합니다. 그렇지 않으면 이탈 메세지를 처리하거나 제어 경계에 일치하는 프로시듀어를 찾기까지 메세지가 호출 스택을 여과합니다.

디폴트 처리

많은 이탈 메세지들이 명령, 프로그램 및 프로시듀어를 호출한 프로시듀어로 송신될 수 있습니다. 사용자는 이 메세지 전부를 모니터하여 처리하기를 원치 않을 수 있습니다. 그러나 프로시듀어의 기능에 대한 이탈 메세지들은 모니터하고 처리하고자 할 수도 있습니다. 시스템은 사용자가 모니터하지 않는 메세지에 대한 디폴트 모니터링 및 처리를 제공합니다.

디폴트 처리는 오류가 프로시듀어 내에서 발견된 것으로 가정합니다. 프로시듀어를 디버깅하는 경우 메세지가 사용자의 표시장치로 송신됩니다. 그러면 사용자는 명령을 입력하여 오류를 분석하고 정정할 수 있습니다. 사용자가 프로시듀어를 디버깅하지 않으면 시스템이 메세지 여과 기능을 수행합니다.

메세지 여과는 다음과 같이 하는 두 단계 기능입니다.

- 이탈 메세지를 호출 스택에서 한 단계 앞으로 이동합니다.
- 프로시듀어가 이탈 메세지에 대한 MONMSG 명령을 갖고 있는지를 알아보기 위해 검사합니다.

프로시듀어에 이탈 메세지에 대한 MONMSG 명령이 있으면, 메세지 여과 조치가 중단되고 MONMSG 명령이 지정한 조치가 수행됩니다. MONMSG 명령을 발견하거나 가장 가까운 제어 경계를 찾을 때까지 메세지 여과가 계속됩니다. 이것은 이탈 메세지가 제어 경계를 통해 여과하지 않음을 의미합니다.

메세지에 적용되는 MONMSG 명령이 있는 프로시듀어를 발견하기 전에 제어 경계가 발견되면 기능 검사 처리가 시작됩니다. 시스템은 원래 이탈 예외에 대한 조치를 완료로 간주합니다. 따라서 시스템이 기능 검사 메세지(CPF9999)를 원래 이탈의 목표인 프로시듀어로 송신합니다. 그 프로시듀어에 기능 검사 메세지를 위한 MONMSG가 있을 경우 그 명령이 지정한 조치가 수행됩니다. 또는 작업이 대화식 작업일 경우 시스템이 조회 메세지를 워크스테이션 오퍼레이터에게 송신합니다. 워크스테이션 오퍼레이터는 다음 응답 중 하나를 사용하여 응답할 수 있습니다.

R 프로시듀어에서 실패 명령 재시도

I 메세지 무시. 프로시듀어 안의 다음 명령에서 계속 처리

C 프로시듀어를 취소하고 호출 스택상의 다음 이전 프로시듀어에 대한 기능 검사 여과

D 실패 프로시듀어에 대한 호출 스택 항목 덤프, 프로시듀어 취소 및 호출 스택상의 다음 이전 프로시듀어에 대한 기능 검사 여과. 응답이 입력되지 않거나 작업이 일괄처리 작업인 경우 이것이 디폴트 조치입니다.

시스템은 제어 경계를 통해 기능 검사를 여과하지 않습니다. 활성 그룹 경계를 통해 이동하여 기능 검사를 하도록 응답이 되면 기능 검사에 대한 더 이상의 조치가 중단됩니다. 시스템이 활성 그룹 경계까지의 모든 프로시듀어를 취소하고 이탈 메세지 CEE9901을 이전 호출 스택 항목으로 송신합니다.

기능 검사 이탈 메세지를 모니터하여 다음을 수행할 수 있습니다.

- 프로시듀어를 정리하고 종료시킴
- 프로시듀어의 다른 요소들에 대해 작업함

주: 모니터되지 않은 이탈 메세지에 대한 메세지 설명이 디폴트 조치를 지정하면 기능 검사 메세지가 송신되기 전에 디폴트 처리 프로그램이 호출됩니다. 디폴트 처리 프로그램이 리턴되면 기능 검사 처리가 시작됩니다.

통지 메세지

이탈 메세지를 모니터하는 것 이외에도, 사용자의 CL 프로시듀어나 프로그램에 있는 명령 또는 이 프로시듀어나 프로그램이 호출하는 프로그램과 프로시듀어에 의해 사용자의 CL 프로시듀어나 프로그램의 호출 메세지 대기행렬로 송신된 통지 메세지에 대해서도 모니터가 가능합니다. 통지 메세지는 일반적인 오류가 아닌 상태임을 사용자의 프로시듀어나 프로그램에 알려주기 위해 송신됩니다. 통지 메세지를 모니터하면 그 상태가 발견되지 않았을 때와는 다른 조치를 지정할 수 있습니다. IBM 제공 명령은 거의 통지 메세지를 송신하지 않습니다.

통지 메세지의 모니터 및 처리는 이탈 메세지의 모니터 및 처리와 유사합니다. 차이점은 사용자가 통지 메세지를 모니터하지 않거나 처리하지 않을 경우 발생하는 현상에 있습니다. 또한 통지 메세지는 프로시듀어에서 활성 그룹 경계 안의 프로시듀어로도 여과됩니다. MONMSG 명령이 활성 그룹 경계를 찾지 않고 활성 그룹 경계에 이르면, 디폴트 응답이 자동으로 통지 메세지의 송신자에게 리턴되고 송신자는 처리를 계속할 수 있습니다. 이탈 메세지와는 달리, 모니터되지 않은 통지 메세지는 사용자의 프로시듀어나 프로그램에서 오류 표시로 간주되지 않습니다.

상태 메세지

사용자의 CL 프로시듀어에 있는 명령 또는 이 프로시듀어가 호출하는 프로그램이나 프로시듀어에 의해 송신되는 상태 메세지도 모니터가 가능합니다. 상태 메세지는 사용자의 프로시듀어에 송신자가 수행한 작업 상태를 알려줍니다. 상태 메세지를 모니터하여 송신 프로그램이나 프로시듀어가 더 이상 처리되는 것을 막을 수 있습니다.

상태 메세지에 관한 메세지 정보는 메세지 대기행렬에 저장되지 않습니다. 따라서 상태 메세지는 수신될 수 없습니다.

상태 메세지가 모니터되지 않으면 이탈 메세지와 통지 메세지에서처럼 여과됩니다. MONMSG 명령이 없는 상태에서 활성 그룹 경계에 이르면, 메세지상의 조치가 완료된 것으로 간주되고 제어가 메세지 송신자에게 리턴되어 처리가 계속됩니다. 상태 메세지는 처리가 계속될 수 있는 지점에서 발견된 정상적인 상태를 알려주기 위해 송신됩니다.

외부 메세지 대기행렬에 송신된 상태 메세지는 사용자에게 기능이 처리 중임을 대화식 화면에 표시됩니다. 예를 들면, CPYF(파일 복사) 명령은 사용자에게 복사 작업이 처리 중임을 알리는 메세지를 송신합니다.

사전정의된 메세지들만이 상태 메세지로 송신될 수 있으며, 즉시 메세지는 송신될 수 없습니다. 사용자는 시스템 제공 메세지 ID CPF9898을 사용할 수 있고, 기존의 메세지 설명이 없을 때는 상태 메세지를 송신하기 위한 메세지 자료를 제공할 수 있습니다.

기능이 완료되면 사용자의 프로시듀어나 프로그램이 대화식 화면에서 상태 메세지를 제거해야 합니다. 명령을 사용해서는 메세지를 제거할 수 없지만, 공백 메세지가 들어 있는 다른 상태 메세지를 *EXT에 송신하면 메세지를 제거한 것과 같은 결과를 얻을 수 있습니다. 시스템 제공 메세지 ID CPI9801이 이러한 목적으로 사용될 수 있습니다. OS/400 프로그램으로 제어를 리턴시키면, *STATUS 메세지는 CPI9801 메세지를 송신하지 않고도 24행부터 지워질 수 있습니다. 다음의 예는 메세지 ID CPF9898 및 CPI9801을 적용한 전형적인 어플리케이션을 나타냅니다.

```
SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) +
          MSGDTA('Function xxx being performed') +
          TOPGMQ(*EXT) MSGTYPE(*STATUS)
    •
    •   /*  Your processing function  */
    •
SNDPGMMSG MSGID(CPI9801) MSGF(QCPFMSG) +
          TOPGMQ(*EXT) MSGTYPE(*STATUS)
```

상태 메세지의 표시 방지

사용자는 명령이 상태 메세지를 송신하는 것은 막을 수 없으나, 상태 메세지가 화면 맨 아래에 표시되는 것은 막을 수 있습니다.

상태 메세지가 표시되는 것을 막기 위해서는 두 가지 방법을 사용할 수 있습니다.

- CHGUSRPRF(사용자 프로파일 변경) 명령

  사용자 프로파일을 변경하여 사용자가 이 프로파일로 사인 온할 때마다 상태 메세지가 표시되지 않도록 할 수 있습니다. 이를 위해서는 CHGUSRPRF 명령을 사용하여 사용자 옵션(USROPT) 매개변수에 *NOSTSMSG를 지정해야 합니다.

- 작업 변경(CHGJOB) 명령

  현재 수행 중인 작업을 변경하여 상태 메세지가 나타나지 않도록 할 수 있습니다. 이를 위해서는 CHGJOB 명령을 사용하여 상태 메세지(STSMSG) 매개변수에 *NONE을 지정해야 합니다. 또한 CHGJOB 명령을 사용하여 상태 메세지를 보려면 STSMSG 매개변수에 *NORMAL을 지정하면 됩니다.

자주 사용되는 것은 아니지만 OVRMSGF(메세지 파일 대체) 명령을 사용하여 상태 메세지 ID를 공백 메세지로 변경하는 세 번째 방법이 있습니다.

일시 중단 처리 프로그램

일시 중단 처리 프로그램은 *BREAK 모드의 메세지 대기행렬에 메세지가 도착할 때 자동으로 호출되는 프로그램입니다. 반드시 프로그램과 일시 중단 전달 이름을 동일한 CHGMSGQ(메세지 대기행렬 변경) 명령에 지정해야 합니다. CHGMSGQ 명령에 프로그램을 지정하더라도 그것은 메세지를 처리하는 프로그램 안에 있는 하나 이상의 프로시듀어입니다. 이 프로그램 안의 프로시듀어가 RCVMSG(메세지 수신) 명령을 실행하여 메세지를 수신해야 합니다. 메세지를 수신하여 처리하기 위해 일시 중단 전달 메세지를 처리하도록 호출된 사용자 정의 프로그램이 매개변수를 수신합니다. 즉 프로그램 안에서 실행되는 첫 번째 프로시듀어가 이 매개변수를 수신합니다. 매개변수가 메세지 대기행렬과 일시 중단의 원인이 되는 메세지의 메세지 참조 키(MRK)를 식별합니다. RCVMSG(메세지 수신) 명령에 대한 온라인 도움말 정보는 일시 중단 처리 종료 프로그램의 매개변수를 설명합니다. **iSeries Information Center** *프로그래밍* 범주의 **CL** 섹션에 나와 있는 명령 문서를 참조하십시오. 시스템이 일시 중단 처리 프로그램을 호출하면 일시 중단(break) 모드에서 메세지 대기행렬을 가지고 있는 작업을 인터럽트합니다. 일시 중단(break) 처리 프로그램이 종료되면 원래 프로그램이 처리를 재개합니다.

다음 프로그램(PGMA)은 일시 중단 처리 프로그램의 예로 하나의 프로시듀어로만 구성되어 있습니다.

```
   PGM PARM(&MSGQ &MSGLIB &MRK)
       DCL VAR(&MSGQ) TYPE(*CHAR) LEN(10)
       DCL VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
DCL VAR(&MRK) TYPE(*CHAR) LEN(4)
DCL VAR(&MSG) TYPE(*CHAR) LEN(75)
RCVMSG  MSGQ(&MSGLIB/&MSGQ) MSGKEY(&MRK) +
        MSG(&MSG)
.
.
.
           ENDPGM
```

일시 중단 처리 프로그램이 작성된 후 다음 명령을 수행하면 그 프로그램을 QSYSMSG 메세지 대기행렬에 연결시킬 수 있습니다.

```
CHGMSGQ   MSGQ(QSYS/QSYSMSG) DLVRY(*BREAK) PGM(PGMA)
```

주:

1. 메세지가 처리되면 그 메세지는 메세지 대기행렬에서 제거되어야 합니다. 메세지 대기행렬이 일시 중단 모드에 있으면, 대기행렬상의 메세지로 인해 일시 중단 처리 프로그램이 호출됩니다.
2. 메세지를 수신하는 프로시듀어나 프로그램이 메세지를 수신하기 위해 0 이외의 대기 시간을 갖도록 코드화되어서는 안됩니다. RCVMSG(메세지 수신) 명령을 사용하여 대기 매개변수에 0 이외의 값을 지정할 수 있습니다. 메세지 도착 이벤트는 작업이 일시 중단(break) 처리 이벤트를 수행 중인 동안 시스템에서 처리될 수 없습니다.

일시 중단 처리 프로그램의 예로 프로그램이 메세지를 송신하도록 하는데, 이 메세지는 보통 QSYSOPR 대기행렬로 송신되거나, QSYSOPR 대신에 다른 대기행렬로 송신되거나 또는 QSYSOPR에 송신된 후 추가로 다른 대기행렬에도 송신됩니다.

다음은 일시 중단 메세지를 처리하는 (하나의 프로시듀어만 사용하는) 사용자 정의 프로그램의 예입니다. 이 프로그램이 사용될 때 표시장치 사용자가 메세지 CPA5243(장치 &1에 대해 준비 완료, 시작 또는 시작-중단을 누르십시오.)과 CPA5316(장치 &3에 대한 정렬을 확인하십시오.)에 응답할 필요는 없습니다.

```
BRKPGM:     PGM (&MSGQ &MSGQLIB &MSGMRK)
            DCL &MSGQ TYPE(*CHAR) LEN(10)
            DCL &MSGQLIB TYPE(*CHAR) LEN(10)
            DCL &MSGMRK TYPE(*CHAR) LEN(4)
DCL &MSGID TYPE(*CHAR) LEN(7)
            RCVMSG MSGQ(&MSGQLIB/&MSGQ) MSGKEY(&MSGMRK) +
                   MSGID(&MSGID) RMV(*NO)
            /* Ignore message CPA5243 */
            IF (&MSGID *EQ 'CPA5243') GOTO ENDBRKPGM
            /* Reply to forms alignment message */
            IF (&MSGID *EQ 'CPA5316') +
                   DO
                   SNDRPY MSGKEY(&MSGMRK) MSGQ(&MSGQLIB/&MSGQ) RPY(I)
 ENDDO
            /* Other messages require user intervention */
            ELSE CMD(DSPMSG MSGQ(&MSGQLIB/&MSGQ))
ENDBRKPGM:  ENDPGM
```

경고:

위의 일시 중단 처리 프로그램의 예에서는 CPA5316 메세지가 DSPMSG 명령이 수행중인 동안 대기행렬에 도달해야 할 경우 일시 중단 및 CPA5316 메세지를 발생시킨 원래의 메세지가 DSPMSG 화면에 표시됩니다. DSPMSG 화면은 처리를 계속하기 전에 오퍼레이터가 CPA5316 메세지에 응답하기를 기다립니다.

주: 인터럽트된 프로그램이 화면에서 입력 자료를 기다리고 있다면 이 프로그램은 화면 파일을 열 수 없습니다.

시스템 응답 리스트를 사용하여 시스템이 사전정의된 조회 메세지에 대해 응답할 것이라는 것을 나타낼 수 있습니다. 따라서 표시장치 사용자는 응답할 필요가 없습니다. 더 자세한 내용은 315 페이지의 『시스템 응답 리스트의 사용』을 참조하십시오.

사용자 일시 중단 처리 프로그램 안의 프로시듀어가 메세지 처리 기능이 수행되고 있는 동안 표시를 일시중단하고 복원되게 하려면 일시 중단 및 복원 프로시듀어가 필요할 수 있습니다. 일시중단 및 복원 프로시듀어는 다음 조건이 존재하는 경우에만 필요합니다.

- 일시 중단 프로그램 안의 프로시듀어가 다른 화면이나 메뉴를 표시하는 경우
- 일시 중단 프로그램이 다른 메뉴나 화면을 표시할 수 있는 다른 프로그램을 호출하는 경우

다음 예는 표시를 일시중단하고 복원하는 데 필요한 사용자 프로시듀어와 화면 파일을 설명합니다.

주: 화면 파일을 작성하려면 RSTDSP(*YES)를 지정해야 합니다.

```
            A          R SAVFMT                      OVERLAY  KEEP
      A*
      A           R DUMMY                        OVERLAY
      A                                          KEEP
      A                                          ASSUME
      A             DUMMYR          1A     1  2DSPATR(ND)


   PGM PARM(&MSGQ &MSGLIB &MRK)
       DCL VAR(&MSGQ) TYPE(*CHAR) LEN(10)
       DCL VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
       DCL VAR(&MRK) TYPE(*DEC) LEN(4)
       DCLF FILE(UDDS/BRKPGMFM)
       SNDF RCDFMT(SAVFMT)
       CALL PGM(User's Break Program)
       SNDF RCDFMT(SAVFMT)
           ENDPGM
```

사용자 지정 일시 중단 처리 프로그램이 대화식 작업을 인터럽트하는 것을 원하지 않으면 일괄처리에서 실행하도록 프로그램을 제출해야 합니다. 일시 중단 처리 프로그램이 메세지를 수신한 후 SBMJOB를 수행하도록 지정함으로써 이를 수행할 수 있습니다. SBMJOB는 사용자가 사용하려는 매개변수로 현재의 일시 중단 처리 프로그램을 호출합니다. (예는 수신 메세지의 정보입니다.) 그러면 대화식 작업으로 제어가 리턴되고 정상적으로 계속됩니다.

QSYSMSG 메세지 대기행렬

QSYSMSG 메세지 대기행렬은 QSYS 라이브러리에 작성할 수 있는 선택 대기행렬입니다. 이것이 존재하고 손상되지 않았으면, 특정 메세지들은 QSYSOPR 메세지 대기행렬 대신에 또는 QSYSOPR에 추가하여 SYSOPR 메세지 대기행렬에 직접 전송됩니다. 이렇게 해서 특정 메세지가 송신되면 사용자 작성 프로그램으로 제어가 이전될 수 있습니다. 특정 메세지의 수신을 원할 경우에만 QSYSMSG 대기행렬을 작성해야 합니다.

QSYSMSG 대기행렬을 작성하려면 다음 명령을 입력하십시오.

```
CRTMSGQ  QSYS/QSYSMSG +
         TEXT('Optional MSGQ to receive specific system messages')
```

QSYSMSG 메세지 대기행렬이 일단 작성되면 모든 특정 메세지(아래의 291 페이지의 『QSYSMSG 메세지 대기행렬로 송신되는 메세지』 참조)가 이 대기행렬로 지정됩니다. 사용자는 특별한 조치를 수행할 수 있는 메세지를 수신하고 다른 메세지를 QSYSOPR 메세지 대기행렬이나 다른 메세지 대기행렬로 송신하기 위한 프로그램을 작성할 수 있습니다. 이 프로그램은 일시 중단 처리 프로그램으로 작성되어야 합니다.

QSYSMSG 메세지 대기행렬로 송신되는 메세지

이 주제는 QSYSMSG 메세지 대기행렬에 송신된 특정 메세지에 대해 설명합니다. QSYSMSG 메세지 대기행렬이 있으면, 시스템은 다음 메세지를 QSYSOPR 대신 QSYSMSG로 송신합니다.

- CPF1269
- CPF1393
- CPF1397
- CPI2209
- CPI9014
- CPI96C7을 통한 CPI96C0

시스템은 메세지와 함께 송신된 시스템 참조 코드(SRC) 및 중요 메세지 처리와 함께 SRC가 기록되고 있는지 여부에 따라 QSYSMSG 및 QSYSOPR 모두에 또는 둘 중 하나에만 특정 메세지를 송신합니다. 이러한 메세지에는 다음이 포함됩니다.

- CPP0DDD
- CPP1604
- CPPEA02
- CPPEA04
- CPPEA05
- CPPEA12
- CPPEA13
- CPPEA26
- CPPEA32
- CPPEA38
- CPPEA39

시스템은 이 주제에 나오는 다른 모든 메세지를 QSYSMSG와 QSYSOPR에 모두 송신합니다.

CPD4070

리모트 위치 &5, 장치 설명 &4에 대해 수신된 부정 응답.

CPF0907

기억장치에 심각한 상태가 존재합니다. HELP를 누르십시오.

시스템 보조 기억장치 풀(system auxiliary storage pool)에서 사용 가능한 보조 기억장치의 크기가 임계값에 도달하면 이 메세지가 송신됩니다.

시스템 서비스 툴 기능을 사용하여 임계값을 표시하고 변경할 수 있습니다. 자세한 정보는 백업 및 회복  책을 참조하십시오.

CPF1269

통신 장치에 수신된 프로그램 시작 요구가 이유 코드와 함께 거부되었습니다.

이 메세지는 시작 요구가 거부되었을 때 송신되며, 시작 요구를 거부한 이유를 나타내는 이유 코드가 들어 있습니다.

암호가 유효하지 않거나, APPC 사용에 대해 권한이 없는 상태가 발생하면 이는 정상적인 작업이 오류 상태이거나 다른 사용자가 보안을 침해하려고 시도하고 있음을 의미합니다. 사용자는 다음 사항을 수행하여 이러한 상태의 원인을 파악할 때까지 APPC 장치 설명이 더 이상 사용되지 못하도록 선택할 수 있습니다.

- 메세지를 QSYSOPR 메세지 대기행렬로 송신함.
- 보안 담당자가 검토하는 사항을 기록함.
- ENDMOD(모드 종료) 명령을 발행하여 허용된 작업을 0으로 설정함. 그러면 현재 대등 장치 설명을 사용하는 작업을 활동 상태(active)로 유지할 수 있으며 상태의 원인을 알 때까지 다른 작업의 시작을 금지할 수 있습니다.
- 주어진 시간 동안의 시도 횟수를 계산함. 사용자는 중대한 조치(최대 세션 수를 0으로 변경하는 것과 같은)를 수행하기 전에 유효하지 않은 시도의 횟수에 대한 임계값을 프로그램에 설정할 수 있습니다. 그러기 위해서 사용자는 작업 단위 ID(공백일 수도 있음)나 APPC 장치 설명으로, 또는 사용 중인 APPC 환경 전체에 대해 이 임계값을 할당할 수 있습니다.

CPF1393

사인 온 시도 최대수에 도달했기 때문에 사용자 프로파일을 사용할 수 없게 되었습니다.

사용자가 사인 온을 여러 번 시도하여 사용자 프로파일이 사용 불가능해진 경우에 이 메세지가 송신됩니다.

CPF1397

서브시스템이 워크스테이션을 단절변환했습니다.

시스템 값 QMAXSIGN에 의해 할당된 임계값에 도달하여 장치가 단절변환된 경우 이 메세지가 송신됩니다. 이 메세지는 사용자가 유효한 암호를 입력하지 않았음을 나타냅니다. CPF1397에 대한 메세지 자료에는 이 메세지를 송신했던 장치명이 들어 있습니다. 이 정보를 사용하여 적절한 조치를 수행하는 프로그램을 설계할 수 있습니다. 다음 조치 중 하나 이상을 수행하는 것이 좋습니다.

- 동일한 메세지를 QSYSOPR 메세지 대기행렬로 송신함

- 보안 담당자의 검토 횟수를 기록함
- 유효 시간이 지난 후 장치를 자동으로 연결변환(vary on)함

CPF510E

장치 &4에 읽거나 쓰는 중에 네트워크 인터페이스 &9가 실패했습니다.

시스템이 네트워크 인터페이스 장애를 감지했으며, 네트워크 인터페이스를 위한 오류 회복을 시도하는 중입니다.

다시 요구를 시도하십시오. 프로그램 회복에 관한 권장사항은 Communications Management 책을 참조하십시오. 문제점이 다시 발생할 경우 ANZPRB(문제점 분석) 명령을 입력하여 문제점 분석을 실행하십시오.

CPF5167

리모트 위치 &5, 장치 설명 &4에 대한 SNA 세션이 비정상 종료되었습니다.

리모트 제어기로부터 수신된 시스템 RSHUTD(종료 요구), RQR(회복 요구), UNBIND 또는 NOTIFY(전원 차단 통지) 명령으로 인해 시스템 네트워크 구조(SNA) 세션이 종료되었습니다.

리모트 제어기 오퍼레이터에게 문의하여 통신 지원이 세션을 종료시킨 이유를 판별하십시오. 오류를 정정한 후 다시 요구하십시오.

CPF5244

리모트 위치 &5, 장치 설명 &4에 대한 내부 시스템 실패입니다.

장치를 단절변환하십시오. 장치를 연결변환한 후 다시 요구하십시오. 문제점이 계속되면 문제점을 보고하십시오(ANZPRB 명령).

CPF5248

리모트 위치 &5, 장치 설명 &4에 대해 수신된 자료의 SNA 프로토콜 위반입니다.

리모트 위치 &5, 장치 설명 &4에 대해 수신된 시스템 네트워크 구조(SNA)가 SNA 프로토콜을 위반했습니다. 시스템이 제어기에 대한 감지 자료 &7의 부정 응답을 수신했습니다.

제어기 프로그램에서 문제점을 정정한 후 다시 요구를 시도하십시오. 감지 자료 및 관련 오류에 대한 자세한 내용은 Finance Communications Programming 책을 참조하십시오.

CPF5250

리모트 위치 &5에 대해 감지 자료 &7의 부정 응답을 수신했습니다.

시스템이 리모트 위치 &5 장치 설명 &4에 대해 감지 자료 &7의 부정 응답을 수신했습니다. 자료의 처음 네 자는 10xx, 08xx 또는 0000으로 시작하지 않습니다. 그와 같은 값이 있을 경우에는 시스템 네트워크 구조(SNA) 세션이 종료됩니다.

오류를 정정한 후 다시 요구를 시도하십시오. 감지 자료 및 부정 응답의 원인에 대한 추가 정보는 *Systems Network Architecture Formats, GA27-3136*을 참조하십시오.

CPF5251

암호나 사용자 ID가 리모트 위치 &5에 대한 요구에 유효하지 않습니다.

시스템 네트워크 구조(SNA) INIT-SELF 명령이 유효한 권한 부여 자료를 포함하지 않은 재무관리 리모트 위치 &5, 장치 설명 &4에 대해 수신되었습니다. 다음 중 하나의 조건이 발생했습니다.

- 시스템이 사용자 ID나 암호를 찾을 수 없습니다.
- 시스템이 사용자 ID를 찾을 수 없습니다.
- 암호가 이 사용자 ID에 유효하지 않습니다.
- 이 사용자 ID에 장치 설명 &4를 사용하기 위한 권한 부여가 없습니다.
- 사용자 프로파일에 액세스할 수 없습니다.
- 사용자 ID에 유효하지 않은 문자가 있습니다.

유효한 사용자 ID와 암호로 사용자가 다시 요구를 시도하도록 하십시오. 사용자에게 장치에 대한 권한 부여가 없을 경우에는 GRTOBJAUT(오브젝트 권한 부여) 명령을 사용하여 사용자에게 이 장치에 대한 권한을 부여하십시오.

CPF5257

라이브러리 &3에 있는 장치나 멤버 &4 파일 &2에 대한 장애입니다.

읽기 또는 쓰기 조작 시 오류가 발생했습니다. 이것이 화면 파일인 경우 화면을 사용하지 못할 수 있습니다.

앞에 나온 메세지를 참조하여 오류를 정정한 후 다시 요구를 시도하십시오. 문제점이 계속되면 문제점을 보고하십시오(ANZPRB 명령).

CPF5260

&3의 파일 &2에서 장치 &4에 대한 교환 연결이 실패했습니다.

파일을 닫은 다음 다시 요구를 시도하십시오.

CPF5274

&3의 리모트 위치 &5 파일 &2에 대한 장치 오류입니다.

프로그램이 이전에 오류가 있었던 프로그램 장치 &4, 리모트 위치 &5에 대해 입력 조작이나 출력 조작을 시도했습니다.

리모트 위치 &5와 관련된 장치를 단절변환한 다음 다시 연결변환하십시오 (VRYCFG나 WRKCFGSTS 명령). 그런 다음 다시 요구를 시도하십시오.

CPF5341

리모트 위치 &5, 장치 설명 &4에 대해 SNA 세션이 설정되지 않았습니다.

시스템 네트워크 구조(SNA) 세션이 설정될 수 없습니다. 동기 자료 링크 제어 (SDLC) 프레임 *크기*가 요구/응답 단위(RU) *크기*와 호환되지 않습니다. 이는 구성 오류이거나, SDLC 프레임 *크기*가 OS/400에 의해 더 작은 값으로 조정된 것입니다. 리모트 제어기가 XID(식별 교환)를 사용하는 중에 이와 같은 일이 발생했습니다.

장치 설명의 MAXLENRU 매개변수에 유통 및 재무관리 장치를 위한 RU 크기의 *스펙*이 포함되어 있습니다.

회선 설명의 MAXFRAME 매개변수에 SDLC 프레임 *크기* *스펙*이 포함되어 있습니다. 제어기 설명의 MAXFRAME 매개변수에도 유통 및 재무관리 장치를 위한 *스펙*이 포함되어 있습니다.

다음 중 하나 이상을 수행한 다음 다시 요구를 시도하십시오.

- 프레임 *크기* 값과 RU *크기*값이 호환될 수 있는지 확인하십시오.
- 필요하면 SDLC 프레임 *크기*를 늘리거나 RU *크기*를 줄이십시오.
- 이 구성이 리모트 제어기 구성과 호환될 수 있는지 확인하십시오.
- 구성을 변경한 경우 변경이 유효하도록 하기 위해서는 먼저 구성을 단절변환하고 다시 연결변환해야 합니다.

CPF5342

장치 설명 &4, 리모트 위치 &5에서 회선 &9가 실패했습니다.

시스템이 입력이나 출력을 처리하는 중에 오류를 감지했으며, 현재 그 회선에 대해 오류 회복을 시도하는 중입니다.

다시 요구를 시도하십시오. 문제점이 계속되면 문제점 분석을 시작하십시오 (ANZPRB 명령). 프로그램 회복에 관한 권장사항은 Communications Management  책을 참조하십시오.

CPF5344

제어기 &9, 장치 설명 &4에 대한 오류입니다.

시스템이 제어기 장애를 감지했으며 제어기를 위한 오류 회복을 시도하는 중입니다.

다시 요구를 시도하십시오. 문제점이 계속되면 문제점 분석을 시작하십시오 (ANZPRB 명령).

CPF5346

리모트 위치 &5, 장치 설명 &4에 대한 오류입니다.

파일을 닫으십시오. 장치를 단절변환하십시오(VRYCFG 명령). 작업 기록부에 있는 시스템 오퍼레이터 메세지를 참조하여 장치를 연결변환하기 전에 필요한 조치가 있는지 알아보십시오. 오류를 정정한 다음 장치를 연결변환하십시오(VRYCFG 명령). 그런 다음 다시 요구를 시도하십시오. 문제점이 계속되면 문제점 분석을 시작하십시오(ANZPRB 명령).

CPF5355

유형 *&9의 &8에서 오브젝트 &7을 찾을 수 없습니다.

오브젝트 &7 유형 *&9가 다른 프로세스에서 사용되는 중이거나 연결변환되지 않았거나 APPC에 사용할 수 있는 세션이 없습니다.

파일 &2를 닫으십시오. 오브젝트 &7을 사용할 수 있거나 연결변환되었을 때 다시 요구를 시도하십시오. APPC에 할당될 수 없는 오브젝트인 경우 사용할 수 있는 세션이 없습니다. WAITFILE 매개변수를 변경하여 시스템이 세션을 사용할 수 있을 때까지 더 오랜 시간 동안 기다리도록 할 수 있습니다. 또한 더 많은 세션을 사용할 수 있도록 모드를 변경할 수도 있습니다(MAXSSN 매개변수). 많은 수의 세션을 허용하기 위해서 리모트 시스템을 다시 구성해야 할 경우도 있습니다. 충분한 수의 세션을 위한 구성이 있을 경우 CHGSSNMAX 명령을 사용하여 현재 세션 한계를 늘리십시오.

CPF8AC4

예약 라이브러리명 &7이 사용 중입니다.

사용자가 QDLS 파일 시스템에 대해 예약된 라이브러리명을 작성했습니다. 예를 들어, 사용자 ASP 5에 대해 시스템에 라이브러리명 QDOC0005가 예약되어 있습니다. 사용자가 사용자 ASP에서 첫 번째 문서 라이브러리 오브젝트(DLO)를 작성하려고 할 때 시스템이 이 라이브러리를 작성하려고 시도합니다. 그러나 사용자가 예약명 QDOC0005를 사용하여 고유의 라이브러리를 작성한 경우에는 CPF8AC4 메세지가 송신됩니다. 사용자 작성 라이브러리는 사용자 ASP에서 DLO를 작성하기 전에 삭제하거나 해당 이름을 변경해야 합니다.

CPF9E7C

Operating System/400의 유예 기간이 만기되었습니다.

Operating System/400의 소프트웨어 사용권 유예 기간이 만기되었습니다. 다음 초기 프로그램 로드(IPL)를 성공적으로 완료하기 위해서는 소프트웨어 사용권 키가 필요합니다.

신규 OS/400 소프트웨어 사용권에 관해서는 IBM 영업대표 또는 IBM 협력업체에 문의하십시오. 소프트웨어 사용권 키를 추가하려면 ADDLICKEY(사용권 키 정보 추가) 명령을 사용하십시오.

CPI091F

PWRDWNSYS &1 명령이 진행 중입니다.

1차 파티션이 비정상 종료되면 이 메세지가 2차 파티션으로 송신됩니다.

CPI0948

이중복사 보호가 디스크 장치 &1에서 일시중단되었습니다.

시스템이 기억장치를 찾을 수 없습니다. 자료는 유실되지 않았습니다. 다음 정보는 시스템 구성에서 기억장치가 누락되기 전 시스템이 기억장치를 배치한 곳을 나타냅니다.

- 디스크 일련 번호: &5
- 디스크 유형: &3
- 디스크 모델: &4
- 장치 자원명: &26

다음과 같이 하십시오.

1. '시스템 자원 구성 리스트' 화면을 사용하여 누락된 것으로 표시된 기억장치를 알아보십시오.
2. 기억장치에 전원 케이블 연결이 제대로 설치되었는지 확인하십시오.

CPI0949

이중복사 보호가 디스크 장치 &1에서 일시중단되었습니다.

디스크 이중복사 보호가 일시중단되었습니다.

CPI0950

현재 기억장치를 사용할 수 있습니다.

시스템 구성에서 누락되었던 기억장치가 지금은 사용 가능합니다. 자료는 유실되지 않았습니다.

CPI0953

ASP 기억장치 임계값에 도달했습니다.

지정된 보조 기억장치 풀(ASP: Auxiliary Storage Pool)의 사용 가능한 기억장치(storage)의 크기가 임계값에 도달하면 이 메세지가 송신됩니다. CPI0953에 대한 메세지 자료에는 보조 기억장치 용량, 사용된 보조 기억장치, 임계 퍼센트, 사용 가능한 보조 기억장치의 퍼센트가 들어 있습니다. 이 정보를 사용하여 적절한 조치를 수행하십시오.

CPI0954

ASP 기억장치 한계를 초과했습니다.

지정된 ASP에서 사용 가능한 기억장치가 모두 사용되면 이 메세지가 송신됩니다.

CPI0955

시스템 ASP 비보호 기억장치 한계를 초과했습니다.

시스템 ASP 안의 사용 가능한 기억장치가 모두 사용되면 이 메세지가 송신됩니다.

CPI095A

IASP &1 이중복사 기억장치 임계값에 도달했습니다.

지정된 독립 보조 기억장치 풀(ISAP)의 이중복사에 사용할 수 있는 기억장치의 크기가 임계값에 도달하면 이 메세지가 송신됩니다. CPI095A에 대한 메세지 자료에는 독립 보조 기억장치 용량과 임계 퍼센트가 포함됩니다. 적절한 조치를 수행하기 위해 이 정보를 사용할 수 있습니다.

CPI0964

약한 배터리 상태가 검출되었습니다.

외부의 무정전 전원 장치(UPS: Uninterruptible Power Supply)나 내부 배터리(battery)가 약한 배터리 상태임을 표시할 경우 이 메세지가 송신됩니다.

CPI0965

시스템 장치 안에 있는 배터리 전원 장치 피처가 작동하지 않습니다.

시스템 장치에서 배터리를 사용할 수 없거나, 배터리 전원 장치 피처를 위한 배터리 충전이 불가능할 경우 이 메세지가 송신됩니다.

CPI0966

확장 장치 안에 있는 배터리 전원 장치 피처가 작동하지 않습니다.

확장 장치에서 배터리를 사용할 수 없거나, 배터리 전원 장치 피처를 위한 배터리 충전이 불가능할 경우 이 메세지가 송신됩니다.

CPI096B

IASP &1에 대한 지리적 이중복사가 일시중단되었습니다.

지정된 독립 보조 기억장치 풀(ISAP)의 지리적 이중복사가 이중복사를 포함하는 시스템과의 통신 불능으로 인해 일시중단되었음을 알리기 위해 이 메세지가 송신됩니다. 시스템을 검사하여 이것이 예상했던 상황인지 또는 통신 문제를 나타내는지를 판별하십시오.

CPI096C

IASP &1에 대한 지리적 이중복사가 여전히 일시중단된 상태입니다.

지정된 독립 보조 기억장치 풀(ISAP)에 대한 지리적 이중복사가 여전히 일시중단된 상태임을 알리기 위해 이 메세지가 송신됩니다. 이 메세지는 이전에 일시중단된 독립 보조 기억장치 풀에서 지리적 이중복사를 재개하기 위해 아무런 조치를 취하지 않았음을 나타냅니다.

CPI096D

IASP의 복사가 거부되었습니다.

독립 보조 기억장치 풀(IASP)의 기간계 사본이 들어있는 것과 동일한 시스템에 있는 독립 보조 기억장치 풀(IASP)의 이중복사를 구성하려고 시도했음을 나타내기 위해 이 메세지가 송신됩니다. 이중 복사본은 다른 시스템에 있어야 합니다.

CPI096E

디스크 장치 연결이 단절되었습니다.

이 메세지는 기업망 기억장치 서브시스템(ESS)에 대해 예상되는 모든 연결이 보고되지 않았음을 나타내기 위해 송신됩니다. 다음 정보는 디스크 장치를 식별합니다.

- 디스크 일련 번호: &5
- 디스크 유형: &3
- 디스크 모델: &4

이 메세지는 케이블이 단절되었고 구성이 변경되었거나 문제가 있음을 나타낼 수 있습니다. 위의 디스크 장치 정보를 사용하여 하드웨어 서비스 관리자의 서비스 툴에서 해당 장치를 찾고 구성이 변경되었나 그 밖에 다른 문제가 있는지 판별하십시오.

CPI0970

디스크 장치 &1이 작동하지 않습니다.

디스크 장치 &1이 작동을 중단했습니다. 유실된 자료는 없습니다. 다음 정보는 작동하지 않는 디스크 장치를 나타냅니다.

- 디스크 일련 번호: &3
- 디스크 유형: &5
- 디스크 모델: &6
- 디스크 주소: &4
- IOP 자원명: &26
- 장치 제어기 자원명: &27
- 장치 자원명: &28

문제점 분석을 실행하려면 F14 키를 누르십시오.

CPI0988

이중복사 보호가 디스크 장치 &1에서 재개되고 있습니다.

디스크 장치의 이중복사 동기화가 시작되고 디스크 이중복사 보호가 재개되면 이 메세지가 송신됩니다. 디스크 이중복사 보호가 재개되기 전에 시스템이 수행할 단계 중의 하나는 하나의 디스크 장치로부터 다른 장치로 자료를 복사하

여 이들 자료가 동일하도록 해야 합니다. 자료가 복사되는 동안 시스템 성능이 느려지는 것을 알 수 있습니다. 디스크 자료의 복사가 완료되면 메세지 CPI0989가 메세지 대기행렬로 송신되고 디스크 이중복사 보호가 재개됩니다.

CPI0989

이중복사 보호가 디스크 장치 &1에서 재개되었습니다.

디스크 장치의 이중복사 동기화가 정상적으로 완료되면 이 메세지가 송신됩니다. 시스템이 하나의 디스크 장치로부터 다른 디스크 장치로 자료 복사를 완료했습니다. 디스크 이중복사 보호가 재개됩니다.

CPI0998

디스크 장치 &1에서 오류가 발생했습니다.

디스크 장치 &1에서 오류가 발견되면 이 메세지가 송신됩니다. 메세지에는 문제점 분석 수행 실패에 대한 정보가 들어 있지 않습니다.

CPI0999

기억장치 디렉토리 임계값(threshold)에 도달했습니다.

기억장치 디렉토리의 용량에 거의 도달했습니다. 이것은 잠재적으로 심각한 시스템 상태입니다. 시스템이 IPL을 수신할 때까지 시스템에서 이 메세지를 반복합니다.

시스템에서 사용하는 기억장치의 양을 줄여야 합니다. 사용하는 기억장치의 양의 줄이려면 다음과 같이 하십시오.

- 필요없는 시스템에서 오브젝트를 삭제하십시오.
- SAVOBJ(오브젝트 저장) 명령에 STG(*FREE)를 지정하여 온라인에 필요없는 오브젝트를 저장하십시오.

CPI099C

심각한 기억장치 하한값에 도달했습니다.

시스템 보조 기억장치 풀(ASP)에서 사용되는 기억장치의 양이 심각한 하한값에 도달했습니다. 이제 시스템이 QSTGLOWACN 시스템 값(&5)에 지정된 조치를 수행합니다. 가능한 조치로는 다음과 같은 것이 있습니다.

- *MSG -- 시스템이 더 이상 조치를 수행하지 않습니다.
- *CRITMSG -- 시스템이 메세지 CPI099B를 CRITMSGUSR 서비스 속성에 의해 지정된 사용자에게 송신합니다.
- *REGFAC -- 시스템이 QIBM_QWC_QSTGLOWACN 나감점에 대해 등록된 나감 프로그램을 실행하기 위해 작업을 제출합니다.
- *ENDSYS -- 시스템이 제한 상태로 종료합니다.
- *PWRDWNSYS -- 시스템이 즉시 전원을 차단하고 재시작합니다.

기억장치의 사용을 줄이기 위해서는 다음과 같은 조치를 수행할 수 있습니다.

- 사용하지 않는 오브젝트를 삭제합니다.
- STG(*FREE)를 지정하여 오브젝트를 저장합니다.
- 사용하지 않는 구 기록부 버전의 QHST를 저장한 다음 삭제합니다.
- 시스템의 스풀 파일들을 인쇄하거나 삭제합니다.

기억장치 사용을 줄이는 데 실패할 경우 보조 기억장치의 초기화가 필요하거나 사용자 자료가 유실되는 결과를 가져올 수 있습니다. 사용되는 기억장치의 양을 모니터하려면 WRKSYSSTS 명령을 사용하십시오. 기억장치 사용에 관한 정보를 인쇄하려면 PRTDSKINF 명령을 사용하십시오. WRKSYSVAL 명령은 보조 기억장치 하한값(QSTGLOWLMT)과 조치(QSTGLOWACN)를 표시하거나 변경하는 데 사용할 수 있는 명령입니다.

CPI099D

시스템이 기억장치 제한 상태로 시작합니다.

사용할 수 있는 기억장치의 양이 보조 기억장치 하한값보다 낮기 때문에 시스템이 제한 상태로 시작되고 있습니다. 기억장치 사용을 줄이는 데 실패할 경우 보조 기억장치의 초기화가 필요하거나 사용자 자료가 유실되는 결과를 가져올 수 있습니다. 콘솔이 유일한 활동 장치입니다.

기억장치의 사용을 줄이기 위해서는 다음과 같은 조치를 수행할 수 있습니다.

- 사용하지 않는 오브젝트를 삭제합니다.
- STG(*FREE)를 지정하여 오브젝트를 저장합니다.
- 사용하지 않는 구 기록부 버전의 QHST를 저장한 다음 삭제합니다.
- 시스템의 스풀 파일들을 인쇄하거나 삭제합니다.

기억장치 사용을 줄이는 데 실패할 경우 보조 기억장치의 초기화가 필요하거나 사용자 자료가 유실되는 결과를 가져올 수 있습니다. 사용되는 기억장치의 양을 모니터하려면 WRKSYSSTS 명령을 사용하십시오. 기억장치 사용에 관한 정보를 인쇄하려면 PRTDSKINF 명령을 사용하십시오. WRKSYSVAL 명령은 보조 기억장치 하한값(QSTGLOWLMT)과 조치(QSTGLOWACN)를 표시하거나 변경하는 데 사용할 수 있는 명령입니다.

CPI099E

기억장치 하한값 나감 프로그램 오류가 발생했습니다.

나감점 QIBM_QWC_QSTGLOWACN에 대해 사용자 나감 프로그램을 호출하는 동안 오류가 발생했습니다. 이유 코드는 &1입니다. 이유 코드와 그 의미는 다음과 같습니다.

1. 사용자 나감 프로그램을 실행하는 동안 오류가 발생했습니다.
2. 시스템이 사용자 나감 프로그램을 찾지 못했습니다.
3. 시스템이 등록된 사용자 나감 프로그램을 찾지 못했습니다.

4. 사용자 나감 프로그램이 30분 안에 완료되지 못했습니다.
5. 사용자 나감 프로그램을 실행하는 작업이 종료되었습니다.
6. 시스템의 종료로 인해 시스템이 사용자 나감 프로그램을 제출하지 못했습니다.
7. 오류 발생으로 인해 시스템이 사용자 나감 프로그램을 제출하지 못했습니다.
8. 시스템이 사용자 나감 프로그램 작업을 제출했으나 경고가 발행되었습니다.
9. 시스템이 나감점에 대한 등록 정보를 검색하지 못했습니다.
10. 실패한 나감 프로그램의 최대 작업 수를 초과함으로 인해 시스템이 사용자 나감 프로그램을 제출하지 못했습니다.
11. 사용자 나감 프로그램 작업에서 예기치 않은 오류가 발생했습니다.

CPI099F

PWRDWNSYS &1 명령이 진행 중입니다.

1차 파티션의 전원이 차단되면 이 메세지가 2차 파티션으로 송신됩니다.

CPI116A

이중복사 보호가 로드 소스 디스크 장치에서 일시중단되었습니다.

이중복사 보호의 일시중단이 디스크 장치 1에서 발생했습니다. 자료는 유실되지 않았습니다. 디스크 장치 1이 다기능 I/O 프로세서(MFIOP)에 접속되어 있습니다. 가급적 신속하게 디스크 장치를 보수하십시오. 장치 1에 대한 디스크를 보수하기까지는 시스템 전원을 차단하거나 시스템을 IPL하거나 시스템을 IPL하도록 하는 어떤 작업도 수행하지 마십시오.

다음 정보는 일시중단된 장치를 나타냅니다.

- 디스크 일련 번호: &5
- 디스크 유형: &3
- 디스크 모델: &4
- 장치 자원명: &26

오류가 정정되면 시스템이 자동으로 이중복사를 재개합니다.

CPI116B

이중복사 보호(mirrored protection)가 로드 소스 디스크 장치에서 계속해서 일시중단되고 있습니다.

이중복사 보호가 디스크 장치 1에서 일시중단되고 있습니다. 자료는 유실되지 않았습니다. 디스크 장치 1이 다기능 I/O 프로세서(MFIOP)에 접속되어 있습

니다. 가급적 신속하게 디스크 장치를 보수하십시오. 디스크 장치 1을 보수하기 까지는 시스템 전원을 차단하거나 시스템을 IPL하거나 시스템을 IPL하도록 하는 어떤 작업도 수행하지 마십시오.

다음 정보는 일시중단된 장치를 나타냅니다.

- 디스크 일련 번호: &5
- 디스크 유형: &3
- 디스크 모델: &4
- 장치 자원명: &26

이중복사 보호의 일시중단을 발생시킨 실패 원인을 판별하려면 이 메세지 대기 행렬에서 앞에 나열한 메세지들을 참조하십시오. 권장 회복 프로시듀어를 수행하십시오.

CPI116C

압축 디스크 장치 &1이 가득 찼습니다.

압축 디스크 장치 &1이 일시적으로 가득 찼습니다. 기억장치 서브시스템 제어기가 이 상태를 감지했으며 압축 디스크 장치의 자료를 다시 배치하고 있습니다. 디스크 장치에 저장할 수 있는 자료의 양을 최대화하도록 시스템이 작업합니다. 이 작업을 완료하기까지는 일정 시간이 소요됩니다. 기억장치 서브시스템 제어기가 자료의 재배치를 완료하면 시스템이 정상 조작을 재개합니다.

다음 정보는 장치가 가득 찼음을 나타냅니다.

- 디스크 일련 번호: &5
- 디스크 유형: &3
- 디스크 모델: &4
- 장치 자원명: &26

압축 디스크 장치에 기억장치 서브시스템 제어기가 자료를 재배치하기까지 기다리십시오. 시스템 전원을 차단하지 마십시오. 압축 디스크 장치가 가득 찼다는 메세지를 자주 수신하는 경우에는 다음 중 하나 이상을 수행하십시오.

1. SAVOBJ(오브젝트 저장) 명령에 STG(*FREE)를 지정하여 보조 기억장치 풀(ASP)에서 필요없는 오브젝트를 저장하십시오.
2. 보조 기억장치 풀(ASP)에서 필요없는 오브젝트를 삭제하십시오.
3. 폴더를 저장하거나 폴더를 삭제하거나 폴더를 다른 보조 기억장치 풀(ASP)에 복원하여 하나 이상의 폴더를 다른 보조 기억장치 풀(ASP)로 이동하십시오.
4. 보조 기억장치 풀(ASP)에 디스크 장치를 추가하여 기억장치 용량을 늘리십시오. 시스템이 사용자 보조 기억장치 풀(ASP)에서 시스템 보조 기억장치 풀(ASP)로 넘침 자료를 즉시 보내도록 할 수 있습니다. 이렇게 하면 기억

장치 서브시스템 제어기가 압축 디스크 장치에 자료를 재배치하기까지 매번 기다릴 필요가 없습니다. CHGASPA(ASP 속성 변경) 명령을 사용하여 압축 디스크 장치가 가득 찰 때마다 시스템 보조 기억장치 풀(ASP)로 넘침 자료를 즉시 보내도록 압축 회복 정책을 변경하십시오.

CPI1117

라이브러리 &2 안의 손상된 작업 스케줄 &1이 삭제되었습니다.

라이브러리 안의 작업 스케줄이 손상되어 삭제되면 이 메세지가 송신됩니다.

CPI1136

이중복사 보호가 아직도 일시중단 상태입니다.

이 메세지는 이중복사 보호가 한 개 또는 그 이상의 디스크 장치상에서 여전히 일시중단되어 있으면 매시간마다 송신됩니다.

CPI1138

기억장치 넘침이 회복되었습니다.

이 메세지는 이유 &2 때문에 시스템 ASP로 넘친 오브젝트가 &1에 더 이상 없을 때 송신됩니다.

CPI1139

기억장치 넘침 회복에 실패했습니다.

기억장치 넘침을 회복하려는 시도가 실패했을 때 이 메세지가 송신됩니다.

CPI1153

시스템 암호 바이패스 기간이 끝났습니다.

시스템 암호 바이패스 기간의 효력이 끝난 뒤에 암호 없이 시스템을 운영하면 이 메세지가 송신됩니다. 바이패스 기간은 종료되었으므로 시스템 암호를 올바로 제공하지 않으면 다음 IPL을 성공적으로 완료할 수 없습니다.

CPI1154

시스템 암호 바이패스 기간이 &5일 안에 끝날 것입니다.

시스템 암호 바이패스가 선택된 경우에 시스템 암호(이전 IPL 동안)가 입력되지 않거나 입력이 틀리면, 이 메세지가 송신됩니다.

CPI1159

시스템 ID가 &1 설치와 함께 만기될 것입니다.

이 메세지는 시스템 ID가 만기될 때 송신됩니다. IBM 서비스 담당자에게 문의하십시오.

CPI1160

시스템 ID가 만기되었습니다.

이 메세지는 시스템 ID가 만기될 때 송신됩니다. IBM 서비스 담당자에게 문의하십시오.

CPI1161

장치 패리티 보호를 가진 장치 &1이 완전하게 작동하지 않습니다.

장치 &1은 장치 패리티 보호를 가진 디스크 장치 서브시스템의 일부입니다. 장치 &1이 서비스를 필요로 합니다. 자료는 저장되었습니다. 조치를 수행하지 않으면 성능이 저하되거나 기계 검사, 그리고 자료 유실이 발생할 수 있습니다.

CPI1162

장치 패리티 보호를 가진 장치 &1이 완전하게 작동하지 않습니다.

장치 &1은 장치 패리티 보호를 가진 디스크 장치 서브시스템의 일부입니다. 장치 &1은 다음 중 한 가지 이유 때문에 완전하게 작동하지 않습니다.

- 서비스 담당자가 그 장치를 보수하고 있습니다.
- 장치가 작동하지 않습니다. 그러나 문제점 분석을 수행하기 위한 정보가 충분하지 않습니다.

CPI1165

하나 또는 그 이상의 장치 패리티 보호 장치가 여전히 완전하게 작동하지 않습니다.

장치 패리티 보호를 가진 디스크 장치 서브시스템 안의 하나 또는 그 이상의 장치들이 오류 때문에 여전히 완전하게 작동하지 않습니다.

CPI1166

장치 패리티 보호를 가진 장치가 완전하게 작동합니다.

장치 패리티 보호를 제공하는 모든 IOP 서브시스템들을 위한 장치들이 완전하게 작동합니다.

CPI1167

임시 I/O 프로세서 오류가 발생했습니다.

오류 상태가 디스크 장치가 있는 I/O 프로세서에서 발생하였습니다.

CPI1168

디스크 장치 &1에서 오류가 발생했습니다.

디스크 장치 번호 &1이 오류를 발견했습니다. 손상된 오브젝트가 발생할 수 있습니다. 문제가 더 심각해지면 기계 검사가 생길 수 있습니다. 다음은 디스크 장치를 식별합니다.

CPI1169

디스크 장치 &1이 작동하지 않습니다.

디스크 장치 &1이 작동을 중단했습니다. 유실된 자료는 없습니다.

CPI1171

내부 시스템 오브젝트가 회복될 수 없습니다.

시스템에 대한 작업 색인을 가진 내부 시스템 오브젝트가 손상되었습니다. &1 시도 이후 시스템이 오브젝트를 회복할 수 없었습니다.

어떤 회복 조치도 필요없으나 작업 탐색 성능이 영향을 받을 수 있습니다.

CPI1468

시스템 작업 제어 블록표가 허용 한계에 도달했습니다.

시스템 작업표 안의 항목 수가 허용 최대수에 도달하면 시스템이 이 메세지를 송신합니다. 작업표가 완전히 차는 것을 허용하면 작업이 성공적으로 제출되지 않거나 후속 IPL이 성공적으로 이루어지지 않을 수 있습니다.

CPI22AA

QAUDJRN에 대한 감사 레코드를 작성할 수 없습니다.

QAUDJRN 감사 저널에 대해 &3 유형의 감사 레코드를 작성하려고 시도할 때 QSYSAUDR 프로그램의 &2 명령어에서 예기치 않은 &1 예외가 발생했습니다. QAUDENDACN(감사 종료 조치) 시스템 값이 지정한 조치가 수행됩니다.

CPI2209

사용자 프로파일 &1이 손상으로 인해 삭제되었습니다.

사용자 프로파일이 손상되어 삭제되었을 때 이 메세지가 송신됩니다. 이 사용자 프로파일은 삭제되기 전에 자신의 오브젝트를 가졌을 수 있습니다. 이 오브젝트들은 이제 소유자가 없습니다. RCLSTG(기억장치 재생) 명령을 사용하여 이 오브젝트들의 소유권을 QDFTOWN 사용자 프로파일로 이전할 수 있습니다.

CPI2239

QAUDCTL 시스템 값이 &1로 변경되었습니다.

설치 중에 보안 감사 기능을 사용할 수 없음으로 인해 QAUDCTL이 *NONE으로 변경되었습니다. 이제 보안 감사 기능을 사용할 수 있으므로 QAUDCTL 시스템 값이 원래 값으로 변경됩니다.

CPI2283

QAUDCTL 시스템 값이 *NONE으로 변경되었습니다.

이 메세지는 감사 실패로 인해, 시스템이 감사를 종료시킨 뒤 1시간 후에 송신됩니다. 감사를 되돌려 보내거나 감사의 실패 원인을 판별하기 위해 시스템 값 QAUDCTL을 *NONE 이외의 값으로 변경할 수 있습니다.

CPI2284

QAUDCTL 시스템 값이 *NONE으로 변경되었습니다.

감사 실패로 인해 시스템이 감사를 종료시킨 경우 IPL 동안에 이 메세지가 송신됩니다. 감사를 되돌려 보내거나 감사의 실패 원인을 판별하기 위해 시스템 값 QAUDCTL을 *NONE 이외의 값으로 변경할 수 있습니다.

CPI8A13

QDOC 라이브러리가 저장 이력 한계에 거의 도달했습니다.

라이브러리 QDOC 안의 오브젝트 수가 시스템이 한 라이브러리에 저장할 수 있는 오브젝트 수 한계에 접근하고 있을 때 이 메세지가 송신됩니다.

CPI8A14

QDOC 라이브러리가 저장 이력 한계를 초과했습니다.

라이브러리 QDOC 안의 오브젝트 수가 시스템이 한 라이브러리에 지원할 수 있는 오브젝트 수 한계를 초과할 때 이 메세지가 송신됩니다.

CPI9014

장치로부터 수신된 암호가 유효하지 않습니다.

문서 교환 세션에서 잘못된 암호가 수신되었을 때 이 메세지가 송신됩니다. 이것은 시스템에 액세스하기 위해 권한이 없는 시도를 했음을 나타냅니다.

CPI9490

장치 &25의 디스크 오류입니다.

디스크 오류가 검출되었을 때 이 메세지가 송신됩니다.

CPI94A0

장치 &25의 디스크 오류입니다.

디스크 오류가 검출되었을 때 이 메세지가 송신됩니다.

CPI94CE

버스 확장 어댑터(bus expansion adapter), 버스 확장 어댑터(bus extension adapter), 시스템 프로세서 또는 케이블에서 오류가 감지되었습니다.

기본 기억장치에서 장애가 감지되면 이 메세지가 송신됩니다. 시스템 성능이 저하됩니다. 장애가 발생한 카드를 판별하려면 문제점 분석을 수행하십시오.

CPI94CF

기본 기억장치 카드 장애가 감지되었습니다.

기본 기억장치에서 장애가 감지되면 이 메세지가 송신됩니다. 시스템 성능이 저하됩니다. 장애가 발생한 카드를 판별하려면 문제점 분석을 수행하십시오.

CPI94FC

장치 &25의 디스크 오류입니다.

9336 디스크 장치의 부품 중 하나가 오류 임계값을 초과하여 시작에 실패하면 이 메세지가 송신됩니다.

CPI96C0

보호 암호가 확인되지 않습니다.

APPC 사인 온 트랜잭션 프로그램이 사용자 프로파일에 대해 수신한 보호 암호가 틀릴 때 시스템에서 이 메세지를 송신합니다. 이 메세지에는 오류를 식별하는 이유 코드가 포함되어 있습니다. 이유 코드를 검사하여 적절한 조치를 수행하십시오.

CPI96C1

사인 온 요구 GDS 변수가 틀립니다.

APPC 사인 온 트랜잭션 프로그램이 수신한 사인 온 요구 GDS 변수가 틀릴 때 시스템에서 이 메세지를 송신합니다. 리모트 프로그램이 반드시 올바른 사인 온 자료를 송신해야 합니다.

CPI96C2

사용자 암호를 변경할 수 없습니다.

이 메세지는 보안 문제가 발생한 경우에 송신됩니다.

CPI96C3

시스템 호출에 대해 메세지 &4가 리턴되었습니다.

APPC 사인 온 트랜잭션 프로그램이 시스템 호출에 대해 메세지를 리턴할 때 시스템에서 이 메세지를 송신합니다.

CPI96C4

사용자 프로파일에 대한 암호가 틀립니다.

지정된 암호가 틀릴 때 시스템에서 이 메세지를 송신합니다.

CPI96C5

사용자 &4가 없습니다.

수신된 사용자가 시스템에 없을 때 시스템에서 이 메세지를 송신합니다.

CPI96C6

CPI 통신으로의 호출에 대해 리턴 코드 &4가 수신되었습니다.

CPI 통신으로의 호출이 리턴 코드를 송신할 때 시스템에서 이 메세지를 송신합니다. 리턴 코드 설명과 장애 이유를 알아보려면 *Common Programming Interface Communications Reference* 책을 참조하십시오.

CPI96C7

APPC 사인 온 트랜잭션 프로그램에서의 시스템 장애입니다.

예기치 않은 오류를 수신할 때 시스템에서 이 메세지를 송신합니다. 장애 이유를 판별하려면 문제점 분석을 실행하십시오.

CPP0DD9

시스템 프로세서 장애가 감지되었습니다.

시스템 프로세서 또는 시스템 프로세서 캐시에 장애가 발생했을 때 이 메세지가 송신됩니다. 시스템 성능이 저하됩니다.

CPP0DDA

슬롯 9에서 시스템 프로세서 장애가 감지되었습니다.

시스템 프로세서 또는 시스템 프로세서 캐시에 장애가 발생했을 때 이 메세지가 송신됩니다. 시스템 성능이 저하됩니다.

CPP0DDB

슬롯 10에서 시스템 프로세서 장애가 감지되었습니다.

시스템 프로세서 또는 시스템 프로세서 캐시에 장애가 발생했을 때 이 메세지가 송신됩니다. 시스템 성능이 저하됩니다.

CPP0DDC

시스템 프로세서 장애가 감지되었습니다.

시스템 프로세서에서 오류가 검출되면 이 메세지가 송신됩니다. 시스템 성능이 저하됩니다.

CPP0DDD

시스템 프로세서 진단 코드에서 오류를 감지했습니다.

IPL 동안에 시스템 프로세서 진단에 의해 장애가 감지되었으나, 시스템이 여전히 기능하고 있을 때 이 메세지가 수신됩니다. 시스템 성능이 저하됩니다.

CPP0DDE

시스템 프로세서 오류가 감지되었습니다.

시스템 프로세서에서 제어 장애가 감지되면 이 메세지가 송신됩니다. 하드웨어 ECC가 장애를 처리하는 중입니다. 그러나 초기 프로그램 로드(IPL)가 수행되면 제어는 초기화될 수 없고, 시스템은 장애가 발생한 프로세서를 제외한 상태에서 스스로를 재구성합니다.

CPP0DDF

시스템 프로세서가 누락되었습니다.

복수 프로세서 시스템에서 이 프로세서가 누락되면 이 메세지가 송신됩니다.

CPP1604

주의 DASD 실패가 예상됩니다. 즉시 하드웨어 서비스 제공자에게 문의하십시오.

이 메세지는 디스크 장치에서 데이터 유실을 초래하는 회복 불가능한 오류가 발생하려고 할 때 송신됩니다.

CPP29B0

장치 &25에서 회복 임계값이 초과되었습니다.

9337 디스크 장치의 한 부분이 시작에 실패하면 이 메세지가 송신됩니다.

CPP29B8

장치 패리티 보호가 장치 &25에서 일시중단되었습니다.

9337 디스크 배열의 한 부분에 장애가 발생하면 이 메세지가 송신됩니다. RAID 5의 이행 기술에 대한 보호가 디스크 배열에서 일시중단되었습니다.

CPP29B9

전원 보호가 장치 &25에서 일시중단되었습니다.

9337 디스크 배열의 전원 모듈 중 하나에 장애가 발생하면 이 메세지가 송신됩니다. 전원 보호가 디스크 배열에서 일시중단되었습니다.

CPP29BA

장치 &25상의 하드웨어 오류

9337 디스크 배열의 한 부분에 장애가 발생하면 이 메세지가 송신됩니다. 서비스 조치가 필요합니다.

CPP951B

배터리 전원 장치 장애

배터리 전원 장치에 장애가 발생했을 때 이 메세지가 송신됩니다.

CPP9522

배터리 전원 장치 장애

5042 확장 장치나 5040 확장 장치의 배터리 전원 장치에 장애가 발생했을 때 이 메세지가 송신됩니다.

CPP955E

배터리 전원 장치가 설치되지 않았습니다.

9406 시스템 장치 전원 기구의 배터리 전원 장치가 설치되지 않았을 때 이 메세지가 송신됩니다.

CPP9575

9406의 배터리 전원 장치를 교체해야 합니다.

9406 시스템 장치의 배터리 전원 장치에 장애가 발생하여 대체시켜야 할 때 이 메세지가 송신됩니다. 계속 작동되더라도, 충전 방전 주기가 권장 횟수보다 많이 발생했을 가능성이 있습니다.

CPP9576

9406의 배터리 전원 장치를 교체해야 합니다.

9406 시스템 장치의 배터리 전원 장치에 장애가 발생하여 대체시켜야 할 때 이 메세지가 송신됩니다. 계속 작동되더라도 권장 기간보다 더 사용했을 가능성이 있습니다.

CPP9589

배터리 전원 장치 테스트를 완료했습니다.

배터리 전원 장치에 대한 테스트가 완료되고 결과가 기록되었을 때 이 메세지가 송신됩니다.

CPP9616

배터리 전원 장치가 설치되지 않았습니다.

배터리 전원 장치가 5042 확장 장치나 5040 확장 장치 전원 기구에 설치되지 않았을 때 이 메세지가 송신됩니다.

CPP9617

배터리 전원 장치를 교체해야 합니다.

5042 확장 장치나 5040 확장 장치의 배터리 전원 장치를 대체시켜야 할 때 이 메세지가 송신됩니다. 계속 작동되더라도, 충전 방전 주기가 권장 횟수보다 많이 발생했을 가능성이 있습니다.

CPP9618

배터리 전원 장치를 교체해야 합니다.

5042 확장 장치나 5040 확장 장치의 배터리 전원 장치를 대체시켜야 할 때 이 메세지가 송신됩니다. 계속 작동되더라도 권장 기간보다 더 사용했을 가능성이 있습니다.

CPP961F

DC 벌크 모듈 3 장애

9406 시스템 장치의 직류(DC) 벌크 모듈 3에 장애가 발생했을 때 이 메세지가 송신됩니다.

CPP9620

DC 벌크 모듈 2 장애

9406 시스템 장치의 직류(DC) 벌크 모듈 2에 장애가 발생했을 때 이 메세지가 송신됩니다.

CPP9621

DC 벌크 모듈 1 장애

9406 시스템 장치의 직류(DC) 벌크 모듈 1에 장애가 발생했을 때 이 메세지가 송신됩니다.

CPP9622

DC 벌크 모듈 1 장애

5042 확장 장치나 5040 확장 장치의 직류(DC) 벌크 모듈 1에 장애가 발생했을 때 이 메세지가 송신됩니다. 다른 직류(DC) 벌크 모듈도 이러한 장애를 발생시킬 수 있습니다.

CPP9623

DC 벌크 모듈 2 장애

5042 확장 장치나 5040 확장 장치의 직류(DC) 벌크 모듈 2에 장애가 발생했을 때 이 메세지가 송신됩니다. 다른 직류(DC) 벌크 모듈도 이러한 장애를 발생시킬 수 있습니다.

CPP962B

DC 벌크 모듈 3 장애

5042 확장 장치나 5040 확장 장치의 직류(DC) 벌크 모듈 3에 장애가 발생했을 때 이 메세지가 송신됩니다. 다른 직류(DC) 벌크 모듈도 이러한 장애를 발생시킬 수 있습니다.

CPPEA02

주의 즉시 하드웨어 서비스 제공자에게 문의하십시오.

이 메세지는 예외의 내부 분석 결과, 즉시 하드웨어 서비스가 필요하다는 것을 나타낼 때 송신됩니다.

CPPEA04

주의 하드웨어 서비스 제공자에게 문의하십시오.

이 메세지는 예외의 내부 분석 결과, 영구적인 실패로 인해 하드웨어 중복이 유실되었음을 나타낼 때 송신됩니다.

CPPEA05

주의 하드웨어 서비스 제공자에게 문의하십시오.

이 메세지는 예외의 내부 분석 결과, 영구적인 실패로 인해 자료 보호 기능이 유실되었음을 나타낼 때 송신됩니다.

CPPEA12

주의 즉시 하드웨어 서비스 제공자에게 문의하십시오.

이 메세지는 예외의 내부 분석 결과, I/O 카드가 성능이 저하된 레벨에서 작동 중임을 나타낼 때 송신됩니다.

CPPEA13

주의 하드웨어 서비스 제공자에게 문의하십시오. 예외의 내부 분석 자료는 시스템 성능을 유지하기 위해 하드웨어 서비스가 권장됨을 알려줍니다. 하드웨어 서비스 제공자에게 문의하여 I/O 카드의 캐시 배터리 팩을 교환하는 것이 좋습니다. 그렇게 하지 않으면 시스템 성능이 저하될 수 있습니다.

CPPEA26

주의 하드웨어 서비스 제공자에게 문의하십시오.

이 메세지는 예외의 내부 분석 결과, 시스템 가용성을 유지하기 위해 하드웨어 서비스가 권장됨을 나타낼 때 송신됩니다.

CPPEA32

기억장치 서브시스템 구성 오류입니다.

이 메세지는 I/O 카드 구성에 너무 많은 장치나 잘못된 유형의 장치가 사용되었을 때 송신됩니다.

CPPEA38

주의 즉시 하드웨어 서비스 제공자에게 문의하십시오. 시스템 오류가 발생했습니다.

CPPEA39

주의 즉시 하드웨어 서비스 제공자에게 문의하십시오. 치명적인 시스템 오류가 발생했습니다. 시스템이 중복 자원을 사용하여 자동으로 IPL을 수행합니다.

QSYSMSG로부터 메세지를 수신하는 샘플 프로그램

다음은 QSYSMSG 메세지 대기행렬로부터 메세지를 수신하는 샘플 프로그램입니다. 프로그램은 메세지들을 수신하고 메세지 CPF1269를 처리하는 하나의 프로시듀어로 구성됩니다. 메세지 CPF1269의 이유 코드는 2진 형식으로 되어 있습니다. 이유 코드 704 및 705와 비교하려면 이 값이 10진값으로 변환되어야 합니다. 프로시듀어가 ENDMOD 명령을 발행하여 상황을 파악할 때까지 새 작업이 시작되는 것을 막습니다. 그러고 나서 보안 담당자가 검토할 사용자 정의 메세지 대기행렬에 동일한 메세지를 송신합니다. 또한 시스템 오퍼레이터에게 발생한 상황을 알려주는 메세지도 송신합니다. 다른 메세지가 수신되면 이를 시스템 오퍼레이터에게 송신합니다.

다음의 샘플 프로그램을 호출하기 위해 개별적으로 작업을 시작할 수 있습니다. 작업은 계속해서 활동 상태이며 메세지가 도착하기를 기다립니다. ENDJOB 명령을 사용하면 작업을 종료시킬 수 있습니다.

```
/*********************************************************************/
/*                                                                   */
/*  Sample program to receive messages from QSYSMSG                  */
/*                                                                   */
/*********************************************************************/
/*                                                                   */
/*  Program looks for message CPF1269 with a reason code of 704      */
/*  or 705.  If found then notify QSECOFR of the security failure.   */
/*  Otherwise resend the message to QSYSOPR.                         */
/*                                                                   */
/*  The following describes message CPF1269                          */
/*                                                                   */
/*      CPF1269:  Program start request received on communications   */
/*                device &1 was rejected with reason codes &6,; &7;  */
/*                                                                   */
/*      Message data from DSPMSGD CPF1269                            */
/*                                                                   */
/*        Data  type   offset length Description                     */
/*                                                                   */
/*         &1   *CHAR      1     10  Device                          */
```

```
/*         &2   *CHAR     11      8  Mode                          */
/*         &3   *CHAR     19     10  Job - number                   */
/*         &4   *CHAR     29     10  Job - user                     */
/*         &5   *CHAR     39      6  Job - name                     */
/*         &6   *BIN      45      2  Reason code - major           */
/*         &7   *BIN      47      2  Reason code - minor           */
/*         &8   *CHAR     49      8  Remote location name          */
/*         &9   *CHAR     57  *VARY  Unit of work identifier       */
/*                                                                 */
/*****************************************************************/

  PGM

          DCL        &MSGID   *CHAR LEN(  7)
          DCL        &MSGDTA  *CHAR LEN(100)
          DCL        &MSG     *CHAR LEN(132)

          DCL        &DEVICE  *CHAR LEN( 10)
          DCL        &MODE    *CHAR LEN(  8)
          DCL        &RMTLOC  *CHAR LEN(  8)

          MONMSG     CPF0000  EXEC(GOTO PROBLEM)
          /*********************************************************/
          /* Fetch messages from QSYSMSG message queue             */
          /*********************************************************/

LOOP:     RCVMSG     MSGQ(QSYS/QSYSMSG) WAIT(*MAX) MSGID(&MSGID) +
                       MSG(&MSG) MSGDTA(&MSGDTA)

          IF         ((&MSGID *EQ 'CPF1269') /* Start failed msg */ +
            *AND      ((%BIN(&MSGDTA 45 2) *EQ 704)                +
              *OR      (%BIN(&MSGDTA 45 2) *EQ 705)) )             +
          THEN(DO)
            /*******************************************************/
            /* Report security failure to QSECOFR                  */
            /*******************************************************/

            CHGVAR     &DEVICE %SST(&MSGDTA 1 10) /* Extract device */
            CHGVAR     &MODE   %SST(&MSGDTA 11 8) /* Extract mode   */
            CHGVAR     &RMTLOC %SST(&MSGDTA 49 8) /* Get loc name   */

            ENDMOD     RMTLOCNAME(&RMTLOC) MODE(&MODE)

            SNDPGMMSG  MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) +
                         TOMSGQ(QSECOFR)

            SNDPGMMSG  MSG('Device ' *CAT &DEVICE *TCAT ' Mode '  +
                         *CAT &MODE *TCAT ' had security failure, +
                         session max changed to zero')           +
                         TOMSGQ(QSYSOPR)
ENDDO
          ELSE DO
            /*******************************************************/
            /* Other message - Resend to QSYSOPR                   */
            /*******************************************************/

            SNDPGMMSG  MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) +
                         TOMSGQ(QSYSOPR)
```

```
            /* SNDPGMMSG would fail if the message does            */
            /*   not have a MSGID or is not in QCPFMSG             */

            MONMSG     MSGID(CPF0000)  +
                       EXEC(SNDPGMMSG MSG(&MSG) TOMSGQ(QSYSOPR))
ENDDO

          GOTO       LOOP    /* Go fetch next message              */


          /*********************************************************/
          /* Notify QSYSOPR of abnormal end                         */
          /*********************************************************/

PROBLEM:  SNDPGMMSG  MSG('QSYSMSG job has abnormally ended') +
                       TOMSGQ(QSYSOPR)
          MONMSG     CPF0000

          SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
                       MSGDTA('Unexpected error occurred')
          MONMSG     CPF0000

          ENDPGM
```

시스템 응답 리스트의 사용

시스템 응답 리스트를 사용하면 지정된 사전정의 조회 메세지에 시스템이 응답하도록 지정할 수 있습니다. 그러면 표시장치 사용자는 응답을 하지 않아도 됩니다. 단, 조회 메세지에 대해서만 자동으로 응답될 수 있습니다.

시스템 응답 리스트에는 메세지 ID, 선택적 비교 자료, 각 메세지에 대한 응답 값 및 덤프 속성이 들어 있습니다. 시스템 응답 리스트는 이 리스트를 사용하는 작업에 의해 송신되는 사전정의 조회 메세지에 대해서만 적용됩니다. 사용자는 작업이 다음 명령들의 INQMSGRPY(*SYSRPYL) 매개변수에 조회 메세지에 대한 시스템 응답 리스트를 사용하도록 지정합니다.

- BCHJOB(일괄처리 작업)
- SBMJOB(작업 제출)
- CHGJOB(작업 변경)
- CRTJOBD(작업 설명 작성)
- CHGJOBD(작업 설명 변경)

시스템 응답 리스트를 사용하는 작업이 사전정의 조회 메세지를 송신할 때 시스템은 응답 리스트를 오름차순으로 탐색하여 메세지 ID에 일치하는 항목과 응답 메세지의 비교 자료(선택적)를 찾아냅니다. 항목이 발견되면 지정된 응답이 발행되며 사용자가 응답을 입력하지 않아도 됩니다. 그러나 항목이 발견되지 않으면 대화식 작업의 경우에는 표시장치 사용자에게, 일괄처리 작업의 경우에는 시스템 오퍼레이터에게 메세지가 송신됩니다.

시스템 응답 리스트는 다음에 정의된 초기 항목과 함께 제공됩니다.

순번	메세지 **ID**	비교값	응답	덤프
10	CPA0700	*NONE	D	*YES
20	RPG0000	*NONE	D	*YES
30	CBE0000	*NONE	D	*YES
40	PLI0000	*NONE	D	*YES

작업이 시스템 응답 리스트를 사용하여 메세지 CPA0700-CPA0799, RPG0000-RPG9999, CBE0000-CBE9999 또는 PLI0000-PLI9999(프로그램 실패를 나타냄)를 송신하면 이 항목들은 응답 D가 송신되고 작업 덤프가 수행되어야 함을 나타냅니다. 이 항목을 사용하는 시스템인 경우 작업이 시스템 응답 리스트를 사용하도록 지정해야 합니다.

ADDRPYLE(응답 리스트 항목 추가) 명령을 사용하여 시스템 응답 리스트에 다른 조회 메세지를 추가할 수 있습니다. 이 명령에서 순번, 메세지 ID, 선택 가능한 비교 자료, 비교 자료 CCSID, 응답 조치 및 덤프 속성을 지정할 수 있습니다. WRKRPYLE(시스템 응답 리스트 항목에 대한 작업) 명령을 사용하면 ADDRPYLE 명령 기능에 쉽게 액세스할 수 있습니다.

다음은 시스템 응답 리스트에 있는 조회 메세지에 지정할 수 있는 응답 조치들입니다(매개변수 값은 괄호 안에 있음).

- 조회 메세지에 대한 디폴트 응답 송신(*DFT). 이 경우 메세지에 대한 디폴트 응답이 송신됩니다. 이 메세지는 표시되지 않으며 디폴트 처리 프로그램이 호출되지도 않습니다.
- 워크스테이션 사용자나 시스템 오퍼레이터가 메세지에 응답하도록 하는 요구(*RQD). 메세지가 송신된 메세지 대기행렬(대화식 작업의 경우 워크스테이션 메세지 대기행렬, 일괄처리 작업의 경우 QSYSOPR)이 중단 모드에 있으면 메세지가 표시되며 이 때 워크스테이션 사용자는 메세지에 응답해야 합니다. 이 옵션은 시스템 응답 리스트가 사용되지 않는 것처럼 작동합니다.
- 시스템 응답 리스트 항목에 지정된 응답 송신(메세지 응답, 최대 32자). 이 경우 지정된 응답이 메세지에 대한 응답으로 송신됩니다. 이 메세지는 표시되지 않으며 디폴트 처리 프로그램이 호출되지도 않습니다.

다음 명령들은 시스템 응답 리스트에 메세지 RPG1241, RPG1200, CPA4002, CPA5316 및 기타 다른 조회 메세지에 대한 항목을 추가합니다.

- `ADDRPYLE SEQNBR(15) MSGID(RPG1241) RPY(C)`
- `ADDRPYLE SEQNBR(18) MSGID(RPG1200) RPY(*DFT) DUMP(*YES)`
- ```
  ADDRPYLE SEQNBR(22) MSGID(CPA4002) RPY(*RQD) +
  CMPDTA('QSYSPRT')
  ```
  ```

- ADDRPYLE SEQNBR(25) MSGID(CPA4002) RPY(G)
- ADDRPYLE SEQNBR(27) MSGID(CPA5316) RPY(I) DUMP(*NO) +
  CMPDTA('QSYSPRT' 21)
- ADDRPYLE SEQNBR(9999) MSGID(*ANY) RPY(*DFT)

시스템 응답 리스트는 다음과 같이 나타납니다.

순번	메세지 ID	비교값 (b는 공백)	비교 시작 위치	응답	덤프
10	CPA0700		1	D	*YES
15	RPG1241		1	C	*NO
18	RPG1200		1	*DFT	*YES
20	RPG0000		1	D	*YES
22	CPA4002	'QSYSPRT'	1	*RQD	*NO
25	CPA4002		1	G	*NO
27	CPA5316	'QSYSPRT'	21	I	*NO
30	CBE0000		1	D	*YES
40	PLI0000		1	D	*YES
9999	*ANY		1	*DFT	*NO

이 시스템 응답 리스트를 사용하는 작업의 경우 응답 리스트에 추가된 메세지들이 작업에 의해 송신되면 다음과 같은 일이 발생합니다.

- 순번 15의 경우 시스템 응답 리스트를 사용하는 작업에 의해 RPG1241 메세지가 송신될 때마다 응답 C가 송신되며 작업은 덤프되지 않습니다.
- 순번 18의 경우 총칭 메세지 ID가 사용되어, 작업에 의해 RPG1200 조회 메세지가 송신될 때마다, 디폴트 응답이 송신됩니다. 디폴트 응답은 메세지 설명에 지정된 디폴트 응답이거나 시스템 디폴트 응답일 수 있습니다. 디폴트 응답이 송신되기 전에 작업이 덤프됩니다. 추가된 이전 항목이 메세지 RPG1241에 대한 이 항목을 대체합니다.
- 순번 22의 경우 조회 메세지 CPA4002가 QSYSPRT라는 비교 자료와 함께 송신되면 이 메세지가 표시장치 사용자에게 송신되며 사용자는 응답을 해야 합니다.

  비교값을 지정할 때 시작 위치를 지정하지 않으면 비교값은 메세지 안의 대체 자료의 1열에서부터 메세지 자료와 비교됩니다.

  순번 22는 QSYSPRT라는 인쇄 장치명을 테스트합니다. 시작 위치가 다른 하나의 대체 변수를 테스트하는 예에 대해서는 순번 27을 참조하십시오.
- 순번 25의 경우 조회 메세지 CPA4002(프린터 &1의 정렬을 확인하십시오)가 QSYSPRT와 다른 것으로 비교되어 송신되면 응답 G가 송신됩니다. 작업은 덤프되지 않습니다. 순번 22의 경우 인쇄 장치가 QSYSPRT이면, 오퍼레이터는 용지 정렬 메세지에 응답해야 합니다. 순번 25의 경우 다른 장치에 대해 용지 정렬 조회 메세지가 발생하면 G(Go)라는 디폴트 응답이 수행됩니다.

- 순번 27의 경우 조회 메세지 CPA5316이 TESTEDFILETSTLIBRARYQSYSPRT라는 비교 자료와 함께 송신되면 응답 I가 송신됩니다.

  비교값과 시작 위치가 모두 지정되면 비교값은 그 시작 위치에서부터 메세지 자료와 비교됩니다. 이 경우 위치 21이 세 번째 대체 변수의 시작 위치입니다. 메세지 CPA5316에서 처음 네 개의 대체 변수는 다음과 같습니다.

&1	ODP 파일명	*CHAR	10
&2	ODP 라이브러리명	*CHAR	10
&3	ODP 장치명	*CHAR	10
&4	첫 번째 행의 행 번호	*BIN	2

  따라서 순번 27은 응답을 송신하기 전에 QSYSPRT의 ODP 장치명을 테스트합니다.
- 순번 9999의 경우 낮은 순번을 가진 항목과 대응되지 않는 모든 사전정의 조회 메세지에 대해 *ANY라는 메세지 ID가 적용되며 이들 조회 메세지에 대해 디폴트 응답이 송신됩니다. 이 항목이 시스템 응답 리스트에 없으면 표시장치 사용자가 시스템 응답 리스트에 들어 있지 않은 다른 모든 조회 메세지에 대해 응답해야 합니다.

비교값에 *CCHAR 자료가 포함되면 비교가 이루어지기 전에 송신 기능에서 나온 메세지 자료가 시스템 응답 리스트에 저장된 메세지 자료의 CCSID로 변환됩니다. 시스템은 *CCHAR 유형의 자료만 변환합니다. IBM에서는 ADDMSGD(메세지 설명 추가) 명령 *CCHAR 자료에 관한 온라인 정보를 제공합니다. iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

주의:
***CCHAR 자료를 비교 자료로 사용할 때는 다음 제한사항이 적용됩니다.**

- **이러한 유형의 응답 리스트 항목을 추가할 때는 *CCHAR 자료를 다른 자료와 혼합할 수 없습니다.**
- ***CCHAR 자료의 길이를 비교 자료에 포함시킬 수 없습니다.**

***CCHAR 자료를 혼합하거나 *CCHAR 자료의 길이를 포함시키면, 예기치 않은 결과가 발생할 수도 있습니다.**

RMVRPYLE(응답 리스트 항목 제거) 명령으로 항목을 제거할 때까지, 응답이 시스템 응답 리스트상에 계속 남아 있습니다. 응답 리스트 항목의 속성을 변경하려면 CHGRPYLE(응답 리스트 항목 변경) 명령을 사용하고, 현재 응답 리스트에 들어 있는 응답 항목들을 표시하려면 WRKRPYLE(시스템 응답 리스트 항목에 대한 작업) 명령을 사용하십시오.

작업 기록부는 시스템 응답 리스트가 (ADDRPYLE), (CHGRPYLE) 또는 (RMVRPYLE)를 사용하여 갱신될 때 성공적으로 변경되었음을 나타내는 완료 메세지를 수신합니다. 이력 기록부 QHST도 완료 메세지를 수신하여 변경을 기록합니다.

응답 처리

응답 처리 종료 프로그램을 사용하면 응답이 조회 메세지에 송신될 때 사용자 제공 종료 프로그램을 호출할 수 있습니다. 종료 프로그램은 응답 값을 허용, 거부 또는 대체할 수 있습니다. 응답 처리 종료 프로그램에 대한 자세한 내용은 **iSeries Information Center 프로그래밍** 범주의 *API* 섹션에 나와 있는 Message Handling API를 참조하십시오.

메세지 기록

메세지 기록부에는 다음 두 가지 유형이 있습니다.

- 작업 기록부
- 이력 기록부

작업 기록부에는 작업에 대해 입력되는 요구와 관련된 정보가 들어 있습니다. QHST 기록부에는 시스템에서 작업 시작 및 활동 종료의 이력과 같은 시스템 자료가 들어 있습니다.

작업 기록부

각 작업에는 작업에 대해 다음 사항을 포함할 수 있는 관련 작업 기록부가 있습니다.

- 작업 안의 명령
- CL 프로그램이 LOG(*YES) 옵션이나 LOG(*JOB) 옵션으로 작성되었고 CHGJOB(작업 변경)이 LOGCLPGM(*YES) 옵션과 함께 수행된 경우 CL 프로그램의 명령들 (CL 프로그램 명령 기록에 대해 자세히 알려면 70 페이지의 『CL 프로시듀어 명령 기록(logging)』 부분을 참조하십시오.)
- 리퀘스터에게 송신되어 호출 메세지 대기행렬에서 제거되지 않은 모든 메세지와 메세지 도움말

작업이 종료된 후 작업 기록부는 출력 파일 QPJOBLOG나 데이터베이스 파일에 기록(write)될 수 있습니다. 출력 파일 QPJOBLOG로부터 작업 기록부를 인쇄할 수 있습니다. 데이터베이스 파일로부터 데이터베이스 피처를 사용하면 작업 기록부 정보를 조회할 수 있습니다. 성공적으로 수행된 작업에 대해서는 작업 기록부를 기록하지 않도록 지정할 수도 있습니다. 작업 기록부를 작성하지 않는 것에 대한 내용은 이 장의 뒷부분에서 설명합니다.

QMHCTLJL API를 사용하여 데이터베이스 파일에 작업 기록부를 쓸 수 있습니다. QMHCTLJL API에 대한 자세한 정보는 **iSeries Information Center 프로그래밍** 범주의 *API* 섹션을 참조하십시오. 작업 기록부를 데이터베이스 파일로 지정하면 시스템이 하나 이상의 파일을 생성할 수 있습니다. 1차 파일에는 메세지 ID, 메세지 심각도, 메세지 유형 및 메세지 자료와 같은 메세지에 대한 필수 정보가 들어 있습니다. 2차 파일에는 메세지 텍스트의 인쇄 이미지가 들어 있습니다. QMHCTLJL API에 있는 매개변

수가 2차 파일의 선택적 생성을 제어합니다. 시스템에서 데이터베이스와 조회 피처를 사용하여 두 가지 파일을 모두 외부적으로 서술하고 처리할 수 있습니다. 1차 및 2차 파일의 형식에 대해 알려면 471 페이지의 부록 B『작업 기록부 출력 파일』부분을 참조하십시오.

사용자가 작업 기록부에 기록되는 정보를 제어할 수 있습니다. 이렇게 하려면 CRTJOBD(작업 설명 작성) 명령에서 LOG 매개변수를 지정해야 합니다. CHGJOB(작업 변경) 명령이나 CHGJOBD(작업 설명 변경) 명령을 사용하여 이 레벨을 변경할 수 있습니다. LOG 매개변수는 세 개의 값(메세지 레벨, 메세지 심각도 및 메세지 텍스트 레벨)으로 구성됩니다. 이들 명령에 대한 자세한 정보는 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

첫 번째 값인 메세지 레벨은 다음과 같습니다.

레벨 **설명**

0 어떤 자료도 기록되지 않습니다.

1 지정된 메세지 심각도 이상의 심각도로서 작업의 외부 메세지 대기행렬에 송신된 메세지들이 모두 기록됩니다. 이 유형의 메세지들은 작업이 시작되는 시점, 종료되는 시점, 그리고 완료 시 작업의 상태 등을 표시합니다.

2 다음 정보가 기록됩니다.

- 레벨 1 기록 정보
- 지정된 심각도 이상의 심각도를 상위 레벨 메세지 결과를 갖는 모든 요구. 이 요구가 기록되어 있으면, 이와 관련된 모든 메세지도 기록되어 있습니다.

3 다음 정보가 기록됩니다.

- 기록 레벨 1과 2 정보
- 모든 요구.
- CL 프로그램 명령 기록 작업 속성 및 CL 프로그램의 기록 속성에 의해 허용된 경우 CL 프로그램에 의해 실행된 명령.

4 다음 정보가 기록됩니다.

- 추적 메세지를 포함하여 지정된 심각도 이상의 심각도 코드를 갖는 모든 요구 및 메세지.
- CL 프로그램 명령 기록 작업 속성 및 CL 프로그램의 기록 속성에 의해 허용된 경우 CL 프로그램에 의해 실행된 명령.

  주: 상위 레벨 메세지는 요구를 수신하는 프로그램의 프로그램 메세지 대기행렬로 송신된 메세지입니다. 예를 들어, QCMD는 요구를 수신하는 IBM 제공의 요구 처리 프로그램입니다.

두 번째 값인 메세지 심각도는 오류 메세지를 작업 기록부에 기록시키는 로그 레벨을 갖는 심각도 레벨을 지정합니다. 0부터 99까지의 값이 허용됩니다.

LOG 매개변수 안의 세 번째 값인 메세지 텍스트 레벨은 작업 기록부에 기록되는 메세지 텍스트의 레벨을 지정합니다. 그 값은 다음과 같습니다.

***SAME**
메세지 텍스트 레벨에 대한 현재 값은 변하지 않습니다.

***MSG**
메세지 텍스트만이 작업 기록부에 기록됨(메세지 도움말은 포함되지 않음).

***SECLVL**
메세지와 메세지 도움말(원인 및 회복)이 작업 기록부에 기록됩니다.

요구 처리 프로그램이 각각의 새로운 요구를 수신하기 전에 메세지 필터링이 발생합니다. **메세지 필터링**은 작업에 대해 설정된 메세지 기록 레벨에 기초하여 작업 기록부로부터 메세지를 제거하는 프로세스입니다.

모든 CL 명령이 프로그램 안에서 호출된 후에는 필터링이 발생하지 않습니다. 따라서 CL 프로그램이 대화식으로 수행되거나 일괄처리로 제출되면 프로그램이 요구 프로세서가 아니기 때문에 프로그램이 종료한 후에야 필터링이 실행됩니다.

주: *NOLIST는 정상 종료된 작업에 대해 어떤 작업 기록부도 스풀되지 않도록 지정하므로 기록부 레벨 0을 지정하여 이 기록부에서 메세지를 제거하는 것은 일괄처리 작업인 경우 시스템 자원의 낭비입니다.

다음의 예는 기록 레벨이 작업 메세지 대기행렬에 저장된 정보에 어떤 영향을 미치는지 즉, 작업 기록부가 어떻게 기록되는지를 보여줍니다. 이 예는 또한 명령이 대화식으로 수행될 때 각 명령 다음에 필터링이 발생함을 보여줍니다.

주: 다음 예에는 상위 레벨 및 상세 메세지 기록 레벨이 둘 다 들어 있습니다. 상위 레벨 메세지는 *메세지*로 식별되며 상세 메세지는 *상세 메세지*로 식별됩니다.

1. CHGJOB 명령은 기록 레벨 2와 메세지 심각도 50을 지정하여 그에 해당되는 메세지들만이 작업 기록부(*MSG)에 기록되도록 지정합니다.

```
                              명령 입력                    SYSTEM1
                                              요구 레벨:       1
이전 명령 및 메세지:
    > CHGJOB LOG(2 50 *MSG)
```

2. PGMA는 심각도 코드 20, 50 및 60인 세 개의 정보용 메세지를 자체 호출 메세지 대기행렬과 지정된 호출(*PRV) 이전에 호출된 프로그램의 호출 메세지 대기행렬로 송신합니다. PGMA가 자신의 호출 메세지 대기행렬로 송신하는 메세지를 상세 메세지라고 합니다. **상세 메세지**는 하위 레벨 프로그램 호출의 호출 메세지 대기행렬로 송신되는 메세지들입니다.

PGMB는 심각도 코드가 각각 40과 50인 두 개의 정보용 메세지를 자신의 호출 메세지 대기행렬로 송신합니다. 이들은 상세 메세지입니다. PGMB는 또한 심각도 코드가 10인 정보용 메세지 한 개를 *PRV로 송신합니다.

CHGJOB 명령이 더 이상 화면상에 표시되지 않음을 주의하십시오. 기록 레벨 2에 따라, 지정된 심각도 이상의 심각도를 가진 메세지가 발행된 요구만이 작업 기록부에 저장되며, 이 요구에 대해 어떤 메세지도 발행되지 않습니다. 이와 같은 메세지가 발행되었으면, 발행된 모든 상세 메세지들은 작업 기록부에 저장되고 F10 키를 눌러 표시될 수 있습니다.

```
                                명령 입력                          SYSTEM1
                                                      요구 레벨:       1
이전 명령 및 메세지:
     > CALL PGMA
       Message sev 20 - PGMA
       Message sev 50 - PGMA
       Message sev 60 - PGMA
   > CALL PGMB
     Message sev 10 - PGMB









        맨 아래
명령을 입력한 후 Enter 키를 누르십시오.
===> ____________________________________________________________________
____________________________________________________________________
____________________________________________________________________
____________________________________________________________________
 F3=나감    F4=프롬트    F9=검색       F10=상세 메세지 포함
 F11=전체 표시          F12=취소      F13=정보 지원              F24=추가 키
```

3. '명령 입력' 화면에서 F10(상세 메세지 포함) 키를 누르면 요구와 관련된 모든 메세지가 표시됩니다. 상세 메세지를 제외시키려면 F10 키를 다시 누르십시오.

```
                                명령 입력                          SYSTEM1
                                                      요구 레벨:       1
모든 이전 명령 및 메세지:
     > CALL PGMA
     Detailed message sev  20 - PGMA
     Detailed message sev  50 - PGMA
     Detailed message sev  60 - PGMA
       Message sev 20 - PGMA
       Message sev 50 - PGMA
       Message sev 60 - PGMA
   > CALL PGMB
     Detailed message sev  40 - PGMB
     Detailed message sev  50 - PGMB
     Message sev 10 - PGMB

        맨 아래
명령을 입력한 후 Enter 키를 누르십시오.
===> ____________________________________________________________________
____________________________________________________________________
____________________________________________________________________
____________________________________________________________________
 F3=나감    F4=프롬트    F9=검색       F10=상세 메세지 제외
 F11=전체 표시          F12=취소      F13=정보 지원              F24=추가 키
```

4. 다른 명령이 입력되면(이 예에서의 CHGJOB 같은), CALL PGMB 명령과 모든 메세지(상세 메세지 포함)가 제거됩니다. 이는 이 요구와 관련된 상위 레벨 메세지의 심각도 코드가 CHGJOB 명령에 지정된 심각도 코드보다 작기 때문입니다. 요구에 대해 발행된 고급 레벨 메세지 중 적어도 하나에는 지정된 심각도 코드 이상의 심각도 코드가 있기 때문에 CALL PGMA 명령 및 관련 메세지가 그대로 남습니다.

   다음 화면에서 CHGJOB 명령은 기록 레벨 3, 메세지 심각도 40을 지정하며, 메세지의 첫 번째 및 두 번째 레벨 텍스트가 작업 기록부에 기록됩니다. 기록 레벨 3이 모든 요구를 기록하기 때문에 다른 명령이 입력될 때 CHGJOB 명령은 화면상에 그대로 남습니다.

   PGMC는 각각 심각도 코드 30과 40을 가진 두 개의 메세지를, 지정된 호출 이전에 호출되는 프로그램의 호출 메세지 대기행렬(*PRV)로 송신합니다.

   PGMD는 심각도 10인 메세지를 *PRV로 송신합니다.

```
                                명령 입력                          SYSTEM1
                                                      요구 레벨:       1
이전 명령 및 메세지:
     > CALL PGMA
       Message sev 20 - PGMA
       Message sev 50 - PGMA
       Message sev 60 - PGMA
     > CHGJOB LOG(3 40 *SECLVL)
     > CALL PGMC
       Message sev 30 - PGMC
       Message sev 40 - PGMC
     > CALL PGMD
       Message sev 10 - PGMD


        맨 아래
명령을 입력한 후 Enter 키를 누르십시오.
===> ____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
 F3=나감    F4=프롬트    F9=검색        F10=상세 메세지 포함
 F11=전체 표시           F12=취소       F13=정보 지원                F24=추가 키
```

5. CALL PGMD 명령이 입력된 후 다른 명령이 입력되면 CALL PGMD 명령은 화면상에 남아 있으나 이와 관련된 메세지는 삭제됩니다. 이는 해당 심각도 코드가 CHGJOB 명령상의 LOG 매개변수에 지정된 심각도 코드보다 작기 때문입니다.

   명령 SIGNOFF *LIST는 작업 기록부를 인쇄하기 위해 입력됩니다.

```
                                명령 입력                          SYSTEM1
                                                        요구 레벨:       1
이전 명령 및 메세지:
     > CHGJOB LOG(3 40 *SECLVL)
     > CALL PGMC
       Message sev 30 - PGMC
       Message sev 40 - PGMC
     > CALL PGMD
     > CALL PGME




        맨 아래
명령을 입력한 후 Enter 키를 누르십시오.
===> SIGNOFF *LIST_____________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
F3=나감    F4=프롬트    F9=검색       F10=상세 메세지 포함
F11=전체 표시          F12=취소      F13=정보 지원              F24=추가 키
```

작업 기록부에는 '명령 입력' 화면에 남아 있는 모든 요구 및 모든 메세지가 들어 있습니다. 그 외에 작업 기록부에는 CHGJOB 명령에 의해 지정된 각 메세지와 관련된 메세지 도움말이 들어 있습니다. 작업 기록부에는 작업하는 동안 발행된 메세지의 메세지 도움말(두 번째 CHGJOB 명령이 입력된 후에 발행된 메세지에 대한 메세지 도움말이 아님)이 들어 있다는 점에 유의하십시오.

```
 5722SS1 V5R3M0 040905                    Job Log                          SYSAS727 01/16/04 07:13:35          Page    1
  Job name . . . . . . . . . . :   QPADEV000C      User  . . . . . . :   JOHNDOE      Number . . . . . . . . . . . :   038518
  Job description  . . . . . . :   QDFTJOBD        Library . . . . . :   QGPL
MSGID      TYPE                   SEV  DATE      TIME            FROM PGM     LIBRARY     INST     TO PGM      LIBRARY     INST
CPF1124    Information            00   01/16/04  07:13:19.570504  QWTPIIPP     QSYS        0613     *EXT                    *N
  Message . . . . :   Job 038518/JOHNDOE/QPADEV000C started on 01/16/04 at
    07:13:19 in subsystem QINTER in QSYS. Job entered system on 01/16/04 at
    07:13:19.
*NONE      Request                     01/16/04  07:13:24.318144  QMHGSD       QSYS        0010     QCMD        QSYS        0178
  Message . . . . :  -CALL PGMA
MSG1001    Information            20   01/16/04  07:13:24.361064  PGMA         JOHNDOE     0029     PGMA        JOHNDOE     0029
  From User . . . . . . . . . :   MARYJANE
  Message . . . . :   Detailed message sev 20 - PGMA
  MSG1001 second level text - PGMA
MSG1002    Information            50   01/16/04  07:13:24.361416  PGMA         JOHNDOE     0032     PGMA        JOHNDOE     0032
  Message . . . . :   Detailed message sev 50 - PGMA
  MSG1002 second level text - PGMA
MSG1003    Information            60   01/16/04  07:13:24.361592  PGMA         JOHNDOE     0036     PGMA        JOHNDOE     0036
  Message . . . . :   Detailed message sev 60 - PGMA
  MSG1003 second level text - PGMA
MSG1004    Information            20   01/16/04  07:13:24.361776  PGMA         JOHNDOE     003A     QCMD        QSYS        01A6
  Message . . . . :   Message sev 20 - PGMA
  MSG1004 second level text - PGMA
MSG1005    Information            50   01/16/04  07:13:24.362192  PGMA         JOHNDOE     0043     QCMD        QSYS        01A6
  From User . . . . . . . . . :   MARYJANE
  Message . . . . :   Message sev 50 - PGMA
  MSG1005 second level text - PGMA
MSG1006    Information            60   01/16/04  07:13:24.362552  PGMA         JOHNDOE     004C     QCMD        QSYS        01A6
  Message . . . . :   Message sev 60 - PGMA
  MSG1006 second level text - PGMA
*NONE      Request                     01/16/04  07:13:24.370240  QMHGSD       QSYS        0018     QCMD        QSYS        0178
  Message . . . . :  -CHGJOB LOG(3 40 *SECLVL)
*NONE      Request                     01/16/04  07:13:24.370672  QMHGSD       QSYS        001C     QCMD        QSYS        0178
  Message . . . . :  -CALL PGMC
MSG100F    Information            30   01/16/04  07:13:24.379256  PGMC         JOHNDOE     *STMT    QCMD        QSYS        01A6
  From User . . . . . . . . . :   MARYJANE
  From module . . . . . . . . :   PGMC
  From procedure  . . . . . . :   PGMC
  Statement . . . . . . . . . :   8000
  Message . . . . :   Message sev 30 - PGMC
  MSG100F second level text - PGMC
MSG1010    Information            40   01/16/04  07:13:24.379608  PGMC         JOHNDOE     *STMT    QCMD        QSYS        01A6
  From module . . . . . . . . :   PGMC
  From procedure  . . . . . . :   PGMC
  Statement . . . . . . . . . :   8200
 5722SS1 V5R3M0 030905                    Job Log                          LPAR3TLM 01/16/04 07:13:35          Page    2
  Job name . . . . . . . . . . :   QPADEV000C      User  . . . . . . :   JOHNDOE      Number . . . . . . . . . . . :   038518
  Job description  . . . . . . :   QDFTJOBD        Library . . . . . :   QGPL
MSGID      TYPE                   SEV  DATE      TIME            FROM PGM     LIBRARY     INST     TO PGM      LIBRARY     INST
  Message . . . . :   Message sev 40 - PGMD
```

```
                                MSG1010 second level text - PGMC
*NONE      Request                 01/16/04  07:13:24.379928  QMHGSD       QSYS        0020     QCMD        QSYS        0178
                                Message . . . . :  -CALL PGMD
*NONE      Request                 01/16/04  07:13:24.383568  QMHGSD       QSYS        0024     QCMD        QSYS        0178
                                Message . . . . :  -CALL PGME
*NONE      Request                 01/16/04  07:13:35.166408  QMHGSD       QSYS        073E     QCMD        QSYS        0178
                                Message . . . . :  -signoff *list
CPF1164    Completion           00   01/16/04  07:13:35.173496  QWTMCEOJ     QSYS        00BD     *EXT                    *N
                                Message . . . . :   Job 038518/JOHNDOE/QPADEV000C ended on 01/16/04 at
                                  07:13:35; 1 seconds used; end code 0 .
                                Cause . . . . . :   Job 038518/JOHNDOE/QPADEV000C completed on 01/16/04 at
                                  07:13:35 after it used 1 seconds processing unit time.  The job had ending
                                  code 0. The job ended after 1 routing steps with a secondary ending code of
                                  0.  The job ending codes and their meanings are as follows:  0 - The job
                                  completed normally. 10 - The job completed normally during controlled ending
                                  or controlled subsystem ending. 20 - The job exceeded end severity (ENDSEV
                                  job attribute). 30 - The job ended abnormally. 40 - The job ended before
                                  becoming active. 50 - The job ended while the job was active. 60 - The
                                  subsystem ended abnormally while the job was active. 70 - The system ended
                                  abnormally while the job was active. 80 - The job ended (ENDJOBABN command).
                                  90 - The job was forced to end after the time limit ended (ENDJOBABN
                                  command). Recovery  . . . :   For more information, see the Work Management
                                  topic in the Information Center, http://www.iseries.ibm.com/infocenter.
```

인쇄된 작업 기록부에서 각 페이지의 최상단에 있는 머리말은 작업 기록부가 적용되는 작업과 각 항목의 특성을 나타냅니다.

- 제품 ID, 버전 및 오퍼레이팅 시스템의 날짜
- 시스템명
- 작업 기록부가 인쇄된 날짜와 시간
- 완전히 규정된 작업 이름(작업명, 사용자명 및 작업 번호).
- 작업을 시작하는 데 사용되는 작업 설명의 이름
- 섹션 번호. 이 번호는 작업 기록부가 랩핑되고 *PRTWRAP가 작업 메세지 대기행렬 전체 조치에 대해 지정되어 작업 기록부가 여러 섹션에 인쇄된 경우에 인쇄됩니다.
- 각 메세지 항목의 첫 번째 행에 있는 정보 머리말

작업 기록부의 각 메세지 항목에 다음 정보가 인쇄됩니다.

- 각 메세지의 첫 번째 행에는 다음 정보가 포함됩니다.
  - 메세지 ID 또는 *NONE.
  - 메세지 유형.
  - 메세지 심각도. 이것은 요구 메세지의 경우 공백입니다.
  - 각각의 메세지가 송신된 날짜와 시간.
  - 프로그램명, 라이브러리명 및 메세지를 송신한 프로그램의 명령어 번호.
  - 프로그램명, 라이브러리명 및 메세지를 수신한 프로그램의 명령어 번호. *EXT는 메세지가 작업의 외부 메세지 대기행렬에 송신되었음을 나타냅니다.
- 규정된 작업명에서 사용자로 식별되지 않은 사용자가 메세지를 송신한 경우 메세지를 송신한 사용자명이 별도의 행에 인쇄됩니다. 이것은 다음 상황 중 하나를 의미할 수 있습니다.
  - 이 메세지가 작업이 다른 사용자 프로파일에서 실행 중일 때 송신되었습니다. 위에 표시된 샘플 작업 기록부에서 작업이 다른 사용자 프로파일에서 실행 중일 때 MSG1001, MSG1005 및 MSG100F 메세지가 송신되었습니다.

  - 조회 메세지에 다른 사용자가 응답했습니다.
  - 일괄처리 작업을 실행 중인 사용자 프로파일이 아닌 다른 사용자가 일괄처리 작업을 제출했습니다. 이 경우 제출한 사용자의 이름이 각 요구 메세지에 포함되어 있습니다.
  - 다른 사용자가 작업 속성을 변경했습니다.
- 송신자가 ILE 프로시듀어이면 모듈, 프로시듀어 및 명령문 번호를 식별하기 위해 추가 행이 인쇄됩니다. 자세한 정보는 『프로그램 또는 프로시듀어의 송/수신』 부분을 참조하십시오. MSG100F 메세지의 경우 위에 표시된 샘플 작업 기록부에서 PGMC는 ILE 프로그램입니다.
- 작업이 다중 스레드 작업이고 두 개 이상의 스레드로부터 메세지가 송신된 경우 각 메세지에 대해 스레드 ID가 인쇄됩니다.
- 메세지는 하나 이상의 행에 인쇄됩니다.
- 기록 레벨이 두 번째 레벨 텍스트가 포함되어야 함을 나타내는 경우 메세지 아래의 후속 행에 두 번째 레벨 텍스트가 표시됩니다.

프로그램 또는 프로시듀어의 송/수신

송신자나 리시버가 ILE 프로시듀어이면, 메세지 항목에 프로시듀어의 전체 이름(프로시듀어명, 모듈명, ILE 프로그램명)이 들어 있습니다. 송신자나 리시버가 최초 프로그램 모델(OPM) 프로그램이면, OPM 프로그램명만이 표시됩니다.

송신자나 리시버가 OPM 프로그램이면, 해당 지시어 번호가 지시어 번호를 요구합니다. 그와 같은 번호는 오직 하나 있습니다. 송신자나 리시버가 ILE 프로시듀어이면, 지시어 번호는 MI 지시어 번호보다는 고급 언어 명령문 번호를 나타냅니다. ILE 프로시듀어가 최적화(최대 효율)되었으면, 최대 세 개의 번호까지 가능합니다. 최적화된 프로시듀어에 대해 하나의 명령문 번호를 판별하는 것이 항상 가능하지는 않습니다. 둘 이상의 번호가 주어지면, 각각의 번호는 메세지가 송신될 때 프로시듀어가 있던 위치를 나타냅니다. 어떤 번호도 판별될 수 없는 경우가 있습니다. 그 경우에는 *N이 번호보다는 메세지에 표시됩니다.

기록 레벨은 앞의 예에서처럼 일괄처리 작업 기록부에 영향을 미칩니다. 작업이 APPC를 사용하는 경우 APPC에 대한 작업 ID의 단위를 표시하는 행이 머리말에 포함됩니다.

추가의 메세지 필터링

QMHCTLJL API를 사용하여 작업 기록부의 방향이 데이터베이스 파일로 정해지면, 추가로 메세지 필터링 기능을 지정할 수 있습니다. 이 API를 통해 지정된 메세지 필터링은 작업이 종료하고 메세지에 대한 레코드가 파일에 기록될 때 적용됩니다. 이때까지 필터링된 메세지가 표시됩니다. 그러므로 작업이 실행 중일 때도 메세지들을 볼 수 있습

니다. 작업 기록부가 기록될 때 필터링된 메세지는 메세지를 위해 파일에 기록된 레코드를 전혀 갖지 않습니다. 따라서 작업이 실행 중인 동안 메세지가 나타나더라도 메세지는 생성된 최종 파일에는 없습니다.

작업 기록부 표시

작업 기록부를 표시하는 방법은 작업 상태에 따라 다릅니다.

- 작업이 종료되었으나 작업 기록부가 아직 인쇄되지 않은 경우에는 다음과 같이 DSPSPLF(스풀 파일 표시) 명령을 사용하십시오.

  ```
  DSPSPLF FILE(QPJOBLOG) JOB(001293/FRED/WS3)
  ```

  표시장치 WS3에서 사용자 FRED와 관련된 작업 번호 001293의 작업 기록부를 표시합니다.
- 작업이 아직 활동 중(일괄처리 작업이든 대화식 작업이든)이거나, 작업 대기행렬 안에 있으면서 아직 시작하지 않은 경우에는 DSPJOBLOG(작업 기록부 표시) 명령을 사용합니다. 예를 들면, 표시장치 WS1의 사용자 JSMITH에 대한 대화식 작업의 작업 기록부를 표시하려면 다음을 입력하십시오.

  ```
  DSPJOBLOG JOB(nnnnnn/JSMITH/WS1)
  ```

  여기서, nnnnnn은 작업 번호입니다.

사용자 자신의 대화식 작업의 작업 기록부를 표시하려면 다음 중 하나를 수행하십시오.

- 다음의 명령을 입력하십시오.

  ```
  DSPJOBLOG
  ```
- WRKJOB 명령을 입력한 후 '작업에 대한 작업' 화면에서 옵션 10(작업 기록부 표시)을 선택하십시오.
- '명령 입력' 화면에서 F10(상세 메세지 포함) 키를 누르십시오(이 키는 작업 기록부에 표시된 메세지를 표시함).
- 사용자의 표시장치에 입력 금지 표시가 계속 나타나면 다음과 같이 하십시오.
  1. 시스템 요구 키를 누른 후 Enter 키를 누르십시오.
  2. 시스템 요구 메뉴에서 옵션 3(현재 작업 표시)을 선택하십시오.
  3. 작업 표시 메뉴에서 옵션 10(활동 중이거나 작업 대기행렬상에 있는 작업 기록부 표시)을 선택하십시오.
  4. '작업 기록부' 화면에 DSPJOB이 처리 요구로서 나타납니다. F10(상세 메세지 표시) 키를 누르십시오.
  5. '모든 메세지 표시' 화면에서 시스템 요구 키를 누르기 전에 위로 이동 키를 눌러 수신된 메세지를 참조하십시오.
- SIGNOFF 명령에 LOG(*LIST)를 지정하여 워크스테이션에서 사인 오프하십시오.

DSPJOBLOG(작업 기록부 표시) 명령을 사용하면 작업 기록부 화면을 볼 수 있습니다. 이 화면에는 다음과 같은 특수 기호와 함께 프로그램명이 나옵니다.

>> 수행 중인 명령 또는 수행될 다음 명령. 예를 들어, 프로그램이 호출되면 프로그램에 대한 호출이 표시됩니다.

> 명령 처리가 완료되었음.

. . 명령이 아직 처리되지 않았음.

? 응답 메세지. 이 기호는 응답을 필요로 하는 메세지와 응답된 메세지를 모두 표시합니다.

'작업 기록부' 화면에서 다음을 수행할 수 있습니다.

- F10 키를 눌러서 상세 메세지를 표시할 수 있습니다. 이 화면은 LOG가 활성화되어 있는 HLL 프로그램이나 CL 프로그램 또는 프로시듀어 안에서 실행된 명령이나 조작을 보여줍니다.
- 커서 이동 키를 사용하면 작업 기록부의 맨 아래로 갈 수 있습니다. 작업 기록부의 맨아래로 빨리 가려면 F18(맨 아래) 키를 누르십시오. F18 키를 누른 후 수행 중인 명령을 참조하려면 화면을 위로 이동해야 합니다.
- 커서 이동 키를 사용하면 작업 기록부의 맨 위로 갈 수 있습니다. 작업 기록부의 맨 위로 빨리 가려면 F17(맨 위) 키를 누르십시오.

작업을 인쇄하거나 표시하는 대신 데이터베이스 파일로 지시하기 위해 DSPJOBLOG 명령을 사용할 수 있습니다. 두 개의 옵션이 사용될 수 있습니다. 첫 번째 옵션에서는 명령에 파일 및 멤버명을 지정할 수 있습니다. 이 옵션에서는 1차 작업 기록부 정보가 명령에서 지정된 데이터베이스 파일에 기록됩니다. 두 번째 옵션에서는 사전에 실행된 QMHCTLJL API에 제공된 정보와 함께 명령을 사용할 수 있습니다. 이 옵션에서는 작업 기록부가 API 호출에서 지정된 파일(들)로 기록됩니다. 이 옵션을 사용하여 메세지가 파일에 기록될 때 1차 및 2차 파일을 모두 생성하고 메세지 필터링 기능을 수행할 수 있습니다. DSPJOBLOG 명령이 완료될 때 이들 두 개의 옵션을 사용하면 출력이 표시되지도 않고 인쇄에 모두 사용할 수 있는 스풀 파일도 없어집니다.

작업 기록부의 생성 금지

BCHJOB(일괄처리 작업), SBMJOB(작업 제출), CHGJOB(작업 변경), CRTJOBD(작업 설명 작성) 또는 CHGJOBD(작업 설명 변경) 명령에서 LOG 매개변수의 메세지 텍스트 레벨 값에 *NOLIST를 지정하면 일괄처리 작업이 완료될 때 작업이 기록되지 않도록 할 수 있습니다. LOG 매개변수의 메세지 레벨 값을 *NOLIST로 지정하면 작업 종료 시 작업 종료 코드가 20 이상인 작업에 대해서만 작업 기록부가 만들어집니다. 작업 종료 코드가 20 미만이면, 작업 기록부가 만들어지지 않습니다.

대화식 작업인 경우 SIGNOFF 명령상의 LOG 매개변수에 지정된 값이 작업에 대해 지정된 LOG 매개변수 값보다 우선합니다.

작업 기록부의 고려사항

다음 사항들이 작업 기록부의 사용에 적용됩니다.

- 시스템상의 모든 작업에 대한 출력 대기행렬을 변경하려면 CHGPRTF(프린터 파일 변경) 명령상의 OUTQ 또는 DEV 매개변수를 사용하여 파일 QSYS/QPJOBLOG를 변경하십시오. 다음은 각 매개변수를 사용한 두 가지 예입니다.

  ```
  CHGPRTF FILE(QSYS/QPJOBLOG)
          DEV (USRPRT)
  ```

  또는

  ```
  CHGPRTF FILE(QSYS/QPJOBLOG)
          OUTQ(USROUTQ)
  ```

- 출력 대기행렬 QEZJOBLOG를 사용하는 QPJOBLOG 프린터 파일을 변경하려면 조작 지원 클린업 기능을 사용하십시오. 작업 기록부의 자동 클린업 기능을 사용하려면 프린터 파일의 방향이 이 출력 대기행렬로 정해져야 합니다. 운영 지원 클린업 기능에 대한 자세한 정보는 iSeries Information Center의 **시스템 관리** 범주에서 *Getting Started with iSeries*를 참조하십시오.
- 작업의 작업 기록부가 기록되는 출력 대기행렬을 지정하려면 파일 QPJOBLOG에 OUTQ(*JOB)가 지정되었는지 확인하십시오. BCHJOB, CRTJOBD, CHGJOBD 또는 CHGJOB 명령 중 어디에서나 OUTQ 매개변수를 사용할 수 있습니다. 예를 들면, 다음과 같습니다.

  ```
  CHGJOB OUTQ(*JOB)
  ```

  작업 초기에 디폴트 OUTQ를 변경하면 모든 스풀 파일에 영향을 미칩니다. 작업이 완료되기 바로 전에 OUTQ를 변경할 경우 작업 기록부에만 영향이 미칩니다. 작업 기록부에 영향을 미치기 위해 OVRPRTF(프린터 파일로 대체) 명령을 사용할 수는 없습니다.
- 작업에 대한 출력 대기행렬이 없으면 작업 기록부가 생성되지 않습니다.
- 모든 작업 기록부를 보류하려면 파일 QSYS/QPJOBLOG에 대한 CHGPRTF 명령에 HOLD(*YES)를 지정하십시오. RLSSPLF(스풀 파일 해제) 명령이 수행되면 작업 기록부가 출력기에 대해 해제됩니다. 예를 들면, 다음과 같습니다.

  ```
  CHGPRTF FILE(QSYS/QPJOBLOG)
          HOLD(*YES)
  ```

- 시스템이 비정상 종료되면 시스템 오퍼레이터는 비정상 종료 시 활동 중이던 모든 작업에 대해 작업 기록부를 인쇄할 것인지의 여부를 시작 프롬프트에 지정할 수 있습니다.
- 작업 기록부를 삭제하려면 DLTSPLF(스풀 파일 삭제) 명령 또는 '출력 대기행렬' 화면에서 삭제 옵션을 사용하십시오.
- QSYS/QPJOBLOG 파일에 대한 사용자 자료 값을 변경하기 위해 CHGPRTF(인쇄 파일 변경) 명령에서 USRDTA 매개변수를 사용하면 지정된 값이 '출력 대기행렬에

대한 작업' 화면 또는 '모든 스풀 파일에 대한 작업' 화면에 표시되지 않습니다. 사용자 자료란에 표시된 값은 작업 기록부가 인쇄한 작업의 작업명입니다.

- 프로그래밍 방법으로 작업 기록부를 분석 중인 경우 QMHCTLJL API를 사용하여 작업 기록부를 데이터베이스 파일로 지시하십시오. 인쇄된 형식은 보장되지 않지만 데이터베이스 파일의 레코드 형식은 보장됩니다. 새 필드가 작업 기록부 레코드에 추가되어야 할 경우 새 필드가 레코드 끝에 추가되어 기존 프로그램이 계속 작업합니다. 시스템에 의해 제공된 조회 피처들이 파일상에 직접 사용될 수 있습니다.

대화식 작업 기록부에 대한 고려사항

IBM 제공 작업 설명 QCTL, QINTER 및 QPGMR은 모두 기록부 레벨 LOG(4 0 *NOLIST)를 가지므로 모든 메세지와 메세지의 첫 번째 및 두 번째 레벨 텍스트가 작업 기록부에 기록됩니다. 그러나 SIGNOFF 명령에 *LIST를 지정하지 않는 한, 작업 기록부는 인쇄되지 않습니다. 대화식 작업에 대한 기록 레벨을 변경하려면 CHGJOB 또는 CHGJOBD 명령을 사용하십시오.

표시장치 사용자가 IBM 제공 메뉴나 '명령 입력' 화면을 사용하면 모든 오류 메세지가 표시됩니다. 표시장치 사용자가 사용자 작성 초기 프로그램을 사용할 경우 모니터되지 않은 임의의 메세지로 인해 초기 프로그램이 종료되고 작업 기록부가 생성됩니다. 그러나 초기 프로그램이 메세지를 모니터하면 메세지가 수신될 때 이 초기 프로그램이 제어를 수신합니다. 이 경우 발생한 특정 오류를 찾기 위해 작업 기록부가 생성되는 것을 확인하는 것이 중요합니다. 예를 들면, 디폴트가 *NOLIST인 사인 오프 옵션이 들어 있는 메뉴를 초기 프로그램이 표시한다고 가정합시다. 초기 프로그램은 모든 예외를 모니터하며, 예외가 발생하면 사인 오프 옵션을 *LIST로 변경시키는 CHGVAR(변수 변경) 명령을 가지고 있습니다.

```
   PGM
         DCLF MENU
         DCL &SIGNOFFOPT TYPE(*CHAR) LEN(7) VALUE(*NOLIST)
         .
         .
         .
         MONMSG MSG(CPF0000) EXEC(GOTO ERROR)
PROMPT:  SNDRCVF RCDFMT(PROMPT)
         CHGVAR &IN41 '0'
         .
         .
         .
         IF (&OPTION *EQ '90') SIGNOFF LOG(&SIGNOFFOPT)
         .
         .
         .
         GOTO PROMPT
ERROR:   CHGVAR &SIGNOFFOPT '*LIST'
         CHGVAR &IN41 '1'
         GOTO PROMPT
           ENDPGM
```

위의 예에서 예외가 발생하면 CHGVAR 명령은 SIGNOFF 명령상의 옵션을 *LIST로 변경시키고 인디케이터를 켭니다. 이 인디케이터는 표시장치 사용자에게 예기치 않은 이벤트가 발생했음을 알리고, 수행해야 할 조치 메세지를 나타내는 상수를 조건 지정하는 데 사용됩니다.

대화식 작업이 CL 프로그램이나 프로시듀어를 실행 중이면, 기록부 레벨이 3 또는 4이고 다음 중 하나가 참인 경우에만 CL 명령이 기록됩니다.

- CRTCLPGM(CL 프로그램 작성) 명령, CRTCLMOD(제어 언어(CL) 모듈 작성) 명령 또는 CRTBNDCL(바인드된 CL 프로그램 작성) 명령에 LOG(*YES)를 지정했습니다.
- CRTCLPGM(CL 프로그램 작성) 명령, CRTCLMOD(제어 언어(CL) 모듈 작성) 명령 또는 CRTBNDCL(바인드된 CL 프로그램 작성) 명령에 LOG(*YES)를 지정했으며 (*YES)가 현재 LOGCLPGM 작업 속성입니다.

  SBMJOB, CRTJOBD, CRTJOBD, CHGJOBD 명령에 LOGCLPGM 매개변수를 사용하여 LOGCLPGM 작업 속성을 설정하고 변경할 수 있습니다.

일괄처리 작업 기록부에 대한 고려사항

일괄처리 어플리케이션의 경우 기록되는 정보의 양을 변경하고자 할 수 있습니다. IBM 제공 서브시스템 QBATCH에 대한 작업 설명에 지정된 기록부 레벨(log level)(LOG(4 0 *NOLIST))은 작업의 비정상 종료 시 기록부 전체를 제공합니다. 작업이 정상적으로 완료되면 작업 기록부는 생성되지 않습니다.

모든 경우에 작업 기록부를 인쇄하려면 CHGJOBD(작업 설명 변경) 명령을 사용하여 작업 설명을 변경하거나 BCHJOB 또는 SBMJOB 명령에 다른 LOG 값을 지정해야 합니다. 기록 레벨에 대해 자세히 알려면 319 페이지의 『작업 기록부』를 참조하십시오.

일괄처리 작업이 CL 프로그램이나 프로시듀어를 실행 중인 경우 다음 명령을 사용하여 모듈이나 프로그램을 작성할 때 LOG(*YES)를 지정하면 CL 명령이 항상 기록됩니다.

- 제어 언어 프로그램 작성(CRTCLPGM)
- 제어 언어 모듈 작성(CRTCLMOD)
- 바인드된 CL 프로그램 작성(CRTBNDCL)

또한 CHGJOB 명령과 SBMJOB 명령을 사용할 때 사용자가 LOGCLPGM(*YES)을 지정하면 CL 명령이 기록됩니다.

QHST 이력 기록부

이력 기록부(QHST)는 메세지 대기행렬과 기록부 버전으로 알려진 실제 파일로 구성됩니다. 이력 기록부 메세지 대기행렬로 송신된 메세지는 시스템에 의해 현재 기록부 버전의 실제 파일에 기록됩니다.

- 이력 기록부(QHST). 시스템, 서브시스템 및 작업 정보와 같이 시스템 활동에 관한 고급 레벨의 추적, 장치 상태, 그리고 시스템 오퍼레이터 메세지가 들어 있습니다. 이 메세지 대기행렬은 QHST입니다.

기록부 버전이 가득 차면 새로운 버전의 기록부가 자동으로 작성됩니다. 각 버전은 다음과 같은 방식으로 명명되는 실제 파일입니다.

```
Qxxxyydddn
```

여기에서

xxx 기록부 유형(HST)의 설명(3자)

yyddd 기록부 버전에서 첫 번째 메세지의 줄리안 날짜입니다.

n 율리우스력 날짜 내의 순번(A-Z 및 0-9)입니다.

주: 이력 기록부의 각 버전에 있는 레코드 수가 시스템 값 QHSTLOGSIZ에 지정됩니다.

기록부 버전 파일의 테스트는 기록부 버전의 처음과 마지막 메세지의 날짜와 시간을 포함합니다. 첫 번째 메세지의 날짜와 시간은 텍스트의 1-13 자리에 있으며, 마지막 메세지의 날짜와 시간은 14-26 자리에 있습니다. 날짜와 시간은 cyymmddhhmmss 형식으로 되어 있습니다. 여기에서

c 세기 자릿수

yymmdd
메세지가 송신된 날짜

hhmmss
메세지가 송신된 시간

같은 줄리안 날짜를 가지고 최대 36개의 기록부 버전을 생성할 수 있습니다. 36개보다 많은 기록부 버전들이 같은 날짜에 생성되면 그 다음으로 가능한 줄리안 날짜가 후속 기록부 버전에 사용됩니다. 몇 개의 이전 기록부 버전들이 삭제되면 그 이름을 다시 사용할 수 있습니다. 기록부 버전들은 이름들을 삭제한 후 다시 그 이름을 사용한 경우 이름순으로 되었을 때 순서에 맞지 않게 됩니다.

사용자는 이력 기록부의 레코드를 처리하는 프로그램을 작성할 수 있습니다. 각 기록부마다 여러 버전을 사용할 수 있으므로 처리될 기록부 버전을 선택해야 합니다. 사용 가능한 기록부 버전을 판별하려면 DSPOBJD(오브젝트 설명 표시) 명령을 사용해야 합니다. 예를 들면, 다음의 DSPOBJD 명령은 사용 가능한 이력 기록부의 버전을 표시합니다.

```
DSPOBJD   OBJ(QSYS/QHST*)   OBJTYPE(*FILE)
```

WRKOBJ(오브젝트에 대한 작업) 명령으로 표시된 화면에서 삭제 옵션을 사용하여 사용자 시스템에 대한 기록부를 삭제할 수 있습니다.

DSPLOG(기록부 표시) 명령을 사용하여 기록부 안의 정보를 표시하거나 인쇄할 수 있습니다. 또한 다음 여러 사항을 지정하여 표시하거나 인쇄하려는 정보를 선택할 수 있습니다.

- 기간
- 기록부 항목을 송신한 작업명
- 항목의 메세지 ID

다음 DSPLOG 명령은 현재 날짜의 작업 OEDAILY에 대한 모든 항목을 표시합니다.

```
DSPLOG JOB(OEDAILY)
```

다음은 명령 사용 후 표시되는 화면입니다.

```
                        이력 기록부 내용 표시

Job OEDAILY started
Database file OEMSTR in library OELIB expired
Job OEDAILY abnormally ended
Job OEDAILY started
Job OEDAILY ended

                                                                  맨 아래
계속하려면 Enter 키를 누르십시오.

 F3=나감      F10=모두 표시      F12=취소
```

시스템 날짜 또는 시간을 이전의 설정값으로 재설정하거나 시스템 날짜와 시간을 48시간보다 많은 시간으로 앞당기면, 새로운 기록부 버전이 시작됩니다. 이것은 단일 기록부 버전 안의 모든 메세지들이 연대순으로 됨을 나타냅니다.

V3R6M0보다 이전의 릴리스로 생성된 기록부 버전들은 시스템 날짜와 시간이 이전의 설정값으로 재설정되면 연대순으로 되지 않은 항목들을 포함합니다. 그러므로 사용자가 기록부 버전을 표시하려고 할 때 몇 개의 항목들이 누락될 수 있습니다. 예를 들면, 기록부 버전에서 날짜가 1988년인 항목 뒤에 1987년 항목이 있고 사용자가 1987년 항목이 표시되길 원할 경우 DSPLOG 명령의 PERIOD 매개변수에 1987년도를 지정하더라도, 원하는 항목은 표시되지 않습니다. 사용자는 시스템 날짜(QDATE) 및 시스템 시간(QTIME)을 사용해야 하며 다음과 같이 PERIOD 매개변수를 지정해야 합니다.

```
PERIOD((start-time start-date) (*AVAIL *END))
```

메세지 대기행렬이 가득 찼거나 DSPLOG 명령이 사용되었을 때 시스템은 기록부 메세지 대기행렬에 송신된 메세지를 현재 버전의 실제 파일에 기록합니다. 현재 버전을 최신 버전으로 하려면 DSPLOG 명령에 ###0000 같은 가상의 메세지 ID를 지정합니다. 그러면 메세지가 표시되지는 않으나 기록부 버전 실제 파일이 현재 버전으로 됩니다.

기록부 표시 명령과 출력 매개변수 *PRINT, DSPLOG OUTPUT(*PRINT)을 사용하여 기록부에 정보를 인쇄하면 각 메세지에서 하나의 행(각 메세지의 처음 105자)만 인쇄됩니다.

기록부 표시 명령과 출력 매개변수 *PRTWRAP, DSPLOG OUTPUT(*PRTWRAP)을 사용하여 기록부에 정보를 인쇄하면 106자 이상의 메세지가 랩되어 최대 2000자까지 허용하도록 추가 행이 포함됩니다.

기록부 표시 명령과 출력 매개변수 *PRTSECLVL을 사용하여 기록부에 정보를 인쇄하면 106자 이상의 메세지가 랩(wrap)되어 최대 2000자까지 허용하는 추가 행이 포함됩니다. 가능한 경우 최대 6000자까지 두 번째 레벨의 메세지 텍스트도 인쇄됩니다.

DSPLOG(기록부 표시) 명령을 사용하여 기록부에 정보를 표시하면 메세지 텍스트 중 105자만 표시됩니다. 105자 다음의 문자는 오른쪽에서 절단됩니다.

이력 기록부 형식

데이터베이스 파일은 시스템에서 기록부로 송신한 메세지를 저장하는 데 사용됩니다. 실제 파일 안의 모든 레코드는 길이가 동일하고 기록부에 송신된 메세지는 길이가 서로 다르므로 해당 메세지가 둘 이상의 레코드에 걸쳐 생성될 수 있습니다. 메세지에 대한 각 레코드에는 다음과 같은 세 개의 필드가 있습니다.

- 시스템 날짜 및 시간(길이가 8자인 문자 필드). 이 필드는 내부 필드입니다. 변환된 날짜와 시간도 메세지에 들어 있습니다.
- 레코드 번호(2바이트 필드). 예를 들면, 필드에는 첫 번째 레코드에 16진 0001, 두 번째 레코드에 16진 0002 등이 들어 있습니다.
- 자료(길이가 132자인 문자 필드).

다음은 첫 번째 레코드의 세 번째 필드(자료) 형식입니다.

내용	유형	길이	레코드에서의 위치
작업명	문자	26	11-36
변환된 날짜와 시간[1]	문자	13	37-49
메세지 ID	문자	7	50-56
메세지 파일명	문자	10	57-66
라이브러리명	문자	10	67-76
메세지 유형[2]	문자	2	77-78
심각도 코드(severity code)	문자	2	79-80

내용	유형	길이	레코드에서의 위치
송신 프로그램명[3]	문자	12	81-92
송신 프로그램 명령어 번호[4]	문자	4	93-96
수신 프로그램명[3]	문자	10	97-106
수신 프로그램 명령어 번호[4]	문자	4	107-110
메세지 텍스트 길이	2진	2	111-112
메세지 자료 길이	2진	2	113-114
텍스트 또는 자료의 코드화 문자 세트 ID(CCSID)[5]	2진	4	115-118
사용자 프로파일 송신	문자	10	119-128
예약	문자	14	129-142

:

1 형식은 cyymmddhhmmss입니다.

여기에서

c 세기 자릿수(century digit)(yy≥40이면 c=0,yy<40이면 c=1)

yymmdd 메세지가 송신된 년월일

hhmmss 메세지가 송신된 시분초

2 이는 RCVMSG(메세지 수신) 명령의 RTNTYPE 매개변수와 값이 동일합니다.

3 송신자 또는 리시버가 ILE 프로시듀어이면, 이력 기록부 내의 항목에 ILE 프로그램명만 들어 있습니다. 모듈명과 프로시듀어명은 이력 기록부 안에 포함되지 않습니다.

4 송신자 또는 리시버가 ILE 프로시듀어이면, 송신 또는 수신 명령어 번호는 0입니다.

5 메세지가 저장된 메세지인 경우 이 CCSID는 *CCHAR 자료로 정의된 메세지 자료에만 적용됩니다. 나머지 메세지 자료는 65 535로 간주될 수 있습니다. 그 이외의 경우 이것은 즉시 메세지의 CCSID입니다.

다음은 나머지 레코드의 세 번째 필드(자료) 형식입니다.

내용	유형	길이
메세지	문자	변수[1]
메세지 자료	문자	변수[2]

:

1 이 길이는 첫 번째 레코드에 지정되며(111 및 112열) 132를 초과할 수 없습니다.

2 이 길이는 첫 번째 레코드에 지정됩니다(113 및 114열).

기록부의 새로운 버전이 시작될 때는 메세지가 분할되지 않습니다. 메세지의 첫 번째 레코드와 최종 레코드는 항상 동일한 QHST 버전에 들어 있습니다.

메세지의 메세지 자료에 대한 설명은 223 페이지의 『대체 변수 정의』 부분을 참조하십시오.

QHST 파일의 처리

HLL 프로그램을 사용하여 QHST 파일을 처리하는 경우 메세지마다 길이가 다를 수 있으므로 유의하십시오. 메세지에는 대체 가능한 변수가 포함되므로 메세지의 실제 길이는 다양합니다. 따라서 동일한 메세지를 각각 사용하도록 메세지 자료가 변수 위치에서 시작됩니다.

QHST 작업 시작 및 완료 메세지

시스템은 작업 시작과 작업 완료 메세지에 대해 특별한 형식화를 수행합니다. 메세지 CPF1124(작업 시작)와 메세지 CPF1164(작업 완료)의 경우 메세지 자료는 항상 세 번째 레코드의 11열에서 시작됩니다.

작업 사용 통계에는 CPF1124 및 CPF1164보다 많은 정보가 들어 있습니다. 간단한 작업 사용 통계 기능의 경우에는 CPF1164 메세지를 사용하십시오.

성능 정보는 메세지 CPF1164의 텍스트로 표시되지 않습니다. 메세지가 QHST 기록부(log)에 들어 있으므로 사용자는 이 자료를 검색하기 위한 어플리케이션 프로그램을 작성할 수 있습니다. 성능 정보의 형식은 다음과 같습니다.

성능 정보는 변수 길이 대체 텍스트 값으로서 전달됩니다. 이는 자료 구조의 첫 번째 항목이 자료의 길이임을 의미합니다. 길이 필드의 크기는 길이에 포함되지 않습니다. 구조 안의 첫 번째 자료 필드는 작업이 시스템에 입력된 시간과 날짜, 그리고 작업에 대한 첫 번째 라우팅 단계가 시작된 시간과 날짜입니다. 시간 형식은 'hh:mm:ss'입니다. 분리자는 언제나 콜론(:)입니다. 날짜는 시스템 값 QDATFMT에 정의된 형식으로 되어 있고 분리자는 시스템 값 QDATSEP에 정의된 형식으로 되어 있습니다. 작업이 시스템에 입력된 시간과 날짜는 구조 안의 작업 시작 시간 및 날짜보다 앞선 것입니다.

시스템에 작업이 입력된 시간과 날짜는 시스템이 작업의 시작을 알게 되는 시점입니다. (작업 구조는 작업과는 별도로 설정됩니다.) 대화식 작업인 경우에 작업 입력 시간은 시스템이 암호를 인식하는 시점입니다. 일괄처리 작업인 경우 이 시간은 BCHJOB 또는 SBMJOB 명령이 처리되는 시점입니다. 모니터 작업이나 판독기 또는 출력기에 있어서 이 시간은 해당 시작 명령이 처리되는 시점이며, 자동 시작 작업인 경우 이 시간은 서브시스템의 시작 시간입니다.

시간과 날짜 뒤에 있는 항목이 총 응답 시간과 트랜잭션 횟수입니다. 총 응답 시간은 초 단위로 지정되며 워크스테이션에서 Enter 키를 누를 때부터 다음 화면이 나타날 때까지 작업이 처리된 모든 시간 간격의 누적치가 들어있습니다. 이 정보는 WRKACTJOB 화면에 나타난 정보와 유사합니다. 이 필드는 대화식 작업에서만 의미가 있습니다.

시스템 장애 또는 작업의 비정상 종료시에는 최종 트랜잭션이 총계에 포함되지 않습니다. 이 경우 작업 종료 코드는 41 이상이 됩니다. 트랜잭션 횟수는 콘솔 작업을 제외한 대화식 작업에서만 의미가 있고, 이 수는 작업하는 동안 시스템에 의해 계산된 응답 시간 간격입니다.

동기 보조 I/O 조작의 수는 트랜잭션 수에 따라 다릅니다. 이는 이 값이 작업에 대한 총계라는 점을 제외하면 WRKACTJOB 화면에 나타나는 AUXIO 필드와 그 값이 동일합니다. 작업이 종료 코드 70으로 종료되면 이 값에는 최종 라우팅 단계에 대한 계산이 포함되지 않을 수도 있습니다. 그 이외에 작업이 IPL 동안 존재하고(TFRBCHJOB 명령을 사용하여) 작업이 IPL 다음에 활성화되기 전에 종료된 경우 값은 0입니다.

성능 통계에서의 최종 필드는 작업 유형입니다. 다음은 이 필드의 값입니다.

A 자동 시작 작업

B 일괄처리 작업

I 대화식 작업

M 서브시스템 모니터

R 스풀링 판독기

S 시스템 작업

W 스풀링 출력기

X 시작 작업

메세지 자료가 변수 위치에서 시작하는 메세지인 경우 다음과 같이 하여 메세지 자료에 액세스할 수 있습니다.

- 메세지 안의 변수 길이를 결정하십시오. 예를 들어, 메세지가 다음 5개의 변수를 사용하는 것으로 가정하십시오.

  ```
  작업명       *CHAR 10
  사용자명     *CHAR 10
  작업 번호    *CHAR 6
  시간         *CHAR 8
  날짜         *CHAR 8
  ```

  이 변수들은 메세지 자료의 처음 42열 안에 고정됩니다.
- 메세지 자료의 위치를 찾으려면 다음 사항을 고려하십시오.
  - 메세지는 항상 두 번째 레코드의 11열에서 시작합니다.
  - 메세지 길이는 첫 번째 레코드의 111열에서 시작하는 2자리 필드에 저장됩니다. 이 길이는 2진값(binary value)으로 저장되어 메세지 길이가 60이면 필드에는 16진 003C가 들어갑니다.

  이때 메세지 길이와 메세지의 시작 위치를 사용하면 메세지 자료의 위치를 알 수 있습니다.

QHST 파일의 삭제

기록부 버전 실제 파일은 시스템상에 누적되므로 필요없는 이전 기록부는 정기적으로 삭제해야 합니다. 기록부 버전은 보안 담당자만이 삭제 권한을 갖도록 작성되어 있습니다.

조작 지원에서는 기존 QHST 파일의 삭제를 포함하여 클린업 기능을 제공합니다. 다른 기능으로는 다음과 같은 것이 있습니다.

- 보안 담당자로서 다음을 지정합니다.

  ```
  WRKOBJ  OBJ(QSYS/QHST*)  OBJTYPE(*FILE)
  ```

  기존의 불필요한 파일을 삭제하려면 옵션 4를 사용하십시오.

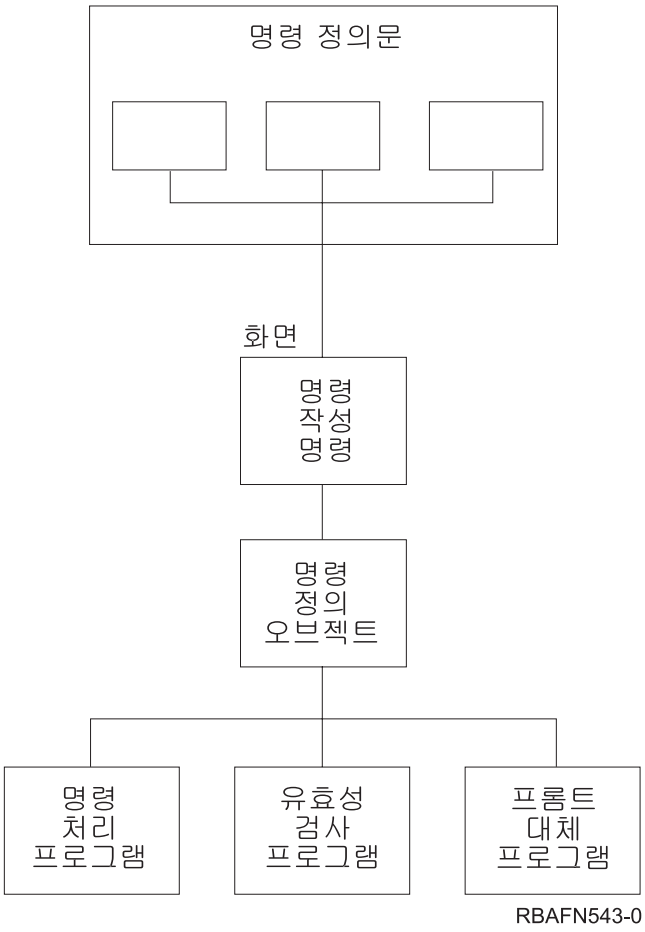
제 9 장 명령 정의

CL 명령은 시스템이 기능을 수행하도록 요구하는 명령문입니다. 명령을 입력하면 기능을 수행하는 프로그램이 시작됩니다. CL 명령을 사용하여 광범위한 기능을 요구할 수 있습니다. IBM이 제공하는 이러한 명령을 사용하거나 IBM에서 제공하는 디폴트 값을 변경하고 사용자 고유의 명령을 정의할 수도 있습니다. 이 장에서는 사용자가 명령을 정의하고 작성하는 방법을 설명합니다. CL 명령 및 키워드 약어에 대한 도움말이 필요하면 489 페이지의 부록 D『CL 명령어 및 키워드의 약어』 부분을 참조하십시오.

사용자 고유의 CL 명령을 정의하고 작성할 때 관련 문서를 제공할 수도 있습니다. 사용자 고유의 명령에 대한 온라인 도움말 작성 방법 및 도움말을 사용한 HTML 페이지 작성 방법에 대한 정보는 413 페이지의 제 10 장『명령 문서화』 부분을 참조하십시오.

명령 정의 방법의 개요

다음 그림은 명령을 작성하는 단계에 대해 설명합니다. 그림 다음에 나오는 텍스트는 각 단계에 대한 설명입니다.



사용자 자신의 유효성 검사 및 프롬트 대체 프로그램을 작성하는 것은 선택적 단계입니다.

단계 설명

명령 정의문

명령 정의문에는 워크스테이션 사용자가 입력하도록 프롬트하고, 그 입력이 유효한지 검사하고, 명령이 실행될 때 호출되는 값(프로그램으로 전달됨)을 정의하는 데 필요한 정보가 들어 있습니다.

명령 정의문은 CRTCMD 명령에 대한 입력으로서 지원되는 모든 파일 안에 존재할 수 있습니다. 341 페이지의 표 9에는 명령 정의에 사용되는 명령문이 들어 있습니다.

표 9. CL 명령을 정의하기 위한 명령문

명령문 유형	명령문 이름	설명
명령	CMD	명령 이름에 대한 프롬트 텍스트(있는 경우) 지정
매개변수	PARM	명령에 대한 매개변수 정의
요소	ELEM	매개변수 값으로 사용되는 리스트 안의 요소 정의
규정자	QUAL	매개변수로 사용되는 이름의 규정자 정의
종속성	DEP	하나의 매개변수에 대한 종속성 또는 여러 매개변수들 간의 종속성 정의
프롬트 제어	PMTCTL	특정 매개변수가 프롬트되는 조건 정의

CRTCMD(명령 작성) 명령

CRTCMD 명령은 명령 정의문을 처리하여 명령 정의 오브젝트를 작성합니다. CRTCMD 명령은 대화식이나 일괄처리 작업으로 수행될 수 있습니다.

명령 정의 오브젝트

명령 정의 오브젝트는 명령이 유효하고 올바른 매개변수가 입력되었는지 확인하기 위해 시스템 프로그램에서 검사하는 오브젝트입니다.

유효성 검사

시스템은 명령의 유효성 검사를 수행합니다. 필요하지 않더라도 사용자 자신의 유효성 검사 프로그램을 작성할 수도 있습니다.

시스템에서 수행하는 유효성 검사는 다음 사항을 확인합니다.

- 필수 매개변수 값의 입력 여부
- 각 매개변수 값의 자료 유형 및 길이 요구사항에 적합한지의 여부
- 각 매개변수 값이 다음과 같은 명령 정의에서 지정된 선택적인 요구와 일치하는지의 여부
  - 유효한 값의 리스트
  - 값의 범위
  - 값의 관계 비교
- 상호 상충되는 매개변수가 입력되었는지의 여부

시스템은 다음의 경우에 유효성 검사를 수행합니다.

- 명령이 표시장치에서 대화식으로 입력되는 경우
- 명령이 스풀링을 사용하여 일괄처리 입력 스트림으로부터 입력되는 경우
- 명령이 소스 입력 유틸리티(SEU)를 통해 데이터베이스 파일에 입력되는 경우

- 명령이 HLL로부터의 호출에 의해 QCMDEXC, QCMDCHK 또는 QCAPCMD 프로그램으로 전달되는 경우 QCMDEXC 프로그램에 대해 자세히 알려면 제 6 장을 참조하십시오.
- CL 모듈이나 OPM 프로그램이 작성되는 경우
- CL 프로시듀어나 프로그램 또는 REXX 프로시듀어에 의해 명령이 수행되는 경우
- C 언어 시스템 기능을 사용하여 명령이 수행되는 경우

시스템이 수행하는 것보다 더 많은 유효성 검사를 필요로 하는 경우 *유효성 검사 프로그램*이라 부르는 프로그램을 작성하거나(404 페이지의 『유효성 검사 프로그램 작성』 참조) 명령 처리 프로그램에 유효성 검사를 포함시킬 수 있습니다. CRTCMD 명령에서 명령 처리와 유효성 검사 프로그램에 모두 대한 이름을 지정합니다.

명령에 유효성 검사 프로그램이 있으면, 시스템이 명령 매개변수 값을 유효성 검사 프로그램으로 전달합니다. 이것은 시스템이 명령 처리 프로그램을 호출하기 전에 발생합니다. 유효성 검사 프로그램은 다음과 같은 조건에서 구문 검사 시에 실행됩니다.

- 명령을 실행할 때
- CL 소스 멤버에 명령을 입력하기 위해 소스 입력 유틸리티(SEU)를 사용할 때와 프로그래머가 명령에 지정된 매개변수의 변수 대신 상수를 사용할 때
- 명령에 지정된 모든 매개변수의 변수 대신 상수를 사용하는 CL 프로그램이나 프로시듀어를 컴파일할 때

프로그램이 오류를 발견하면 사용자에게 메세지를 수신하므로 사용자는 오류를 즉시 정정할 수 있습니다. 명령 처리 프로그램은 전달된 자료를 맞는 것으로 가정합니다. 유효성 검사 프로그램을 작성하는 데 관한 추가 정보는 404 페이지의 『유효성 검사 프로그램 작성』 부분을 참조하십시오.

프롬트 대체 프로그램

명령이 프롬트될 때 매개변수 디폴트의 현재 값을 제공하려는 경우 프롬트 대체 프로그램을 작성할 수 있습니다. 예를 들어, 매개변수 값에 디폴트 값인 *SAME을 제공할 경우 변경 명령에 프롬트 대체 프로그램이 자주 사용됩니다. 상세한 내용은 381 페이지의 『키 매개변수 및 프롬트 대체 프로그램 사용』 부분을 참조하십시오. 프롬트 대체 프로그램은 선택적 프로그램입니다.

명령 처리 프로그램

명령 처리 프로그램(CPP)은 명령 분석 프로그램이 요구받은 기능을 수행하기 위해 호출하는 프로그램입니다. CPP는 CL 프로그램, 다른 HLL 프로그램 또는 REXX 프로시듀어가 될 수 있습니다. 예를 들어, 사용자의 명령이 호출하는 어플리케이션 프로그램이나, 시스템 명령 또는 일련의 명령을 포함하는 CL 프로그램 또는 REXX 프로시듀어가 될 수 있습니다.

CPP는 명령 정의문에 의해 정의되는 매개변수를 반드시 허용해야 합니다.

명령 나감 프로그램 및 독립 ASP

특정 명령이 필요로 하는 명령 처리 프로그램, 유효성 검사 프로그램, 프롬트 대체 프로그램, 선택사항 프로그램 또는 프롬트 제어 프로그램을 포함하여 어떤 나감 프로그램이든지 명령과 같은 독립 보조 기억장치 풀(ASP)이나 시스템 ASP(ASP 1)나 기본 ASP(ASP 2-32)에 있어야 합니다. 명령이 하나의 독립 ASP에 있고 나감 프로그램이 다른 독립 ASP에 있어서는 안됩니다. 나감 프로그램이 상주하는 독립 ASP를 사용할 수 없을 때 명령을 실행하면 문제가 발생합니다(예를 들어, 독립 ASP가 단절변환 상태일 때).

사용자 정의 명령에 필요한 권한

사용자들이 프로그래머가 작성한 명령을 사용하기 위해서는 명령에 대한 조작 권한과 명령 처리 프로그램 및 선택적 유효성 검사 프로그램에 대한 자료 권한을 가지고 있어야 합니다. 사용자들은 명령을 포함한 라이브러리, 명령 처리 프로그램 및 유효성 검사 프로그램에 대한 읽기 권한도 가지고 있어야 합니다. 명령 처리 프로그램이나 유효성 검사 프로그램이 임의의 서비스 프로그램을 참조하는 경우 사용자에게는 그 서비스 프로그램과 서비스 프로그램 라이브러리에 대한 실행 권한이 있어야 합니다. 사용자에게는 아래에 나열된 프로그램에 대한 실행 권한이 있어야 합니다.

- 명령 처리 프로그램(CPP)
- 유효성 검사 프로그램(VCP)
- CPP 또는 VCP에 의해 사용되는 임의의 서비스 프로그램
- CPP, VCP 또는 서비스 프로그램을 포함한 라이브러리

사용자에게는 명령 처리 프로그램에서 실행하는 다른 명령에 대해서도 올바른 권한이 있어야 합니다. 또한 파일을 열기 위해서는 파일에 대한 권한도 반드시 있어야 합니다.

명령 작성의 예

시스템 오퍼레이터가 프로그램을 호출하여 시스템을 시작할 수 있도록 하는 명령을 작성하기 위해서는 다음과 같이 해야 합니다. (이 예는 IBM 제공 소스 파일을 사용한다고 가정한 것입니다.)

1. STARTUP 멤버명을 사용하여 QCMDSRC 소스 파일에 명령 정의 소스문을 입력하십시오.

   ```
   CMD PROMPT('S Command for STARTUP')
   ```

2. 다음 명령을 입력하여 명령을 작성하십시오.

   ```
   CRTCMD CMD(S) PGM(STARTUP) SRCMBR(STARTUP)
   ```

3. STARTUP 프로그램(명령 처리 프로그램)에 대한 소스문을 입력하십시오.

   ```
            PGM
   STRSBS QINTER
   STRSBS QBATCH
   STRSBS QSPL
   ```

```
STRPRTWTR DEV(QSYSPRT) OUTQ(QPRINT) WTR(WTR)
STRPRTWTR DEV(WSPR2) OUTQ(WSPRINT) WTR(WTR2)
SNDPGMMSG MSG('STARTUP procedure completed') MSGTYPE(*COMP)
          ENDPGM
```

4. CRTBNDCL(바인드된 CL 프로그램 작성) 명령을 사용하여 프로그램을 작성하십시오.

```
CRTBNDCL STARTUP
```

이전 예에서 S는 새로운 명령(CMD 매개변수에 의해 지정됨)의 이름입니다. STARTUP은 명령 처리 프로그램(PGM 매개변수에 의해 지정됨)의 이름이면서 또한 명령 정의문이 들어 있는 소스 멤버(SRCMBR 매개변수에 의해 지정됨)의 이름입니다. 이제 시스템 오퍼레이터가 명령을 호출하기 위해 S를 입력하거나 명령 처리 프로그램을 호출하기 위해 CALL STARTUP을 입력할 수 있습니다.

명령 정의 방법

명령을 작성하기 위해서는 먼저 명령 정의문을 사용하여 명령을 정의해야 합니다. 자세한 정보는 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오. 다음은 명령 정의문의 일반적인 형식 및 코딩 규칙의 요약입니다.

명령문 코딩 규칙

CMD 하나의 CMD문만 사용되어야 합니다. CMD문은 소스 파일 안의 어느 곳에든지 위치할 수 있습니다.

PARM

최대 99개의 PARM문을 사용할 수 있습니다. PARM문을 소스 파일에 입력하는 순서가 명령 처리 프로그램(CPP)과 유효성 검사 프로그램(VCP)으로 매개변수가 전달되는 순서를 결정합니다. 명령 처리 프로그램에 전달될 각 매개변수마다 하나의 PARM문이 필요합니다. 매개변수를 키 매개변수로 지정하려면 PARM문에 대해 KEYPARM(*YES)을 지정해야 합니다. KEYPARM(*YES)으로 코드화되는 매개변수의 수는 변경될 오브젝트를 고유하게 정의하는 데 필요한 매개변수 숫자로 제한됩니다. 키 매개변수를 사용하려면 명령을 작성할 때 프롬트 대체 프로그램이 지정되어야 합니다. 키 매개변수는 PMTCTL(*PMTRQS)이나 PMTCTL(레이블)과 함께 정의할 수 없습니다.

ELEM

하나의 리스트에 최대 300개까지의 ELEM문이 허용됩니다. ELEM문을 소스 파일에 입력하는 순서가 리스트 안의 각 요소의 순서를 결정합니다. 첫 번째 ELEM문은 리스트에 대한 PARM이나 ELEM문의 TYPE 매개변수에 있는 명령문 레이블과 일치되는 명령문 레이블을 가지고 있어야 합니다.

QUAL

하나의 규정된 이름에는 최대 300개의 규정자가 허용됩니다. QUAL문을 소스

파일에 입력하는 순서가, 규정자가 지정되어야 하는 순서 및 규정자가 유효성 검사 프로그램과 명령 처리 프로그램으로 전달되는 순서를 결정합니다.

DEP DEP문은 DEP문이 참조되는 모든 PARM문 다음에 위치해야 합니다. 따라서 DEP문은 일반적으로 소스 파일의 거의 끝 부분에 위치합니다.

PMTCTL

PMTCTL문은 PMTCTL문이 참조되는 모든 PARM문 다음에 위치해야 합니다. 따라서 PMTCTL문은 일반적으로 소스 파일의 끝에 위치합니다.

소스 파일에서는 적어도 하나 이상의 PARM문이 ELEM문이나 QUAL문 앞에 있어야 합니다. 명령 정의문을 입력하는 소스 파일은 명령을 작성할 때 CRTCMD 명령에 의해 사용됩니다. 소스 파일에 명령문을 입력하는 것에 관한 자세한 정보는 AS/400® 용 ADTS: 소스 입력 유틸리티(SEU)  책을 참조하십시오.

CMD문 사용

명령을 정의할 때 명령 정의문과 함께 단 하나의 CMD문만 포함시켜야 합니다.

명령을 정의할 때 사용자에게 명령 프롬트 텍스트를 제공할 수 있습니다. 사용자가 명령 전체를 입력하는 대신 명령이 프롬트되도록 선택했으면, 사용자는 명령어를 입력하고 F4(프롬트) 키를 누르면 됩니다. 이때 명령어와 머리말 프롬트 텍스트가 표시된 명령 프롬트가 화면의 1행에 나타납니다.

명령에 대한 프롬트 텍스트를 지정하려면 CMD문의 PROMPT 매개변수를 사용하여 프롬트에 대한 머리말을 지정하십시오. 그런 다음, PARM, ELEM, QUAL문의 PROMPT 매개변수에 매개변수, 리스트의 요소, 규정자에 대한 프롬트를 지정하십시오.

CMD문의 PROMPT 매개변수에 실제 프롬트 머리말 텍스트를 최대 30자의 문자 스트링으로 지정하거나 메세지 설명의 메세지 ID를 지정할 수 있습니다. 다음은 명령 ORDENTRY에 문자 스트링이 지정된 예입니다.

```
CMD PROMPT('Order Entry')
```

다음은 사용자가 명령어를 입력하고 F4 키를 누른 후에 나타나는 프롬트의 제 1행입니다.

```
Order Entry (ORDENTRY)
```

정의할 명령에 대한 프롬트 텍스트를 제공하지 않으려면 CMD문에 단어 CMD만 사용하면 됩니다. 그러나 문서화 목적으로 PROMPT 키워드를 사용할 수도 있습니다.

매개변수 정의

각 명령에 대해 최대 99개의 매개변수를 정의할 수 있습니다. 매개변수를 정의하려면 PARM문을 사용하십시오.

PARM문에는 다음 사항을 지정합니다.

- 매개변수에 대한 키워드명
- 매개변수가 키 매개변수인지의 여부
- 전달되는 매개변수 값의 유형
- 매개변수 값의 길이
- 필요한 경우 매개변수의 디폴트 값

또한 매개변수를 정의할 때에는 다음 사항을 고려해야 합니다(괄호 안은 관련된 PARM 문 매개변수).

- 명령 처리 프로그램이 값을 리턴시키는지 여부(RTNVAL). RTNVAL(*YES)이 지정되었을 때 사용자가 값을 리턴시키기 위해서는 명령이 호출될 때 리턴 변수가 명령 내에 코드화되어 있어야 합니다. RTNVAL(*YES) 매개변수에 변수가 지정되지 않으면 널 포인터가 명령 처리 프로그램으로 전달됩니다.
- 매개변수가 사용자에게 프롬트로 표시되지 않지만, 상수로서 명령 처리 프로그램에 전달되는지의 여부(CONSTANT)
- 매개변수가 VALUES, SPCVAL 또는 SNGVAL 매개변수상에 지정된 특정한 값으로 제한되는지의 여부(RSTD) 또는 매개변수의 유형, 길이, 값의 범위, 지정된 관계 등에 일치하는 값이 매개변수에 포함되는지의 여부
- 특정한 유효 매개변수 값의 종류(VALUES, SPCVAL 및 SNGVAL)
- 매개변수 값의 유효성을 결정하기 위해 매개변수 값에 대해 수행해야 할 테스트의 종류(REL 및 RANGE)
- 매개변수가 선택적인지 필수인지의 여부(MIN)
- 간단한 리스트를 요구하는 매개변수에 지정될 수 있는 값의 한계(MIN 및 MAX)
- 인쇄 불능 자료(16진수 00에서 3F 또는 FF 사이의 문자)가 매개변수 값으로 입력될 수 있는지의 여부(ALWUNPRT)
- 매개변수 값에 변수명이 입력될 수 있는지의 여부(ALWVAR)
- 값이 프로그램명인지의 여부(PGM)
- 값이 자료 영역명인지의 여부(DTAARA)
- 값이 파일명인지의 여부(FILE)
- 값이 지정된 길이와 정확하게 일치하는지의 여부(FULL)
- 값과 함께 값의 길이가 제공되어야 하는지 여부(VARY)
- 매개변수 값에 표현식(expression)을 지정할 수 있는지의 여부(EXPR)
- 매개변수로 전달되는 값에 관하여 속성 정보가 제공되어야 하는지의 여부(PASSATR)
- 정의하려는 매개변수가 지정되지 않은 경우 명령 처리 프로그램이나 유효성 검사 프로그램에 값(PASSVAL)이 전달되어야 하는지의 여부

- 대소문자 값이 그대로 유지되는지 아니면 대소문자 값이 대문자로 변환되는지의 여부(CASE)
- 리스트 변위(LISTDSPL) 안의 리스트가 2바이트 2진값 또는 4바이트 2진값인지의 여부
- 메세지 ID 또는 매개변수에 대한 프롬트 텍스트의 종류(PROMPT)
- 프롬트 화면상의 가능한 선택 필드에 표시되는 유효 값(CHOICE)
- 프로그램이 선택값을 제공하는지의 여부(CHOICEPGM)
- 매개변수에 대한 프롬팅이 다른 매개변수에 의해 제어되는지의 여부(PMTCTL)
- 프로그램이 PMTCTL문에 대해 값을 제공하는지의 여부(CTL 키워드에서 참조되는 매개변수에 대해)(PMTCTLPGM)
- 작업 기록부 안에서 또는 명령이 프롬트될 때 값을 은닉(hidden)시킬지의 여부(DSPINPUT)

매개변수에 대한 키워드 명명

매개변수로 선택하는 키워드명은 매개변수 값으로 요구받은 정보에 대해 설명하는 것이어야 합니다. 예를 들면, 사용자명은 USER, 비교값은 CMPVAL, 그리고 주문 입력 유형은 OETYPE 등입니다. 키워드는 최대 10자까지 가능한 영숫자이며 첫자는 영문자여야 합니다.

매개변수 유형

다음은 기본 매개변수의 유형입니다(괄호 안은 TYPE 매개변수 값임).

- 10진(*DEC). 매개변수 값이 10진수입니다. 이 유형의 매개변수는 LEN 매개변수상에 지정된 길이의 팩 10진값으로 명령 처리 프로그램(CPP: Command Processing Program)에 전달됩니다. 매개변수에 정의된 값 이상의 소수 자릿수는 절단됩니다.
- 논리(*LGL). 매개변수 값이 논리값 '1' 또는 '0'입니다. 이 유형의 매개변수는 길이 1인 문자 스트링으로 CPP에 전달됩니다(`F1` 또는 `F0`).
- 문자(*CHAR). 매개변수 값은 문자 스트링입니다. 이는 어포스트로피로 묶을 수 있으며 LEN 매개변수에 지정된 길이의 문자 스트링으로 CPP에 전달됩니다. 값은 어포스트로피가 제거되고 좌측 정렬되며 여분은 공백으로 채워져서 전달됩니다.
- 이름(*NAME). 매개변수 값은 기본명을 나타내는 문자 스트링입니다. 이름의 최대 길이는 256자입니다. 첫 번째 문자는 영문(A-Z), $, # 또는 @입니다. 나머지 문자로는 첫 번째 문자에 사용되는 문자들, 숫자 0-9, 밑줄(_) 및 마침표(.)가 사용될 수 있습니다. 이름은 또한 큰따옴표(″)로 시작하고 종료되는 문자 스트링도 될 수 있습니다. 시스템이 값을 LEN 매개변수에 지정된 길이의 문자 스트링으로 CPP에 전달합니다. 값은 좌측 정렬되며 공백으로 채워집니다. 보통, 오브젝트명에는 *NAME 유형을 사용합니다. *LIBL 또는 *NONE과 같은 특수 값을 이름 매개변수 값으로 입

력할 때에는 SPCVAL 매개변수에서 이 특수 값을 설명해야 합니다. 표시장치 사용자가 매개변수에 허용되는 특수 값 중 하나를 입력하면 시스템이 이름 확인 규칙을 바이패스합니다.

- 단순명(*SNAME). 매개변수 값은 문자 스트링으로서 *NAME과 같은 명명 규칙을 따르지만, 마침표(.)는 허용되지 않습니다.
- 통신명(*CNAME). 매개변수 값은 문자 스트링으로서 *NAME과 같은 명명 규칙을 따르지만, 마침표(.)는 밑줄(_)은 허용되지 않습니다.
- 경로명(*PNAME). 매개변수 값은 문자 스트링입니다. 이는 어포스트로피로 묶을 수 있으며 LEN 매개변수에 지정된 길이의 문자 스트링으로 CPP에 전달됩니다. 값은 어포스트로피가 제거되고 좌측 정렬되며 여분은 공백으로 채워져서 전달됩니다.
- 총칭명(*GENERIC). 매개변수 값은 별표(*)로 끝나는 총칭명입니다. 이름이 별표로 끝나지 않으면 총칭명은 완전한 오브젝트명으로 간주됩니다. 총칭명은 별표 앞에 있는 문자로 시작하는 모든 오브젝트명을 식별합니다. 예를 들면, INV*는 INV, INVOICE, INVENTORY 등과 같이 INV로 시작하는 오브젝트명을 식별합니다. 총칭명은 총칭명에 나타난 문자로 시작하는 오브젝트명을 찾아내도록 명령 처리 프로그램(CPP: Command Processing Program)에 전달됩니다.
- 날짜(*DATE). 매개변수 값은 명령 처리 프로그램으로 전달되는 문자 스트링입니다. 문자 스트링은 cyymmdd(c = 세기 숫자, y = 년, m = 월, d = 일) 형식을 사용합니다. 시스템은 지정된 날짜의 연도에 따라 세기 숫자를 설정합니다. 지정된 연도가 4자리를 포함한 경우 시스템이 19로 시작되는 연도에 대해 세기 숫자를 0으로 설정합니다. 20으로 시작되는 연도에는 시스템이 세기 자릿수를 1로 설정합니다. 2자리로 지정된 연도의 경우 yy가 숫자 40-99에 해당되면 시스템이 세기 숫자를 0으로 설정합니다. 그러나 yy가 00-39의 숫자이면 시스템이 세기 숫자를 1로 설정합니다. 명령의 날짜 매개변수에는 반드시 날짜 형식(DATFMT) 작업 속성에 지정된 형식으로 날짜를 입력해야 합니다. 날짜 분리자(DATSEP) 작업 속성은 날짜가 입력될 때 사용될 분리문자(선택적)를 결정합니다. DATFMT와 DATSET 작업 속성을 변경하려면 CHGJOB(작업 변경) 명령을 사용하십시오. 프로그램은 1940년 1월 1일부터 2039년 12월 31일까지 범위에 속한 2자리 연도를 갖는 날짜를 읽습니다. 4자리 연도의 날짜는 1928년 8월 24일부터 2071년 5월 9일까지의 범위에 속해야 합니다.
- 시간(*TIME). 매개변수 값은 문자 스트링입니다. 시스템에서는 이 스트링을 hhmmss(h=시, m=분, s=초)의 형식으로 명령 처리 프로그램(CPP: Command Processing Program)에 전달합니다. 시간 분리자(TIMSEP) 작업 속성이 시간을 입력하기 위해 사용할 선택적 분리자를 결정합니다. TIMSEP 작업 속성을 변경하려면 CHGJOB(작업 변경) 명령을 사용하십시오.
- 16진(*HEX). 매개변수 값이 16진값입니다. 지정된 문자는 0-F까지의 문자여야 합니다. 값은 16진(EBCDIC) 문자(바이트당 2자리의 16진)로서 CPP에 전달되며, 우측 정렬되고 앞부분의 여분은 0으로 채워집니다. 값이 어포스트로피로 묶이면 숫자는 짝수여야 합니다.

- 0 요소(*ZEROELEM). 매개변수 값은 명령에 값이 지정될 수 없는 0 요소의 리스트로 간주됩니다. 이 매개변수 유형은 CPP가 값을 기대할지라도 매개변수 리스트에 값이 입력되지 못하도록 하기 위해 사용됩니다. 예를 들면, 두 개의 명령이 동일한 CPP를 사용할 때 하나의 명령이 매개변수의 리스트를 전달하고 나머지 다른 명령은 전달할 값을 갖지 못할 수 있습니다. 두 번째 명령에 대한 매개변수는 TYPE(*ZEROELEM)으로 정의되어야 합니다.
- 정수(*INT2 또는 *INT4). 매개변수 값은 2바이트 또는 4바이트의 부호화 2진수로 전달되는 정수입니다. CL 프로시듀어나 프로그램에 TYPE(*INT) 변수로 2진 숫자를 선언할 수 있습니다. 또한 TYPE(*CHAR)을 사용할 수 있으며 %BINARY 내장 기능을 사용해서 처리할 수 있습니다.
- 부호 없는 정수(*UINT2 또는 *UINT4). 매개변수 값은 2바이트 또는 4바이트의 비부호화 2진수로 전달되는 정수입니다. CL 프로시듀어나 프로그램에 TYPE(*UINT) 변수로 2진 숫자를 선언할 수 있습니다. 또한 TYPE(*CHAR)을 사용할 수 있으며 %BINARY 내장 기능을 사용해서 처리할 수 있습니다.
- 널(*NULL). 매개변수 값은 널 포인터이며, 이는 항상 위치 보유자로서 CPP에 전달됩니다. 이 매개변수 유형에 대해 유효한 PARM 키워드는 KWD, MIN 및 MAX 뿐입니다.
- 명령 스트링(*CMDSTR). 매개변수 값이 명령입니다. *CMDSTR 매개변수에 지정된 명령에 매개변수를 지정하기 위해 CL 변수를 사용할 수 있습니다. 그러나 전체 *CMDSTR 매개변수를 지정하기 위해서는 사용할 수 없습니다. 예를 들어, ″SBMJOB CMD(DSPLIB LIB(&LIBVAR))″는 CL 프로그램이나 프로시듀어에서 유효하지만 ″SBMJOB CMD(&CMDVAR)″는 유효하지 않습니다.
- 명령문 레이블. 명령문 레이블은 이 PARM문에 의해 정의되는 규정된 이름명 또는 혼합 리스트를 자세히 설명하는 일련의 QUAL문이나 ELEM문의 첫 번째 명령문을 식별합니다.

다음 매개변수 유형은 IBM 제공 명령에만 사용될 수 있습니다.

- 표현식(*X). 매개변수 값이 문자 스트링, 변수명 또는 숫자값입니다. 값에 숫자값, +, - 부호 또는 소수점만 들어 있으면 이 값은 숫자값으로 전달됩니다. 그렇지 않으면 문자 스트링으로 전달됩니다.
- 변수명(*VARNAME). 매개변수 값이 변수명이며, 이는 문자 스트링으로서 CPP에 전달됩니다. 값은 좌측 정렬되며 여분은 공백으로 채워집니다. 변수는 처리 시 실제 자료값을 참조하는 이름입니다. 변수명은 앰퍼샌드(&)가 앞에 오며 길이는 영숫자 10자(첫 번째 문자는 영문자여야 함)까지 가능합니다. 한 예로 &PARM을 들 수 있습니다. 변수명이 OS/400에서 사용되는 명명 규칙을 따르지 않는 경우 변수명을 어포스트로피로 묶어야 합니다.
- 명령(*CMD). 매개변수 값이 명령입니다. 예를 들면, CL 명령 IF가 THEN이라고 명명된 매개변수를 가질 때 이때 매개변수 THEN의 값은 또 다른 명령이 됩니다.

매개변수 값의 길이

다음 유형의 매개변수 값에 대해서는 길이(LEN 매개변수)도 지정할 수 있습니다. 날짜 또는 시간 매개변수 유형의 경우 날짜는 7자로, 시간은 6자로 지정됩니다. 다음은 사용자가 길이를 지정할 수 있는 각 매개변수 유형의 최대 길이와 디폴트 길이입니다.

자료 유형	최대 길이	디폴트 길이
*DEC	24(소수 자릿수는 9자리)	15(소수 자릿수는 5자리)
*LGL	1	1
*CHAR	5000	32
*NAME	256	10
*SNAME	256	10
*CNAME	256	10
*GENERIC	256	10
*HEX	256	1
*X	(256 24 9)	(1 15 5)
*VARNAME	11	11
*CMDSTR	20K	256
*PNAME	5000	32

여기서 최대 길이는 명령이 수행될 때 이러한 매개변수 유형에 허용되는 최대 길이입니다. 그러나 명령 정의문에서 문자 상수에 허용되는 최대 길이는 32자입니다. CONSTANT, DFT, VALUES, REL, RANGE, SPCVAL 및 SNGVAL 매개변수에 대해서도 이와 같은 제한사항이 적용됩니다. CL 명령에 대한 프롬트시, 사용이 가능한 입력 필드에는 지정 길이가 있습니다. 입력 필드 길이는 1-12, 17, 25, 32, 50, 80, 132, 256, 512자입니다. 특정 매개변수가 허용되지 않는 길이를 가진 경우 입력 필드에는 그 다음으로 큰 필드 길이가 표시됩니다. 프롬터는 512자를 초과할 수 있는 매개변수에 대해 512자의 입력 필드를 표시합니다.

디폴트 값

선택적 매개변수를 정의할 경우 사용자가 명령에 매개변수 값을 입력하지 않을 때 사용되는 값을 DFT 매개변수에 정의할 수 있습니다. 이 값을 *디폴트 값*이라고 합니다. 디폴트 값은 해당 매개변수의 값에 대한 모든 요구사항들(유형, 길이 및 특수 값 등과 같은)을 만족시켜야 합니다. 선택적 매개변수에 디폴트 값을 지정하지 않으면 아래의 디폴트 값이 사용됩니다.

자료 유형	디폴트
*DEC	0
*INT2	0
*INT4	0
*UINT2	0
*UINT4	0
*LGL	'0'
*CHAR	공백

자료 유형	디폴트
*NAME	공백
*SNAME	공백
*CNAME	공백
*GENERIC	공백
*DATE	0('F0')
*TIME	0('F0')
*ZEROELEM	0
*HEX	0('00')
*NULL	널(null)
*CMDSTR	공백
*PNAME	공백

매개변수 정의의 예

다음은 주문 입력 어플리케이션을 호출하는 명령에 매개변수 OETYPE을 정의하는 예입니다.

```
PARM   KWD(OETYPE)  TYPE(*CHAR)  RSTD(*YES) +
       VALUES(DAILY WEEKLY MONTHLY)  MIN(1) +
       PROMPT('Type of order entry:')
```

OETYPE 매개변수는 필수 매개변수이며(MIN 매개변수는 1), 그 값은 DAILY, WEEKLY 또는 MONTHLY로 제한됩니다. (RSTD 매개변수는 *YES입니다.) PROMPT 매개변수에는 매개변수의 프롬트 텍스트가 들어 있습니다. LEN 키워드가 지정되지 않고 TYPE(*CHAR)이 정의되었으므로 길이 32가 디폴트입니다.

자료 유형 및 매개변수에 대한 제한사항

다음 그림은 매개변수 유형에 따른 매개변수의 유효 조합을 보여줍니다. X는 조합이 유효함을 나타내며, 숫자는 주로 표의 아래에 명시된 제한사항을 참조하는 것입니다.

	LEN	RTNVAL	CONSTANT	RSTD	DFT	VALUES	REL	RANGE	SPCVAL	SNGVAL
*DEC	X	2	X	X	X	X	X	X	3	1
*LGL	X	2	X	X	X	X			3	1
*CHAR	X	2	X	X	X	X	X	X	3	1
*NAME	X		X	X	X	X	X	X	3	1
*SNAME	X		X	X	X	X	X	X	3	1
*CNAME	X		X	X	X	X	X	X	3	1
*PNAME	X	2	X	X	X	X	X	X	3	1
*GENERIC	X		X	X	X	X	X	X	3	1
*DATE			X	X	X	X	X	X	3	1
*TIME			X	X	X	X	X	X	3	1
*HEX	X		X	X	X	X	X	X	3	1
*ZEROELEM										
*INT2		2	X	X	X	X	X	X	3	1
*INT4		2	X	X	X	X	X	X	3	1

	LEN	RTNVAL	CONSTANT	RSTD	DFT	VALUES	REL	RANGE	SPCVAL	SNGVAL
*UINT2		2	X		X		X	X	3	1
*UINT4		2	X		X		X	X	3	1
*CMDSTR	X		X		X					
*NULL	X									
STMT LABEL			X		X					X

주:

1. MAX의 값이 1보다 큰 경우에만 유효합니다. 또한 CPP가 REXX 프로시듀어인 경우에는 To 값이 무시됩니다. REXX 프로시듀어 매개변수로서 전달된 값은 각 매개변수에 대해 입력된 값이거나 디폴트입니다.
2. CPP 명령이 REXX 프로시듀어이면 유효하지 않습니다.
3. CPP가 REXX 프로시듀어이면 To 값이 무시됩니다. REXX 프로시듀어 매개변수로서 전달된 값은 각 매개변수에 대해 입력된 값이거나 디폴트입니다.

	MIN	MAX	ALWUNPRT	ALWVAR	PGM	DTAARA	FILE	FULL	EXPR	VARY
*DEC	X	X		X		X				
*LGL	X	X		X		X	X	1		
*CHAR	X	X	X	X	X	X	X	X	X	1
*NAME	X	X		X	X	X	X	X	X	1
*SNAME	X	X		X	X	X	X	X	X	1
*CNAME	X	X		X	X	X	X	X	X	1
*PNAME	X	X	X	X	X	X	X	X	X	1
*GENERIC	X	X		X	X	X	X	X	X	1
*DATE	X	X		X		X				
*TIME	X	X		X					X	
*HEX	X	X		X				X	X	
*ZEROELEM	X	X								
*INT2	X	X		X					X	
*INT4	X	X		X					X	
*UINT2	X	X		X					X	
*UINT4	X	X		X					X	
*CMDSTR	2	3		4						1
*NULL	2	3								
STMT LABEL	X	X			X					

주:

1. CPP가 REXX 프로시듀어이면 매개변수가 무시됩니다.
2. TYPE(*NULL)에 대해 MIN 값이 1보다 클 수 없습니다.
3. TYPE(*NULL) 또는 TYPE(*CMDSTR)에 대해 MAX 값이 1보다 클 수 없습니다.
4. 이러한 매개변수 유형에 대해 ALWVAR 값은 무시됩니다. 매개변수 유형이 *CMDSTR일 경우 CL 변수는 허용되지 않습니다.

	PASSATR	PASSVAL	CASE	LISTDSPL	DSPINPUT
*DEC	1	X		X	X
*LGL	1	X		X	X
*CHAR	1	X	3	X	X
*NAME	1	X		X	X
*SNAME	1	X		X	X
*CNAME	1	X		X	X
*PNAME	1	X	3	X	X
*GENERIC	1	X		X	X
*DATE	1	X		X	X
*TIME	1	X		X	X
*HEX	1	X		X	X
*ZEROELEM					
*INT2	1	X		X	X
*INT4	1	X		X	X
*UINT2	1	X		X	X
*UINT4	1	X		X	X
*CMDSTR	1			X	X
*NULL					
STMT LABEL		2			

	CHOICE	CHOICEPGM	PMTCTL	PMTCTLPGM	PROMPT	INLPMTLEN
*DEC	X	X	X	X	X	
*LGL	X	X	X	X	X	
*CHAR	X	X	X	X	X	4
*NAME	X	X	X	X	X	4
*SNAME	X	X	X	X	X	4
*CNAME	X	X	X	X	X	4
*PNAME	X	X	X	X	X	4
*GENERIC	X	X	X	X	X	4
*DATE	X	X	X	X	X	
*TIME	X	X	X	X	X	
*HEX	X	X	X	X	X	4
*ZEROELEM						
*INT2	X	X	X	X	X	
*INT4	X	X	X	X	X	
*UINT2	X	X	X	X	X	
*UINT4	X	X	X	X	X	
*CMDSTR	X	X	X	X	X	4
*NULL						
STMT LABEL	X	X	X	X	X	X

주:

1. CPP가 REXX 프로시듀어이면 매개변수가 무시됩니다.
2. CPP가 REXX 프로시듀어이면 PASSVAL은 괄호 안에 공백이나 다른 문자가 없는 키워드를 전달합니다.
3. 대소문자(*MIXED)는 *CHAR 및 *PNAME 유형에만 허용됩니다.
4. INLPMTLEN(*PWD)를 *CHAR, *NAME, *SNAME, *CNAME 및 *PNAME 유형으로만 사용할 수 있습니다.

다음 그림은 유효한 매개변수 조합과 PARM, ELEM 및 QUAL문에 대한 제한사항을 보여줍니다. 예를 들어, LEN의 행과 DFT의 열의 교차점은 공백입니다. 따라서 제한사항이 없으며 LEN(XX)과 DFT(XX)의 조합은 유효합니다. 그러나 DFT 행과 CONSTANT 열의 교차점에 표시된 숫자 4는 표의 아래에 있는 주에서 설명된 제한사항의 번호입니다.

	LEN	RTNVAL	CONSTANT	RSTD	DFT	VALUES	REL	RANGE	SPCVAL	SNGVAL
LEN										
RTNVAL			1	1	1	1	1	1	1	1
CONSTANT		1			4					16
RSTD		1				7	9	9	7	7
DFT		1	4							
VALUES		1		7						
REL		1		9				9		
RANGE		1		9			9			
SPCVAL		1		7						
SNGVAL		1	21	7						
MIN					8					
MAX		2	2							10
ALWUNPRT										
ALWVAR		12								
PGM		1								
DTAARA		1								
FILE		1								
FULL		1								
EXPR		1	5							
VARY		3								
PASSATR		3								
PASSVAL		13						11		
CASE										
LISTDSPL										
CHOICE			14							
CHOICEPGM										
PMTCTL			15							
PMTCTLPGM			15							
PROMPT			6							
INLPMTLEN		17	17	17						

주:

1. RTNVAL 매개변수는 CONSTANT, RSTD, DFT, VALUES, REL, RANGE, SPCVAL, SNGVAL, PGM, DTAARA, FILE, FULL 또는 EXPR 매개변수와는 함께 사용할 수 없습니다. RTNVAL 매개변수는 CPP로서 REXX 프로시듀어를 사용한 명령에는 사용될 수 없습니다.
2. MAX 값은 1보다 작거나 같아야 합니다.
3. RTNVAL(*YES)과 PASSATR(*YES)이 지정되어 있으면 반드시 VARY(*YES)도 지정해야 합니다. RTNVAL(*YES)과 VARY(*YES)가 지정되면 *INT2 또는 *INT4를 사용해야 합니다. *INT2와 *INT4의 조합은 유효하지 않습니다.

4. CONSTANT와 DFT 매개변수는 상호 배타적입니다.
5. EXPR(*YES)과 CONSTANT 매개변수는 상호 배타적입니다.
6. PROMPT 매개변수는 허용되지 않습니다.
7. RSTD 매개변수가 지정되면 VALUES, SPCVAL 또는 SNGVAL 매개변수 중 반드시 하나를 지정해야 합니다.
8. MIN 값은 0이어야 합니다.
9. REL, RANGE 및 RSTD(*YES) 매개변수는 상호 배타적입니다.
10. MAX 값은 1보다 크거나, 매개변수 유형이 명령문 레이블이거나 또는 이 두 가지를 모두 만족시켜야 합니다.
11. 매개변수는 PASSVAL(*NULL) 매개변수로 정의된 매개변수를 참조하지 않을 수 있습니다. PASSVAL(*NULL)로 정의된 PARM문에서는 매개변수 간의 범위가 유효하지 않습니다.
12. RTNVAL(*YES)이 지정되면 ALWVAR(*NO)을 지정할 수 없습니다.
13. PASSVAL(*NULL)을 RTNVAL(*YES) 또는 0보다 큰 MIN 값과 함께 사용할 수 없습니다.
14. CHOICE와 CONSTANT 매개변수는 상호 배타적입니다.
15. CONSTANT는 PMTCTL 및 PMTCTLPGM 매개변수와 상호 배타적입니다.
16. SNGVAL 매개변수가 PARM문에 정의되는 경우 CONSTANT 매개변수는 ELEM/QUAL문에 정의할 수 없습니다.
17. INLPMTLEN 매개변수는 CONSTANT와 함께 사용할 수 없습니다. INLPMTLEN(*CALC)를 지정하거나 FULL(*YES), RTNVAL(*YES) 또는 RSTD(*YES)를 지정했으면 이를 디폴트로 사용해야 합니다.

	MIN	MAX	ALWUNPRT	ALWVAR	PGM	DTAARA	FILE	FULL	EXPR	VARY
LEN										
RTNVAL		2		8	1	1	1	1	1	3
CONSTANT		2							4	
RSTD										
DFT	5									
VALUES										
REL										
RANGE										
SPCVAL										
SNGVAL		7								
MIN		6								
MAX	6									
ALWUNPRT										
ALWVAR										
PGM						9	9			
DTAARA					9		9			
FILE						9	9			
FULL										

	MIN	MAX	ALWUNPRT	ALWVAR	PGM	DTAARA	FILE	FULL	EXPR	VARY
EXPR										
VARY										
PASSATR										3
PASSVAL	10									
CASE										
LISTDSPL										
CHOICE										
CHOICEPGM										
PMTCTL	11									
PMTCTLPGM										
PROMPT										
INLPMTLEN								12		

주:

1. RTNVAL 매개변수는 CONSTANT, RSTD, DFT, VALUES, REL, RANGE, SPCVAL, SNGVAL, PGM, DTAARA, FILE, FULL 또는 EXPR 매개변수와는 함께 사용할 수 없습니다. RTNVAL 매개변수는 CPP로서 REXX 프로시듀어를 사용한 명령에는 사용될 수 없습니다.
2. MAX 값은 1보다 작거나 같아야 합니다.
3. RTNVAL(*YES)과 PASSATR(*YES)이 지정되어 있으면 반드시 VARY(*YES)도 지정해야 합니다. RTNVAL(*YES)과 VARY(*YES)가 지정되면 *INT2 또는 *INT4를 사용해야 합니다. *INT2와 *INT4의 조합은 유효하지 않습니다.
4. EXPR(*YES)과 CONSTANT 매개변수는 상호 배타적입니다.
5. MIN 값은 0이어야 합니다.
6. MIN 매개변수에 지정된 값은 MAX 매개변수에 지정된 값보다 크지 않아야 합니다.
7. MAX 값은 1보다 크거나, 매개변수 유형이 명령문 레이블이거나 또는 이 두 가지를 모두 만족시켜야 합니다.
8. RTNVAL(*YES)이 지정되면 ALWVAR(*NO)을 지정할 수 없습니다.
9. PGM(*YES), DTAARA(*YES) 및 FILE 매개변수에 지정된 값(*NO는 제외)은 상호 배타적입니다.
10. PASSVAL(*NULL)을 RTNVAL(*YES) 또는 0보다 큰 MIN 값과 함께 사용할 수 없습니다.
11. PMTCTL에는 0보다 큰 MIN 값이 허용되지 않습니다.
12. INLPMTLEN(*CALC)을 지정하거나 FULL(*YES), RTNVAL(*YES) 또는 RSTD(*YES)를 지정했으면 이를 디폴트로 사용해야 합니다.

	PASSATR	PASSVAL	CASE	LISTDSPL	DSPINPUT
LEN					
RTNVAL	1	4			
CONSTANT				9	5
RSTD					
DFT					
VALUES					
REL					
RANGE		3			
SPCVAL					
SNGVAL					
MIN		4			
MAX					
ALWUNPRT					
ALWVAR					
PGM					
DTAARA					
FILE					
FULL					
EXPR					
VARY	1				
PASSATR					
PASSVAL					
CASE			10		
LISTDSPL				11	
CHOICE					
CHOICEPGM					
PMTCTL					
PMTCTLPGM					
PROMPT					
INLPMTLEN					

	CHOICE	CHOICEPGM	PMTCTL	PMTCTLPGM	PROMPT	INLPMTLEN
LEN						
RTNVAL						12
CONSTANT			7	7	2	12
RSTD						12
DFT						
VALUES						
REL						
RANGE						
SPCVAL						
SNGVAL						
MIN			8			
MAX						
ALWUNPRT						
ALWVAR						
PGM						
DTAARA						
FILE						
FULL						12
EXPR						

	CHOICE	CHOICEPGM	PMTCTL	PMTCTLPGM	PROMPT	INLPMTLEN
VARY						
PASSATR						
PASSVAL						
CASE						
LISTDSPL						
CHOICE		6				
CHOICEPGM	6					
PMTCTL						
PMTCTLPGM						
PROMPT						
INLPMTLEN						

주:

1. RTNVAL(*YES)과 PASSATR(*YES)이 지정되어 있으면 반드시 VARY(*YES)도 지정해야 합니다. RTNVAL(*YES)과 VARY(*YES)가 지정되면 *INT2 또는 *INT4를 사용해야 합니다. *INT2와 *INT4의 조합은 유효하지 않습니다.
2. PROMPT 매개변수는 허용되지 않습니다.
3. 매개변수는 PASSVAL(*NULL) 매개변수로 정의된 매개변수를 참조하지 않을 수 있습니다. PASSVAL(*NULL)로 정의된 PARM문에서는 매개변수 간의 범위가 유효하지 않습니다.
4. PASSVAL(*NULL)을 RTNVAL(*YES) 또는 0보다 큰 MIN 값과 함께 사용할 수 없습니다.
5. CHOICE와 CONSTANT 매개변수는 상호 배타적입니다.
6. CHOICE(*PGM)는 CHOICEPGM에 대한 이름을 요구합니다.
7. CONSTANT는 PMTCTL 및 PMTCTLPGM 매개변수와 상호 배타적입니다.
8. PMTCTL에는 0보다 큰 MIN 값이 허용되지 않습니다.
9. CONSTANT는 DSPINPUT(*NO) 및 DSPINPUT(*PROMPT)와 상호 배타적입니다.
10. CASE 매개변수는 PARM과 ELEM문에서만 유효합니다. 그러나 QUAL문에서는 유효하지 않습니다.
11. LISTDSPL 매개변수는 PARM문에서만 유효합니다.
12. INLPMTLEN 매개변수는 CONSTANT와 함께 사용할 수 없습니다. INLPMTLEN(*CALC)을 지정하거나 FULL(*YES), RTNVAL(*YES) 또는 RSTD(*YES)를 지정했으면 이를 디폴트로 사용해야 합니다.

매개변수 리스트 정의

매개변수가 단 한 개의 값 대신 여러 개의 값을 나열하도록 정의할 수 있습니다. 다음은 사용자가 정의할 수 있는 리스트의 유형입니다.

- 단순 리스트. 유형이 같은 한 개 이상의 값을 하나의 매개변수에 지정한 리스트
- 혼합 리스트. 개별 정의된 값의 세트를 하나의 매개변수에 지정한 리스트
- 리스트 안의 리스트. 하나의 매개변수에 두 번 이상 지정할 수 있는 리스트 또는 혼합 리스트 안의 값에 리스트를 지정할 수 있는 리스트

다음의 간단한 명령 소스는 여러 유형의 리스트에 관한 샘플입니다.

```
                            CMD        PROMPT('Example of lists command')


   /* THE FOLLOWING PARAMETER IS A SIMPLE LIST. IT WILL ACCEPT UP TO   */
   /* 5 NAMES.                                                        */
             PARM       KWD(SIMPLST) TYPE(*NAME) LEN(10) DFT(*ALL) +
                         SPCVAL((*ALL)) MAX(5) PROMPT('Simple list +
                         of up to 5 names')


   /* THE FOLLOWING PARAMETER IS A MIXED LIST OF 3 VALUES, EACH OF A   */
   /* DIFFERENT TYPE AND/OR LENGTH. EACH ELEMENT MAY NOT BE REPEATED.  */
             PARM       KWD(MXDLST) TYPE(MLSPEC) PROMPT('This is a +
                  mixed list of 3 val')
MLSPEC:     ELEM       TYPE(*CHAR) LEN(4) PROMPT('Elem 1 of 3')
            ELEM       TYPE(*DEC) LEN(3 0) PROMPT('Second of three')
            ELEM       TYPE(*CHAR) LEN(10) PROMPT('Last of three +
                         elements')


/* THE FOLLOWING PARAMETER IS A LIST WITHIN A LIST. IT CONTAINS A   */
/* LIST OF UP TO 2 ELEMENTS, WHICH MAY BE REPEATED UP TO 3 TIMES.   */
             PARM       KWD(LWITHINL1) TYPE(LWLSPECA) MAX(3) +
                          PROMPT('Repeatable list of 2 elements')
LWLSPECA:    ELEM         TYPE(*CHAR) LEN(10) PROMPT('1st part of +
                            repeatable list')
             ELEM         TYPE(*DEC) LEN(5 0) PROMPT('2nd part of +
                            repeatable list')


/* THE FOLLOWING PARAMETER IS A LIST WITHIN A LIST. IT CONTAINS A   */
/* LIST OF UP TO 2 ELEMENTS, THE FIRST OF WHICH MAY BE REPEATED     */
/* UP TO 3 TIMES.                                                  */
            PARM       KWD(LWITHINL2) TYPE(LWLSPECB) MAX(1) +
                         PROMPT('Repeated simple within mixed')
LWLSPECB:   ELEM       TYPE(*CHAR) LEN(10) MAX(3) PROMPT('Simple +
                         list within a list')
            ELEM       TYPE(*DEC) LEN(5 0) PROMPT('Single parm +
                         within a list')
```

다음 화면은 앞에서 샘플로 보여 주었던 명령에 대한 프롬트를 표시합니다.

```
                              리스트 명령 예(LSTEXAMPLE)

선택사항을 입력한 후 Enter 키를 누르십시오.

최대 다섯 개 이름의 단순 리스트  SIMPLST        *ALL
                        추가하려면 + 입력
세 값의 혼합 리스트               MXDLST
  세 요소 중 첫 번째 값 . . . . .
  세 요소 중 두 번째 값  . . . .
  세 요소 중 마지막 값 . . . . .
두 값의 반복 리스트               LWITHINL1
  반복 리스트의 첫 번째 부분  . .
  반복 리스트의 두 번째 부분 . .
                        추가하려면 + 입력
혼합 리스트내 단순 반복  . . . . LWITHINL2
  리스트내 단순 리스트 . . . . .
                        추가하려면 + 입력
  리스트내 단일 매개변수 . . . .


                                                                  맨 아래
F3=나감   F4=리스트  F5=화면정리  F12=취소  F13=프롬트되는 도움말
F24=추가 키
```

단순 리스트 정의

단순 리스트는 매개변수에 의해 지정된 유형에 대해 한 개 이상의 값을 허용할 수 있습니다. 예를 들면, 매개변수가 사용자명에 관한 것일 때 단순 리스트에서는 두 명 이상의 사용자명을 이 매개변수에 지정할 수 있습니다.

```
USER(JONES SMITH MILLER)
```

매개변수의 값이 단순 리스트이면 PARM문의 MAX 매개변수를 사용하여 리스트에 허용되는 요소의 최대 수를 지정합니다. 단순 리스트에서는 PARM문을 제외한 다른 명령 정의문을 지정할 필요가 없습니다.

다음은 표시장치 사용자가 5개까지의 사용자명을 지정할 수 있는 매개변수 USER를 정의한 예입니다(단순 리스트).

```
PARM      KWD(USER) TYPE(*NAME) LEN(10) MIN(0) MAX(5) +
          SPCVAL(*ALL) DFT(*ALL)
```

매개변수는 MIN(0)에 의해 지정된 선택적 매개변수이며 디폴트는 DFT(*ALL)에 의해 지정된 *ALL입니다.

단순 리스트 안의 요소가 명령 처리 프로그램으로 전달될 경우 형식은 CL, 고급 언어(HLL) 또는 REXX의 사용 여부에 따라 달라집니다. 다음 섹션에서는 이전 예에서 사용된 요소가 CL과 HLL을 사용하여 전달되는 방법을 설명합니다. REXX를 사용할 때의 차이점에 대한 설명은 363 페이지의 『단순 리스트에 대해 REXX 사용』 부분을 참조하십시오.

단순 리스트에 대해 CL 또는 HLL 사용

명령이 CL 또는 HLL을 사용하여 수행될 경우 단순 리스트의 요소가 다음과 같은 형식으로 명령 처리 프로그램에 전달됩니다.

RBAFN509-0

전달되는 값의 수는 길이가 2인 2진값으로 지정됩니다. 이 숫자는 지정될 수 있는 값의 수가 아닌 실제로 입력된(전달되는) 값의 수입니다. 값은 한 개의 매개변수 값이 전달될 때와 마찬가지로 매개변수 유형별로 전달됩니다(345 페이지의 『매개변수 정의』에 설명된 대로). 예를 들어, USER 매개변수에 두 개의 사용자명(BJONES와 TBROWN)이 지정되면 다음과 같이 전달됩니다.

RBAFN510-0

사용자명은 길이가 10자인 값으로 전달되고 좌측 정렬되며 여분은 공백으로 채워집니다.

단순 리스트가 전달될 때 명령에 지정된 갯수 만큼의 요소만이 전달됩니다. 전달된 마지막 요소 바로 뒤의 기억장치는 리스트의 일부가 아니며 리스트의 일부로서 참조되어서도 안됩니다. 따라서 명령 처리 프로그램이 단순 리스트를 처리할 때 처리될 요소의 수를 결정하는 데 전달된 요소의 수를 사용합니다.

362 페이지의 그림 13은 단순 리스트를 처리하기 위해 2진 내장 기능을 사용하는 CL 프로시듀어의 예를 보여줍니다.

```
            PGM PARM (...&USER..)
          .
          .
          .
          /* Declare space for a simple list of up to five */
          /* 10-character values to be received            */
          DCL VAR(&USER) TYPE(*CHAR) LEN(52)
          .
          DCL VAR(&CT)    TYPE(*DEC)  LEN(3 0)
          DCL VAR(&USER1) TYPE(*CHAR) LEN(10)
          DCL VAR(&USER2) TYPE(*CHAR) LEN(10)
          DCL VAR(&USER3) TYPE(*CHAR) LEN(10)
          DCL VAR(&USER4) TYPE(*CHAR) LEN(10)
          DCL VAR(&USER5) TYPE(*CHAR) LEN(10)
          .
          .
          .
          CHGVAR  VAR(&CT) VALUE(%BINARY(&USER 1 2))
          .
          IF (&CT > 0) THEN(CHGVAR &USER1 %SST(&USER 3 10))
          IF (&CT > 1) THEN(CHGVAR &USER2 %SST(&USER 13 10))
          IF (&CT > 2) THEN(CHGVAR &USER3 %SST(&USER 23 10))
          IF (&CT > 3) THEN(CHGVAR &USER4 %SST(&USER 33 10))
          IF (&CT > 4) THEN(CHGVAR &USER5 %SST(&USER 43 10))
          IF (&CT > 5) THEN(DO)
          /*  If CT is greater than 5, the values passed     */
          /*  is greater than the program expects, and error */
          /*  logic should be performed                      */
          .
          .
          .
ENDDO
             ELSE DO
          /* The correct number of values are passed */
          /* and the program can continue processing */
          .
          .
          .
ENDDO
             ENDPGM
```

그림 13. 단순 리스트의 예

이와 동일한 방법을 사용하여 CL 프로시듀어나 프로그램 안의 다른 리스트를 처리할 수 있습니다.

단순 리스트(simple list)에서 *ALL이나 *NONE과 같은 단일 값은 리스트 대신 명령에서 입력될 수 있습니다. 단일 값은 개별 값으로 전달됩니다. 이와 마찬가지로 매개변수에 값이 지정되지 않는 경우 디폴트 값(정의되어 있을 경우)이 리스트 안의 유일한 값으로서 전달됩니다. 예를 들어, 디폴트 값 *ALL이 USER 매개변수에 사용되면 다음과 같이 전달됩니다.

0001	*ALL

RBAFN511-0

*ALL은 길이가 10자이고 좌측 정렬되며 여분이 공백으로 채워진 값으로 전달됩니다.

단순 리스트 매개변수(선택적)에 대하여 디폴트 값이 정의되어 있지 않으면 다음과 같이 전달됩니다.

0000

RBAFN512-0

단순 리스트에 대해 REXX 사용

동일한 명령이 수행될 경우 단순 리스트의 요소가 다음과 같은 형식으로 인수 스트링에 있는 REXX 프로시듀어로 전달됩니다.

```
. . . USER(value1 value2  . . . valueN) . . .
```

여기서 valueN은 단순 리스트의 최종 값입니다.

예를 들면, USER 매개변수에 두 개의 사용자명(BJONES와 TBROWN)이 지정되면 다음과 같이 전달됩니다.

```
. . . USER(BJONES TBROWN) . . .
```

단순 리스트가 전달될 때 명령에 지정된 갯수 만큼의 요소만이 전달됩니다. 따라서 명령 처리 프로그램이 단순 리스트를 처리할 때 전달된 요소의 갯수를 사용하여 몇 개의 요소가 처리될 수 있는지를 결정합니다.

364 페이지의 그림 14에 나오는 REXX의 예는 362 페이지의 그림 13에 있는 프로시듀어와 동일한 결과를 생성합니다.

```
.
.
.
PARSE ARG . 'USER(' user ')' .
.
.
CT = WORDS(user)
IF CT > 0 THEN user1 = WORD(user,1) else user1 = '
IF CT > 1 THEN user2 = WORD(user,2) else user2 = '
IF CT > 2 THEN user3 = WORD(user,3) else user3 = '
IF CT > 3 THEN user4 = WORD(user,4) else user4 = '
IF CT > 4 THEN user5 = WORD(user,5) else user5 = '
IF CT > 5 THEN
           DO
    /* If CT is greater than 5, the values passed
       is greater than the program expects, and error
       logic should be performed */
.
.
.
  END
ELSE
           DO
    /* The correct number of values are passed
       and the program can continue processing */
  END
EXIT
```

그림 14. REXX 단순 리스트의 예

이와 같은 프로시듀어는 REXX 프로그램 안의 다른 리스트를 처리하는 데 사용될 수 있습니다.

단순 리스트(simple list)에서 *ALL이나 *NONE과 같은 단일 값은 리스트 대신 명령에서 입력될 수 있습니다. 단일 값은 개별 값으로 전달됩니다. 이와 마찬가지로 매개변수에 값이 지정되지 않는 경우 디폴트 값(정의되어 있을 경우)이 리스트 안의 유일한 값으로서 전달됩니다. 예를 들면, 디폴트 값 *ALL이 USER 매개변수에 사용되면 다음과 같이 전달됩니다.

```
. . . USER(*ALL) . . .
```

단순 리스트 매개변수(선택적)에 대하여 디폴트 값이 정의되어 있지 않으면 다음과 같이 전달됩니다.

```
. . . USER() . . .
```

REXX 프로시듀어에 관한 자세한 정보는 REXX/400 Programmer's Guide 및 REXX/400 Reference 를 참조하십시오.

혼합 리스트 정의

혼합 리스트는 일반적으로 다른 의미나 다른 유형을 나타내고 리스트에서 고정 위치에 놓이는 개별적으로 정의된 값 세트를 허용합니다. 예를 들면, LOG(4 0 *SECLVL)는 혼합 리스트를 지정할 수 있습니다. 첫 번째 값 4는 기록될 메세지 레벨을 식별합니다. 두 번째 값 0은 기록될 메세지의 최하위 심각도입니다. 세 번째 값 *SECLVL은 기록되는 정보의 양을 지정합니다(1차 및 2차 레벨 메세지 모두에 대해). 매개변수의 값이 혼합 리스트일 때는 리스트 안의 각 요소에 대해 요소(ELEM)문을 사용하여 개별적으로 정의해야 합니다.

관련된 PARM문의 TYPE 매개변수에는 리스트의 첫 번째 ELEM문을 참조하는 레이블을 지정해야 합니다.

```
                PARM    KWD(LOG)    TYPE(LOGLST) ...

LOGLST:  ELEM    TYPE(*INT2)      ...
         ELEM    TYPE(*INT2)      ...
         ELEM    TYPE(*CHAR)      LEN(7)
```

첫 번째 ELEM문만 레이블을 가질 수 있습니다. 리스트 안에 요소가 들어 있는 순서대로 ELEM문을 지정하십시오.

PARM문에서 MAX 매개변수의 값이 1보다 크고 TYPE 매개변수가 ELEM문을 참조하면 정의되는 매개변수는 리스트 내의 리스트입니다.

ELEM문에 지정할 수 있는 매개변수에는 TYPE, LEN, CONSTANT, RSTD, DFT, VALUES, REL, RANGE, SPCVAL, SNGVAL, MIN, MAX, ALWUNPRT, ALWVAR, PGM, DTAARA, FILE, FULL, EXPR, VARY, PASSATR, CHOICE, CHOICEPGM 및 PROMPT가 있습니다.

다음은 표시장치 사용자가 비교하기(혼합 리스트) 위해 비교 값 및 시작 위치를 지정하는 CMPVAL 매개변수를 정의한 예입니다.

```
            PARM   KWD(CMPVAL) TYPE(CMP) SNGVAL(*ANY) DFT(*ANY) +
              MIN(0)
CMP:  ELEM   TYPE(*CHAR) LEN(80) MIN(1)
      ELEM   TYPE(*DEC) LEN(2 0) RANGE(1 80) DFT(1)
```

혼합 리스트 안의 요소가 명령 처리 프로그램으로 전달될 경우 형식은 CL, 고급 언어(HLL) 또는 REXX의 사용 여부에 따라 달라집니다. 다음 섹션에서는 이전 예에서 사용된 요소가 CL과 HLL을 사용하여 전달되는 방법을 설명합니다. REXX를 사용할 때의 차이점에 대한 설명은 367 페이지의 『혼합 리스트에 대한 REXX 사용』 부분을 참조하십시오.

혼합 리스트에 대해 CL 또는 HLL 사용

명령이 CL 또는 HLL을 사용하여 수행되는 경우 혼합 리스트의 요소가 다음과 같은 형식으로 명령 처리 프로그램에 전달됩니다.

혼합 리스트 안의 값의 수	요소 1의 값	요소 2의 값	. . .	요소 n의 값

RBAFN513-0

혼합 리스트 안의 값의 갯수는 길이가 2인 2진값으로 전달됩니다. 이 값은 혼합 리스트에 정의되었던 값의 갯수를 표시합니다(명령에 실제로 입력된 값의 갯수가 아님). SNGVAL 매개변수가 입력되거나 디폴트 값으로서 전달될 때 이 값은 1이 될 수 있습니다. 사용자가 요소에 대한 값을 입력하지 않으면 디폴트 값이 전달됩니다. 요소는 한 개의 매개변수 값이 전달될 때처럼 유형별로 전달됩니다(345 페이지의 『매개변수 정의』 참조). 예를 들면, 앞의 예에서 사용자가 CMPVAL 매개변수에 대한 비교 값 QCMDI를 입력하지만, 시작 위치에 대한 값을 입력하지 않으면(디폴트 값이 1인 경우) 다음과 같이 전달됩니다.

0002	QCMDI	1

RBAFN514-0

자료 QCMDI는 길이가 80자인 값으로 전달되고 좌측 정렬되며 여분은 공백으로 채워집니다. 요소의 갯수는 길이가 2인 2진값으로 송신됩니다.

표시장치 사용자가 단일 값을 입력하거나 혼합 리스트에 대한 디폴트가 단일 값일 때 이 값은 리스트 안의 첫 번째 요소로서 전달됩니다. 예를 들면, 표시장치 사용자가 매개변수에 대한 단일 값으로 *ANY를 입력하면 다음과 같이 전달됩니다.

0001	*ANY

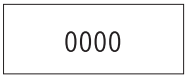
RBAFN515-0

*ANY는 길이가 80자인 값으로 전달되고 좌측 정렬되며 여분은 공백으로 채워집니다.

혼합 리스트는 CL 프로그램 안에서 처리될 수 있습니다. 단순 리스트와는 달리 리스트 안의 값의 갯수를 판별하기 위해 2진값을 테스트하지 않아도 됩니다. 이는 SNGVAL 매개변수가 명령 처리 프로그램으로 전달되지 않는 한 이 값은 제공된 혼합 리스트에 대해 동일하기 때문입니다. 이 경우 값은 1이 됩니다. 명령이 매개변수에 대한 단일 값으로 입력되면 그 하나의 값만이 전달됩니다. CL 프로시듀어에 있는 혼합 리스트를 처리하려면 서브스트링 내장 기능을 사용해야 합니다(제 2 장 참조).

어떤 경우에는 혼합 리스트에 대한 값의 갯수로 2진값 0000만이 전달됩니다. 선택적 매개변수에 대한 PARM문에서 디폴트 값이 정의되지 않고 리스트의 첫 번째 값이 요구

되면(MIN(1)), 매개변수 자체는 필수가 아니지만, 요소가 지정되면 첫 번째 요소는 필수입니다. 이 경우 매개변수에 값을 지정하지 않고 명령을 입력하면 다음과 같이 전달됩니다.



RBAFN512-0

다음은 이러한 매개변수의 예입니다.

```
            PARM  KWD(KWD1)    TYPE(E1) MIN(0)
E1:   ELEM  TYPE(*CHAR)  LEN(10)  MIN(1)
      ELEM  TYPE(*CHAR)  LEN(2)   MIN(0)
```

CL 프로시듀어가 이 매개변수를 처리하면 매개변수 값은 14자의 CL 변수로 수신될 수 있습니다. 처음 두 자를 다음 중 한 가지와 비교할 수 있습니다.

- %SUBSTRING 기능을 사용하여 16진 0000으로 초기화된 두 자의 변수
- %BINARY 내장 기능을 사용한 10진수 0

혼합 리스트에 대한 REXX 사용

명령이 REXX를 사용하여 수행되는 경우 혼합 리스트의 요소가 다음과 같은 형식으로 명령 처리 프로그램에 전달됩니다.

```
. . . CMPVAL(value1 value2  . . . valueN) . . .
```

여기서 valueN은 혼합 리스트의 최종 값입니다.

사용자가 요소에 대한 값을 입력하지 않으면 디폴트 값이 전달됩니다. 예를 들면, 앞의 예에서 사용자가 CMPVAL 매개변수에 대한 비교 값 QCMDI를 입력하지만 시작 위치에 대한 값을 입력하지 않으면(디폴트 값이 1인 경우) 다음과 같이 전달됩니다.

```
. . . CMPVAL(QCMDI 1)  . . .
```

후미 공백은 REXX 값으로 전달되지 않음을 유의하십시오.

표시장치 사용자가 단일 값을 입력하거나 단일 값이 혼합 리스트에 대한 디폴트 값이면, 그 값은 리스트 안의 첫 번째 요소로서 전달됩니다. 예를 들면, 표시장치 사용자가 매개변수에 대한 단일 값으로 *ANY를 입력하면 다음과 같이 전달됩니다.

```
. . . CMPVAL(*ANY) . . .
```

다시 한번 강조하지만 후미 공백은 REXX 값으로 전달되지 않습니다.

선택적 매개변수에 대해 PARM문에 디폴트 값이 정의되어 있지 않고, 리스트의 첫 번째 값이 필수인(MIN(1)) 경우 매개변수 자체는 필수 매개변수가 아닙니다. 그러나 어떤 요소라도 지정되는 경우 첫 번째 요소는 필수입니다. 이 경우 매개변수에 값을 지정하지 않고 명령을 입력하면 다음과 같이 전달됩니다.

```
. . . CMPVAL() . . .
```

리스트 안의 리스트 정의

리스트 안의 리스트는 다음과 같습니다.

- 하나의 매개변수에 대해 두 번 이상 지정될 수 있는 리스트(단순 또는 혼합 리스트)
- 혼합 리스트 안의 값으로서 지정될 수 있는 리스트

다음은 리스트 안의 리스트에 대한 예입니다.

```
STMT((START RESPND) (ADDDSP CONFRM))
```

바깥쪽의 괄호는 매개변수로 지정될 수 있는 리스트를 묶고(바깥쪽 리스트), 안쪽의 괄호는 각각 리스트 안의 리스트를 묶습니다(안쪽 리스트).

다음 예에서는 단순 리스트 안에 혼합 리스트가 정의되어 있습니다. 혼합 리스트가 지정되고 PARM문의 MAX 값이 1보다 큽니다. 따라서 MAX 매개변수에 지정된 횟수만큼 혼합 리스트를 지정할 수 있습니다.

```
                PARM KWD(PARM1) TYPE(LIST1) MAX(5)
LIST1:  ELEM TYPE(*CHAR) LEN(10)
        ELEM TYPE(*DEC) LEN(3 0)
```

이 예에서 두 개의 요소는 다섯 번까지 지정될 수 있습니다. 이 매개변수에 대해 값이 입력되면 결과는 다음과 같습니다.

```
PARM1((VAL1 1.0) (VAR2 2.0) (VAR3 3.0))
```

다음의 예에서는 단순 리스트가 혼합 리스트 안의 한 값으로 지정됩니다. 이 예에서는 ELEM문의 MAX 값이 1보다 큽니다. 따라서 MAX 매개변수에 지정된 횟수만큼 요소를 반복할 수 있습니다.

```
                PARM KWD(PARM2) TYPE(LIST2)
LIST2:  ELEM TYPE(*CHAR) LEN(10) MAX(5)
        ELEM TYPE(*DEC) LEN(3 0)
```

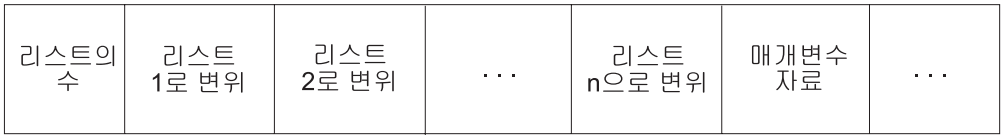
이 예에서 첫 번째 요소는 다섯 번까지 지정될 수 있으나, 두 번째 요소는 한 번만 지정될 수 있습니다. 이 매개변수에 값이 입력되면 결과는 다음과 같습니다.

```
PARM2((NAME1 NAME2 NAME3) 123.0)
```

리스트 안의 리스트가 명령 처리 프로그램으로 전달될 경우 형식은 CL, 고급 언어(HLL) 또는 REXX의 사용 여부에 따라 달라집니다. 다음 섹션에서는 요소가 CL과 HLL을 사용하여 전달되는 방법을 설명합니다. REXX를 사용할 때의 차이점에 대한 설명은 371 페이지의 『리스트 안의 리스트에 대해 REXX 사용』 부분을 참조하십시오.

리스트 안의 리스트에 대해 CL 또는 HLL 사용

명령이 CL 또는 HLL을 사용하여 수행되는 경우 리스트 안의 리스트가 다음과 같은 형식으로 명령 처리 프로그램에 전달됩니다.



RBAFN534-0

리스트의 수는 길이 2의 2진값으로 전달됩니다. 리스트의 수에 뒤이어 리스트의 변위가 전달됩니다(리스트에 입력된 값이 전달되는 것이 아님). 각 변위는 LISTDSPL 매개변수의 값에 따라서 길이 2 또는 길이 4의 2진값으로서 전달됩니다.

다음의 예는 매개변수 KWD2(단순 리스트 안의 혼합 리스트)에 대한 정의, 표시장치 사용자가 매개변수를 지정하는 방법, 그리고 전달되는 내용을 보여줍니다. 매개변수 정의는 다음과 같습니다.

```
       PARM    KWD(KWD2)    TYPE(LIST) MAX(20) MIN(0) +
               DFT(*NONE)  SNGVAL(*NONE)  LISTDSPL(*INT2)
LIST:  ELEM    TYPE(*CHAR)  LEN(10) MIN(1)       /*From value*/
       ELEM    TYPE(*CHAR)  LEN(5) MIN(0)        /*To value*/
```

표시장치 사용자가 KWD2 매개변수를 다음과 같이 입력할 경우:

```
KWD2((A B))
```

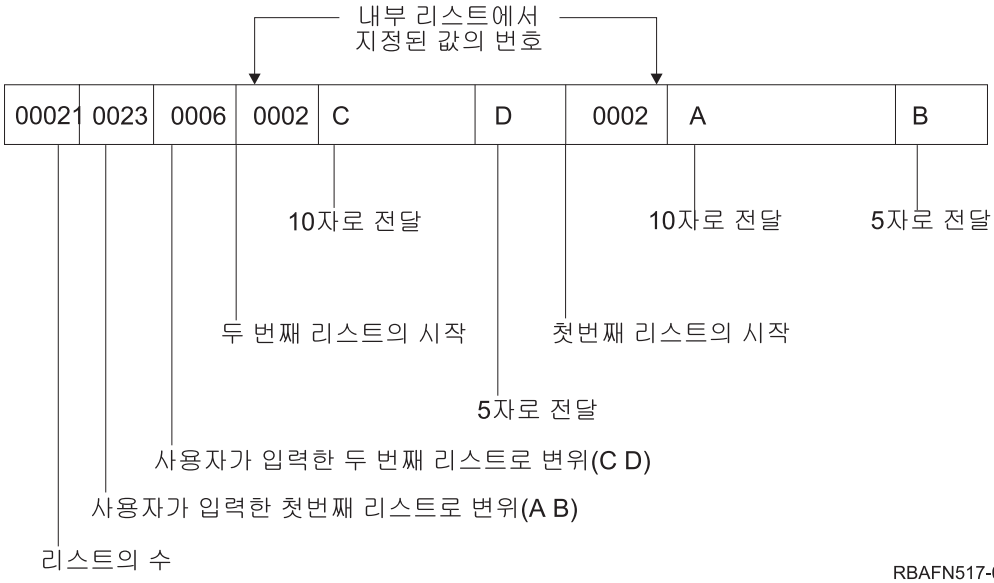
명령 처리 프로그램으로 다음과 같이 전달됩니다.

RBAFN516-0

표시장치 사용자가 다음과 같이 입력할 경우:

```
KWD2((A B) (C D))
```

명령 처리 프로그램으로 다음과 같이 전달됩니다.



리스트 안의 리스트는 n(표시장치 사용자가 최종 입력한 리스트)에서 1(표시장치 사용자가 첫 번째로 입력한 리스트)의 순서로 명령 처리 프로그램에 전달됩니다. 그러나 변위는 1부터 n의 순으로 전달됩니다.

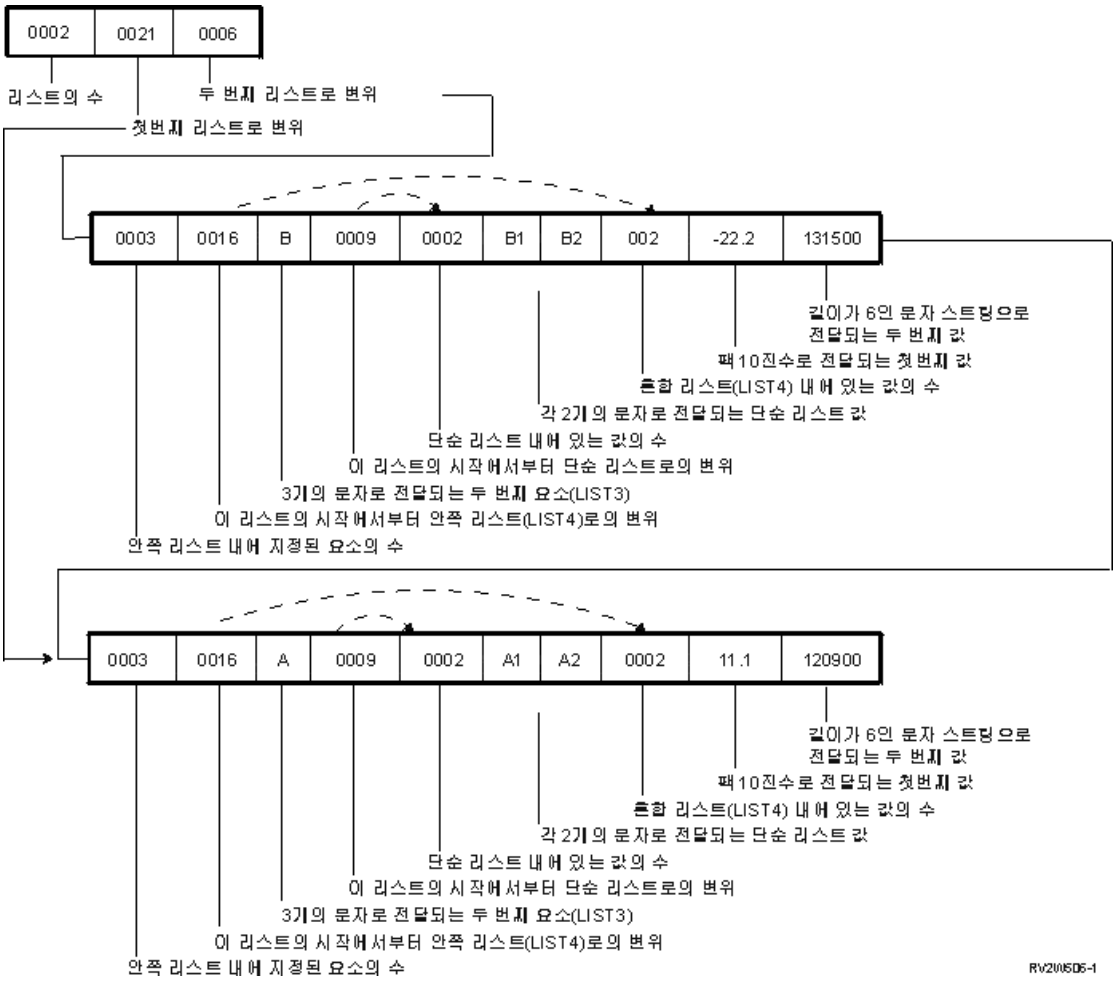
다음은 리스트 안의 리스트에 대한 보다 복잡한 예입니다. 매개변수 정의는 다음과 같습니다.

```
                PARM    KWD(PARM1)  TYPE(LIST3)  MAX(25)
LIST3:  ELEM    TYPE(LIST4)
        ELEM    TYPE(*CHAR)  LEN(3)
        ELEM    TYPE(*NAME)  LEN(2)  MAX(5)
LIST4:  ELEM    TYPE(*DEC) LEN(7 2)
        ELEM    TYPE(*TIME)
```

표시장치 사용자가 PARM1 매개변수를 다음과 같이 입력할 경우:

```
PARM1(((11.1  120900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

명령 처리 프로그램으로 다음과 같이 전달됩니다.



리스트 안의 리스트에 대해 REXX 사용

명령이 REXX를 사용하여 수행될 경우 매개변수의 값이 입력될 때처럼 리스트 안의 리스트가 명령 처리 프로그램으로 전달됩니다. 후미 공백은 전달되지 않습니다.

다음 예는 매개변수 KWD2(단순 리스트 안의 혼합 리스트)에 대한 정의, 표시장치 사용자가 이 매개변수를 지정하는 방법 및 전달되는 내용을 보여줍니다. 매개변수 정의는 다음과 같습니다.

```
       PARM    KWD(KWD2)     TYPE(LIST) MAX(20) MIN(0) +
               DFT(*NONE)  SNGVAL(*NONE)
LIST:  ELEM    TYPE(*CHAR)  LEN(10) MIN(1)       /*From value*/
       ELEM    TYPE(*CHAR)  LEN(5) MIN(0)        /*To value*/
```

표시장치 사용자가 KWD2 매개변수를 다음과 같이 입력할 경우:

```
KWD2((A B))
```

명령 처리 프로그램으로 다음과 같이 전달됩니다.

```
KWD2(A B)
```

표시장치 사용자가 다음과 같이 입력할 경우:

```
KWD2((A B) (C D))
```

명령 처리 프로그램으로 다음과 같이 전달됩니다.

```
KWD2((A B) (C D))
```

다음은 리스트 안의 리스트에 대한 보다 복잡한 예입니다. 매개변수 정의는 다음과 같습니다.

```
               PARM    KWD(PARM1)  TYPE(LIST3)  MAX(25)
LIST3:  ELEM   TYPE(LIST4)
        ELEM   TYPE(*CHAR)  LEN(3)
        ELEM   TYPE(*NAME)  LEN(2)  MAX(5)
LIST4:  ELEM   TYPE(*DEC) LEN(7 2)
        ELEM   TYPE(*TIME)
```

표시장치 사용자가 PARM1 매개변수를 다음과 같이 입력할 경우:

```
PARM1(((11.1  12D900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

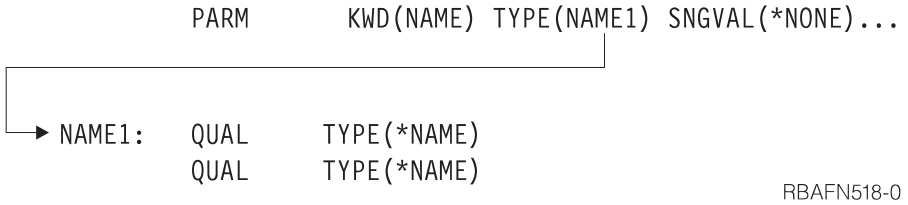
명령 처리 프로그램으로 다음과 같이 전달됩니다.

```
PARM1(((11.1  12D900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

규정된 이름 정의

규정된 이름은 오브젝트가 저장되는 라이브러리명이 뒤에 오는 오브젝트의 이름입니다. 매개변수 값이나 리스트 항목이 규정된 이름명이면, QUAL(규정자)문을 사용하여 이름을 개별적으로 정의해야 합니다. 규정된 이름의 각 부분들은 반드시 QUAL문으로 정의되어야 합니다. 규정된 이름의 각 부분들은 규정된 이름에 나타나는 순서대로 설명되어야 합니다. 첫 번째 QUAL문에는 *NAME이나 *GENERIC를 지정해야 합니다. 관련된 PARM 또는 ELEM문은 규정된 이름을 위한 첫 번째 QUAL문을 참조하는 레이블을 나타냅니다.

다음의 명령 정의문은 가장 보편적인 규정된 이름을 정의합니다. 규정된 오브젝트는 오브젝트가 들어 있는 라이브러리명과 그 뒤에 오는 오브젝트명으로 이루어집니다. QUAL문은 규정된 이름에 나오는 순서대로 표시되어야 합니다.



RBAFN518-0

QUAL문에 지정될 수 있는 매개변수 중 대부분은 PARM문에 대해 서술한 매개변수와 같습니다(345 페이지의 『매개변수 정의』를 참조). 그러나 TYPE 매개변수에는 다음 값만 지정할 수 있습니다.

- *NAME

- *GENERIC
- *CHAR
- *INT2
- *INT4

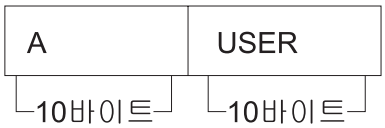
규정된 이름이 명령 처리 프로그램으로 전달될 경우 형식은 CL, 고급 언어(HLL) 또는 REXX의 사용 여부에 따라 달라집니다. 다음 섹션에서는 규정된 이름이 CL과 HLL을 사용하여 전달되는 방법을 설명합니다. REXX를 사용할 때의 차이점에 대한 설명은 375 페이지의 『규정된 이름에 대해 REXX 사용』 부분을 참조하십시오.

규정된 이름에 대해 CL 또는 HLL 사용

규정된 이름은 CL 또는 HLL을 사용할 때 다음과 같은 형식으로 명령 처리 프로그램에 전달됩니다.

RBAFN519-0

예를 들면, 표시장치 사용자가 이전에 정의된 QUAL문에 대해 NAME(USER/A)을 입력하면 이름이 명령 처리 프로그램에 다음과 같이 전달됩니다.



RBAFN520-0

규정자는 한 개의 매개변수 값이 전달될 때처럼 유형과 길이별로 연속하여 명령 처리 프로그램에 전달됩니다(345 페이지의 『매개변수 정의』 참조). 분리자(/)는 전달되지 않습니다. 이는 단일 매개변수, 혼합 리스트의 요소 또는 규정된 이름의 단순 리스트 중 어느 것이 전달되든 관계 없이 적용됩니다.

표시장치 사용자가 규정된 이름에 단일 값을 입력하면 전달되는 값의 길이는 규정된 이름의 각 부분들의 길이의 총계가 됩니다. 예를 들면, 규정된 이름에 길이가 각각 10인 두 개의 값을 정의하였을 때 표시장치 사용자가 단일 값을 입력하면 전달되는 단일 값은 좌측 정렬되어 오른쪽에 공백이 채워지며, 그 결과 총 20자가 전달됩니다. 표시장치 사용자가 단일 값으로서 *NONE을 입력하면 그 다음 20자가 전달됩니다.

RBAFN521-0

규정된 이름은 다음 예에 나오는 것처럼 서브스트링 내장 기능을 사용하여 CL 프로그램에서 처리될 수 있습니다.

서브스트링 내장 기능(%SUBSTRING 또는 %SST)은 규정된 이름을 두 개의 값으로 분리하는 데 사용됩니다.

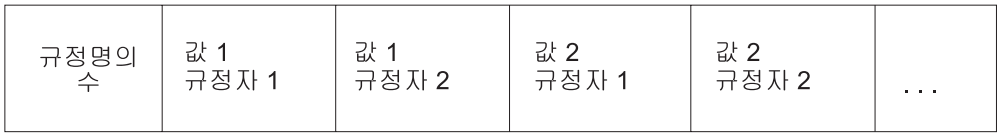
```
PGM PARM(&QLFDNAM)
DCL &QLFDNAM TYPE(*CHAR) LEN(20)
DCL &OBJ TYPE(*CHAR) LEN(10)
DCL &LIB TYPE(*CHAR) LEN(10)
CHGVAR &OBJ %SUBSTRING(&QLFDNAM 1 10) /* First 10 */
CHGVAR &LIB %SST(&QLFDNAM 11 10) /* Second 10 */
 .
 .
 .
          ENDPGM
```

그런 후 적합한 CL 구문(syntax)으로 규정된 이름을 지정할 수 있습니다. 한 예로, OBJ(&LIB/&OBJ)를 들 수 있습니다.

다음 방법을 사용하여 규정된 이름을 두 개의 값으로 분리시킬 수도 있습니다.

```
PGM PARM(&QLFDNAM)
DCL &QLFDNAM TYPE(*CHAR) LEN(20)
CHKOBJ (%SST(&QLFDNAM 11 10)/%SST(&QLFDNAM 1 10)) *PGM
 .
 .
 .
          ENDPGM
```

규정된 이름의 단순 리스트는 다음과 같은 형식으로 명령 처리 프로그램에 전달됩니다.



RBAFN522-0

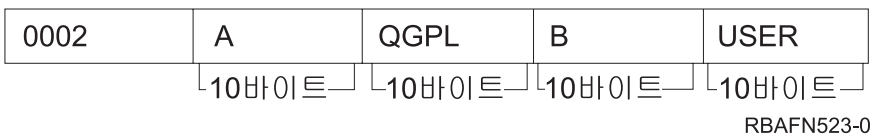
예를 들어, MAX(3)가 다음과 같이 NAME 매개변수에 대한 PARM문에 추가되는 것으로 가정하십시오.

```
                PARM   KWD(NAME) TYPE(NAME1) SNGVAL(*NONE) MAX(3)
NAME1:  QUAL   TYPE(*NAME)
        QUAL   TYPE(*NAME)
```

표시장치 사용자가 다음을 입력하면

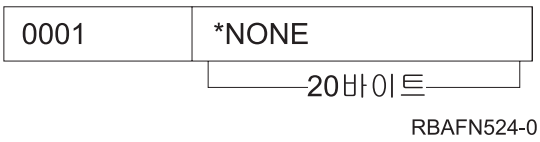
```
NAME(QGPL/A USER/B)
```

이름 매개변수가 다음과 같이 명령 처리 프로그램으로 전달됩니다.



RBAFN523-0

표시장치 사용자가 단일 값 NAME(*NONE)을 입력하면 이름 매개변수가 다음과 같이 전달됩니다.



규정된 이름에 대해 REXX 사용

명령이 REXX를 사용하여 수행되는 경우 매개변수의 값이 입력될 때처럼 규정된 이름이 명령 처리 프로그램으로 전달됩니다. 후미 공백은 전달되지 않습니다.

예를 들면, 표시장치 사용자가 이 섹션에서 이전에 정의된 QUAL문에 대해 다음을 입력하면

```
NAME(USER/A)
```

규정된 이름이 다음과 같은 형식으로 명령 처리 프로그램에 전달됩니다.

```
NAME(USER/A)
```

규정자는 한 개의 매개변수 값이 전달될 때처럼 유형과 길이별로 연속하여 명령 처리 프로그램에 전달됩니다(345 페이지의 『매개변수 정의』 참조).

표시장치 사용자가 단일 값으로서 *NONE을 입력하면 그 다음 20자가 전달됩니다.

```
NAME(*NONE)
```

다음 예는 표시장치 사용자가 규정된 이름의 단순 리스트를 입력하는 방법을 보여줍니다.

```
NAME(QGPL/A USER/B)
```

REXX를 사용하여 이름 매개변수가 다음과 같이 명령 처리 프로그램에 전달됩니다.

```
NAME(QGPL/A USER/B)
```

종속 관계 정의

매개변수들 간에 필요한 관계가 존재하고 명령이 수행될 때 매개변수 값을 검사하는 경우 종속(DEP) 명령문을 사용하여 그 관계를 정의하십시오. DEP문을 사용하면 다음에 열거한 기능들을 수행할 수 있습니다.

- PARM문에 정의된 매개변수 관계가 참이어야 하기 전에 반드시 참이 되어야 하는 제어 조건을 지정합니다(CTL).
- CTL에 의해 정의된 제어 조건이 참인 경우 테스트를 필요로 하는 매개변수 관계를 지정합니다(PARM).
- 제어 조건이 참일 때 반드시 참이어야 하는 관련 PARM문에 정의된 매개변수 관계의 수를 지정합니다(NBRTRUE).

- 매개변수 종속성이 만족스럽지 않을 경우 시스템이 표시장치 사용자에게 송신할 메세지 파일에서 오류 메세지의 ID를 지정합니다.

다음 예에서 표시장치 사용자가 TYPE(LIST) 매개변수를 지정하는 경우 ELEMLIST 매개변수도 반드시 지정해야 합니다.

```
DEP CTL(&TYPE *EQ LIST) PARM(ELEMLIST)
```

다음 예에서 매개변수 &WRITER가 매개변수 &NEWWTR와 같아서는 안됩니다. 이 조건이 참이 아니면(not true), 메세지 USR0001이 표시장치 사용자에게 보내집니다.

```
DEP CTL(*ALWAYS) PARM((&WRITER *NE &NEWWTR)) MSGID(USR0001)
```

다음 예에서 표시장치 사용자가 FILE 매개변수를 지정할 때에는 VOL과 LABEL 매개변수도 함께 지정해야 합니다.

```
DEP CTL(FILE) PARM(VOL LABEL) NBRTRUE(*EQ 2)
```

가능한 선택사항 및 값

프롬터는 프롬트 화면에서 입력 필드의 오른쪽에 매개변수에 대한 가능한 선택사항을 표시합니다. 표시되는 텍스트는 자동으로 작성되어 명령 정의 소스에 지정될 수 있습니다. 또는 나감 프로그램에 의해 동적으로 작성될 수 있습니다. PARM, ELEM 또는 QUAL문에 가능한 선택사항을 설명하는 텍스트를 정의할 수 있지만 화면 형식상의 제한 때문에 12자 이하의 필드 길이, 10자 이하의 필드 길이를 가진 값에 대해서만 텍스트가 표시됩니다(그룹 안의 첫 번째 규정자는 제외).

가능한 선택사항용 텍스트는 CHOICE 매개변수에 의해 정의됩니다. 이 매개변수에 대한 디폴트는 *VALUES이며 TYPE, RANGE, VALUES, SPCVAL 및 SNGVAL 키워드에 지정된 값으로부터 텍스트가 자동으로 작성됨을 표시합니다. 텍스트는 30자로 제한됩니다. 이 크기보다 더 많은 값들이 남아 있으면 생략 기호(...)가 텍스트 끝에 추가되어 텍스트가 완전하지 않음을 나타냅니다.

가능한 선택사항이 표시되지 않도록(*NONE) 지정하거나 텍스트 스트링을 표시되도록 지정하거나 CRTCMD 명령의 PMTFILE 매개변수에서 지정된 메세지 파일에서 검색되는 텍스트 메세지의 ID를 지정할 수 있습니다.

프롬팅중에는 나감 프로그램을 수행시켜 가능한 선택사항 텍스트를 제공하도록 지정할 수도 있습니다. 예를 들면, 사용자에게 현재 시스템에 존재하는 오브젝트 리스트를 보여주려 할 때 위와 같이 지정할 수 있습니다. 매개변수 값 지정 화면에 나오는 허용 가능한 값들의 리스트를 제공하는 데 있어서도 동일한 나감 프로그램을 사용할 수 있습니다. 나감 프로그램을 지정하려면 CHOICE 매개변수에 *PGM을, 그리고 PARM, ELEM 또는 QUAL문의 CHOICEPGM 매개변수에 나감 프로그램의 규정된 이름을 지정하십시오.

나감 프로그램은 다음 두 개의 매개변수를 받아들여야 합니다.

- **매개변수 1**: 프롬터에 의해 선택사항 프로그램으로 전달되는 21바이트 필드로서 다음 내용이 포함됩니다.

  위치 **설명**

  1-10 명령어. 프로그램을 수행시키기 위해 처리할 명령어를 지정합니다.

  11-20 키워드명. 가능한 선택사항이나 허용 가능한 값을 요구할 키워드를 지정합니다.

  21 프롬터가 요구한 자료의 유형을 나타내는 C 또는 P 문자. 문자 C는 이 필드가 가능한 선택사항에 대해 텍스트를 리턴시킬 30바이트의 필드임을 나타냅니다. 문자 P는 이 필드가 허용 가능한 값 리스트를 리턴시킬 2000바이트의 필드임을 나타냅니다.

- **매개변수 2**: 다음 중 하나를 리턴하기 위한 30바이트 또는 2000바이트의 필드
  - 첫 번째 매개변수의 C가 21바이트로 되어 있는 경우 이는 가능한 선택사항에 대해 텍스트를 리턴시킬 것임을 나타냅니다. 뿐만 아니라, 이것은 프로그램이 프롬트 화면에서 입력 필드의 오른쪽에 텍스트를 위치시키는 30바이트 필드입니다.
  - P가 첫 번째 매개변수(허용 가능한 값 리스트가 리턴되어야 함을 나타내는)의 21바이트에 있다면, 이것은 프로그램이 리스트를 위치시킬 2000바이트 필드입니다. 리스트의 첫 번째 2바이트에는 리스트 안의 항목 수(2진수로)가 들어 있어야 합니다. 이 값 뒤에는 2바이트 2진수 길이로 구성된 항목이 나오고 그 뒤로 1부터 34바이트 길이의 값이 옵니다.

    2진수 0 값이 첫 번째 2바이트로 리턴되면 허용 가능한 값이 표시되지 않습니다.

    2진 음수 값이 첫 번째 2바이트로 리턴되면 허용 가능한 값의 리스트가 명령으로부터 선택됩니다.

프로그램이 호출될 때 예외가 발생하면 가능한 선택사항 텍스트는 공백으로 남겨지며 허용 가능한 값의 리스트가 명령으로부터 선택됩니다.

프롬트 제어 사용

프롬트 제어 스펙을 사용하면 프롬팅 시 명령에 대해 표시될 매개변수를 제어할 수 있습니다. 이 제어는 보고자 하는 매개변수만 표시함으로써 명령에 대한 프롬팅을 간단하게 만들 수 있습니다.

한 매개변수가 다른 매개변수에 지정된 값에 따라 표시되도록 지정할 수 있습니다. 이 스펙은 제어 매개변수가 특정 값을 가지고 있을 경우에 한해, 한 매개변수가 의미를 가지게 될 때 유용합니다.

프롬트되는 동안 기능 키를 눌러서 매개변수를 추가로 요구할 때만 프롬트할 매개변수가 선택되도록 지정할 수 있습니다. 이 스펙은 사용자가 거의 지정하지 않는 매개변수에 대해 사용될 수 있습니다. 이는 디폴트가 사용되거나 이 매개변수가 거의 사용되지 않는 기능을 제어하기 때문입니다.

프롬트 제어가 지정된 명령의 매개변수를 모두 보려면 프롬트되는 동안 F9 키를 눌러서 모든 매개변수가 표시되도록 요구할 수 있습니다.

조건 프롬팅

다음의 경우 사용자에게 명령을 입력하도록 프롬팅할 때 다른 매개변수들에 의해 조건이 지정되는 매개변수가 표시됩니다.

- 제어 매개변수에 지정된 값에 의해 선택될 경우
- 제어 매개변수에 지정된 값에 오류가 있을 경우
- 조건 매개변수에 값이 지정되었을 경우
- 프롬트되는 동안 모든 매개변수가 표시되도록 요구하기 위해 기능 키를 눌렀을 경우

조건 매개변수가 프롬트되고 제어 매개변수에 아직 값이 지정되지 않았을 때에는 이전에 선택된 모든 매개변수가 표시됩니다. 사용자가 Enter 키를 누르면, 제어 매개변수는 조건 매개변수를 표시할 것인지의 여부를 결정하기 위해 테스트됩니다.

명령 정의 소스에 조건 프롬팅을 지정하려면 다른 매개변수에 의해 조건이 지정되는 각 매개변수에 대해 PARM문의 PMTCTL 매개변수에서 레이블명을 지정하십시오. 지정된 레이블은 제어 매개변수를 지정하는 그리고 프롬팅할 매개변수를 선택할 때 테스트되는 조건을 지정하는 PMTCTL문에 정의되어야 합니다. 둘 이상의 PARM문이 동일한 레이블을 참조할 수 있습니다.

PMTCTL문에서는 제어 매개변수명, 테스트되는 하나 이상의 조건, 그리고 프롬팅할 조건 매개변수를 선택하기 위해 참이어야 하는 조건의 수 등을 지정합니다. 제어 매개변수에 특수 값 맵핑이 있으면, PMTCTL문에 입력된 값이 To 값이어야 합니다. 제어 매개변수가 리스트이거나 규정된 이름이면, 첫 번째 리스트 항목이나 규정자만이 비교됩니다.

다음 예에서 매개변수 OUTFILE과 OUTMBR은 OUTPUT 매개변수에 대해 *OUTFILE이 지정된 경우에만 선택되고, 매개변수 OUTQ는 OUTPUT 매개변수에 대해 *PRINT가 지정된 경우에만 선택됩니다.

```
     PARM OUTPUT TYPE(*CHAR) LEN(1) DFT(*) RSTD(*YES) +
       SPCVAL((*) (*PRINT P) (*OUTFILE F))
PARM OUTFILE TYPE(Q1) PMTCTL(OUTFILE)
PARM OUTMBR TYPE(*NAME) LEN(10) PMTCTL(OUTFILE)
PARM OUTLINK TYPE(*CHAR) LEN(10)
PARM OUTQ TYPE(Q1) PMTCTL(PRINT)
Q1: QUAL TYPE(*NAME) LEN(10)
```

```
    QUAL TYPE(*NAME) LEN(10) SPCVAL(*LIBL) DFT(*LIBL)
OUTFILE: PMTCTL CTL(OUTPUT) COND((*EQ F)) NBRTRUE(*EQ 1)
PRINT:   PMTCTL CTL(OUTPUT) COND((*EQ P)) NBRTRUE(*EQ 1)
```

이전 예에서 OUTMBR 매개변수에 대한 조건이 테스트된 후에 OUTLINK 매개변수가 프롬트됩니다. 어떤 경우에는 OUTMBR 매개변수가 테스트되기 전에 OUTLINK 매개변수가 프롬트되어야 합니다. 프롬트 순서를 다르게 지정하려면 명령 정의 소스에 있는 매개변수를 재배열하거나 PARM문에서 OUTLINK 매개변수에 대해 PROMPT 키워드를 사용해야 합니다.

레이블은 PMTCTL문 그룹을 참조할 수 있습니다. 따라서 둘 이상의 제어 매개변수로 매개변수를 조건 지정할 수 있습니다. PMTCTL문의 그룹을 지정하려면 그룹의 첫 번째 명령문에 레이블을 입력하십시오. 그룹 안의 PMTCTL문 사이에는 다른 명령문이 위치할 수 없습니다.

그룹 안에 있는 명령문 간의 논리 관계를 지정하려면 LGLREL 매개변수를 사용하십시오. 그룹 안의 첫 번째 PMTCTL문에서는 LGLREL 매개변수를 사용할 수 없습니다. 그 다음의 PMTCTL문에 있어서 LGLREL 매개변수는 PMTCTL문이나 선행되는 명령문에 대해 논리 관계(*AND 또는 *OR)를 지정합니다. 그룹 안의 명령문에 대한 논리 관계는 *AND 및 *OR 관계의 어떤 조합이라도 가능합니다. (*AND 관계가 먼저 검사된 후 *OR 관계가 검사됩니다.)

다음 예는 논리 관계를 사용하여 PMTCTL문을 그룹화하는 방법을 보여줍니다. 이 예에서 다음 조건 중 하나가 존재하면 매개변수 P3이 선택됩니다.

- P1에 *ALL이 지정됨
- P1에는 *SOME이, P2에는 *ALL이 지정됨
- P1에는 *NONE이 지정되고, P2에는 *ALL이 지정되지 않음

```
PARM P1 TYPE(*CHAR) LEN(5) RSTD(*YES) VALUES(*ALL *SOME *NONE)
PARM P2 TYPE(*NAME) LEN(10) SPCVAL(*ALL)
PARM P3 TYPE(*CHAR) LEN(10) PMTCTL(PMTCTL1)
PMTCTL1:PMTCTL CTL(P1) COND((*EQ *ALL))
        PMTCTL CTL(P1) COND((*EQ *SOME)) LGLREL(*OR)
        PMTCTL CTL(P2) COND((*EQ *ALL)) LGLREL(*AND)
        PMTCTL CTL(P1) COND((*EQ *NONE)) LGLREL(*OR)
        PMTCTL CTL(P2) COND((*NE *ALL)) LGLREL(*AND)
```

제어 매개변수가 테스트되기 전에 제어 매개변수의 추가 처리를 수행하기 위해 나감 프로그램을 지정할 수 있습니다. 나감 프로그램은 다음 사항에 근거하여 프롬팅의 조건을 지정하는 데 사용될 수 있습니다.

- 오브젝트의 유형 또는 다른 속성
- 첫 번째가 아닌 리스트 항목 또는 규정자
- 전체 리스트 또는 규정된 이름

나감 프로그램을 지정하려면 제어 매개변수에 대한 PARM문의 PMTCTLPGM 매개변수에 프로그램의 규정된 이름을 지정하십시오. 매개변수 검사 시 프롬팅중에 나감 프로그램이 수행됩니다. PMTCTL문의 조건들은 제어 매개변수에 지정된 값이 아닌 나감 프로그램에 의해 리턴되는 값과 비교됩니다.

시스템이 나감 프로그램을 찾을 수 없거나 성공적으로 실행할 수 없을 경우에는 시스템이 참(true) 값의 리턴 값을 사용하는 조건을 가정합니다.

나감 프로그램은 다음 세 가지 매개변수를 받아들이도록 작성되어야 합니다.

- 20자 필드. 프롬터는 처음 10자에 명령어를, 마지막 10자에 제어 매개변수명을 전달합니다. 이 필드는 변경할 수 없습니다.
- 제어 매개변수의 값. 이 필드는 명령 처리 프로그램(CPP)에 전달되었을 때와 동일한 형식을 가지며 변경할 수 없습니다.
- 제어 매개변수가 VARY(*YES)로 정의된 경우 값 뒤에 길이 값이 오지 않습니다. 제어 매개변수가 PASSATR(*YES)인 경우에는 속성 바이트가 포함되지 않습니다.
- 32자 필드. 나감 프로그램이 PMTCTL문에서 테스트될 값을 위치시킬 필드입니다.

  PMTCTL문에서 테스트되고 있는 값은 선언된 자료 유형과 동일한 형식으로 리턴되어야 합니다.

다음 예에서 OBJ는 명령어, 프로그램명 또는 파일명이 될 수 있는 규정된 이름입니다. 나감 프로그램은 오브젝트 유형을 판별하여 그 유형을 변수 &RTNVAL로 리턴시킵니다.

```
CMD
PARM OBJ TYPE(Q1) PMTCTLPGM(CNVTYPE)
     Q1: QUAL TYPE(*NAME) LEN(10)
    QUAL TYPE(*NAME) LEN(10) SPCVAL(*LIBL) DFT(*LIBL)
PARM CMDPARM TYPE(*CHAR) LEN(10) PMTCTL(CMD)
PARM PGMPARM TYPE(*CHAR) LEN(10) PMTCTL(PGM)
PARM FILEPARM TYPE(*CHAR) LEN(10) PMTCTL(FILE)
CMD: PMTCTL CTL(OBJ) COND((*EQ *CMD) (*EQ *)) NBRTRUE(*EQ 1)
PGM: PMTCTL CTL(OBJ) COND((*EQ *PGM) (*EQ *)) NBRTRUE(*EQ 1)
FILE: PMTCTL CTL(OBJ) COND((*EQ *FILE) (*EQ *)) NBRTRUE(*EQ 1)
```

다음은 나감 프로그램의 소스문입니다.

```
PGM PARM(&CMD &PARMVAL &RTNVAL)
DCL &CMD *CHAR 20                /* Command and parameter name      */
DCL &PARMVAL *CHAR 20            /* Parameter value                 */
DCL &RTNVAL *CHAR 32             /* Return value                    */
DCL &OBJNAM *CHAR 10             /* Object name                     */
DCL &OBJLIB *CHAR 10             /* Object type                     */
CHGVAR &OBJNAM %SST(&PARMVAL 1 10)
CHGVAR &OBJLIB %SST(&PARMVAL 11 10)
CHGVAR &RTNVAL '*'               /* Initialize return value to error*/
CHKOBJ &OBJLIB/&OBJNAM *CMD      /* See if command exists           */
MONMSG CPF9801 EXEC(GOTO NOTCMD) /* Skip if no command              */
CHGVAR &RTNVAL '*CMD'            /* Indicate object is a command    */
RETURN                           /* Exit                            */
```

```
NOTCMD:
CHKOBJ &OBJLIB/&OBJNAM *PGM      /* See if program exists           */
MONMSG CPF9801 EXEC(GOTO NOTPGM) /* Skip if no program              */
CHGVAR &RTNVAL '*PGM'            /* Indicate object is a program    */
RETURN                           /* Exit                            */
NOTPGM:
CHKOBJ &OBJLIB/&OBJNAM *FILE     /* See if file exists              */
MONMSG CPF9801 EXEC(RETURN)      /* Exit if no file                 */
CHGVAR &RTNVAL '*FILE'           /* Indicate object is a file       */
           ENDPGM
```

추가 매개변수

사용자가 프롬팅 시 기능 키를 눌러 추가 매개변수를 요구하지 않는 한, 자주 사용되지 않는 매개변수에 대해서는 프롬트되지 않도록 지정할 수 있습니다. 이렇게 하려면 해당 매개변수에 대한 PARM문에 PMTCTL(*PMTRQS)을 지정하면 됩니다. 명령이 프롬트될 때 PMTCTL(*PMTRQS)로 코드화된 매개변수들은 지정된 값이 있거나 사용자가 F10 키를 눌러서 이러한 추가 매개변수들을 요구하지 않는 한 프롬트되지 않습니다.

프롬터는 PMTCTL(*PMTRQS)이 지정된 매개변수 앞에 분리선을 표시하여 이 매개변수와 다른 매개변수를 구분시킵니다. 디폴트에는 명령 정의 소스에서 맨 나중 순서로 정의되지 않더라도, PMTCTL(*PMTRQS)이 지정된 모든 매개변수는 맨 끝에 프롬트됩니다. PROMPT 키워드에 상대 프롬트 번호를 지정함으로써 이를 대체할 수 있습니다. 그러나 이렇게 대체할 경우 F10 키를 눌렀을 때 어떤 매개변수가 프롬트에 추가되었는지 아는 것이 어렵습니다.

키 매개변수 및 프롬트 대체 프로그램 사용

프롬트 대체 프로그램은 명령이 프롬트될 때 디폴트 값을 표시하지 않고 현재 값을 허용합니다.

프롬트 대체 프로그램이 명령에 정의되면 다음 두 가지로 프롬트 대체 프로그램을 호출한 결과를 볼 수 있습니다.

- 명령 행에 매개변수 없이 명령어를 입력한 후 F4(프롬트) 키를 누르십시오. 다음 화면은 명령에 대한 키 매개변수를 보여줍니다. 키 매개변수란 오브젝트를 고유하게 식별하는 매개변수(예: 오브젝트명)입니다.

  화면에 모든 필드가 나타나면 Enter 키를 누르십시오. 다음 화면에서는 모든 명령 매개변수를 표시하며, 키 매개변수 필드가 아닌 매개변수 필드에는 디폴트(*SAME이나 *PRV 같은)가 아닌 현재 값이 들어 있습니다.

  예를 들면, 명령 행에 CHGLIB를 입력한 후 F4(프롬트) 키를 누르면 라이브러리 매개변수만 볼 수 있습니다. *CURLIB를 입력한 후 Enter 키를 누르면 현재 라이브러리에 대한 현재 값이 표시됩니다.

- 명령 행에 명령어와 모든 키 매개변수에 대한 값을 입력한 다음, F4(프롬트) 키를 누르십시오. 다음 화면은 모든 명령 매개변수를 표시하며, 키 매개변수 필드가 아닌 매개변수 필드에는 디폴트(*SAME이나 *PRV 같은)가 아닌 현재 값이 들어 있습니다.

  예를 들면, 명령 행에 `CHGLIB LIB(*CURLIB)`를 입력한 후 F4(프롬트) 키를 누르면 현재 라이브러리에 대한 현재 값이 표시됩니다.

F10(추가 매개변수) 키를 누르면, PMTCTL(*PMTRQS)로 정의된 모든 매개변수가 현재 값과 함께 표시됩니다. 추가 매개변수에 대해 자세히 알려면 381 페이지의 『추가 매개변수』를 참조하십시오.

명령 프롬트를 나가려면 F3(나감) 키를 누르십시오.

프롬트 대체 프로그램을 사용하기 위한 절차

프롬트 대체 프로그램을 사용하려면 다음과 같이 하십시오.

1. 명령 정의 소스에서 PARM문의 키 매개변수가 될 매개변수를 지정하십시오. KEYPARM 매개변수에 대해 알려면 다음 섹션 『키 매개변수 식별』 부분을 참조하십시오.
2. 프롬트 대체 프로그램을 작성하십시오. 프롬트 대체 프로그램 작성에 대해 알려면 383 페이지의 『프롬트 대체 프로그램 작성』 부분을 참조하십시오.
3. 명령을 작성하거나 변경할 때 PMTOVRPGM 매개변수에 프롬트 대체 프로그램명을 지정하십시오. 프롬트 대체 프로그램을 사용하는 명령 작성이나 변경에 대해 알려면 386 페이지의 『명령의 작성 및 변경 시 프롬트 대체 프로그램 지정』 부분을 참조하십시오.

키 매개변수 식별

키 매개변수의 수는 변경될 오브젝트를 유일하게 정의하는 데 필요한 매개변수의 수로 제한되어야 합니다.

키 매개변수가 명령 정의 소스에서 올바로 코드화되었는지 확인하려면 다음과 같이 하십시오.

- 명령 정의 소스에서 PARM문에 KEYPARM(*YES)을 지정하십시오.
- KEYPARM(*NO)을 지정하는 모든 매개변수 앞에 KEYPARM(*YES)을 지정하는 매개변수를 모두 정의하십시오.

  주: PARM문에 KEYPARM(*NO)를 지정한 다음에 KEYPARM(*YES)을 지정하면 그 매개변수는 키 매개변수로서 처리되지 않고 경고 메세지가 발행됩니다.
- PARM문에 MAX 값을 둘 이상 지정하지 마십시오.
- 키 매개변수와 관련된 ELEM문에 대해 MAX 값을 둘 이상 지정하지 마십시오.
- PARM문의 PMTCTL 키워드에 *PMTRQS나 프롬트 제어문을 지정하지 마십시오.

- 키 매개변수가 프롬트될 때 나타나기를 원하는 순서대로 명령 정의 소스에 키 매개변수를 위치시키십시오.

프롬트 대체 프로그램 작성

명령이 프롬트될 경우 현재 값을 리턴하기 위한 특정 정보를 프롬트 대체 프로그램에 전달해야 할 필요가 있습니다. 사용자는 프롬트 대체 프로그램을 작성할 때 전달되는 정보와 리턴 값을 모두 고려해야 합니다.

프롬트 대체 프로그램의 CL 소스 예에 대해서는 386 페이지의 『프롬트 대체 프로그램을 사용한 CL의 샘플』을 참조하십시오.

프롬트 대체 프로그램으로 전달된 매개변수: 프롬트 대체 프로그램은 다음 매개변수를 전달받습니다.

- 20자 필드. 필드의 처음 10자는 명령어이고 마지막 10자는 라이브러리명입니다.
- 각 키 매개변수(있는 경우) 값. 두 개 이상의 키 매개변수가 정의된 경우 명령 정의 소스에서 키 매개변수가 정의된 순서대로 매개변수 값이 전달됩니다.
- 프롬트 대체 프로그램에 의해 작성되는 명령 스트링을 보유하는 32676바이트(32K)의 공간. 이 필드의 처음 2바이트에는 리턴되는 16진 길이의 명령 스트링이 들어 있어야 합니다. 처음 2바이트 다음에는 실제 명령 스트링이 옵니다.

  예를 들면, 명령에 두 개의 키 매개변수를 정의할 때 네 개의 매개변수가 다음과 같이 프롬트 대체 프로그램으로 전달됩니다.

  – 명령용으로 하나의 매개변수
  – 키 매개변수용으로 두 개의 매개변수
  – 명령 스트링 공간용으로 하나의 매개변수

프롬트 대체 프로그램으로부터 리턴되는 정보: 전달된 값에 따라 프롬트 대체 프로그램은 키 매개변수가 아닌 매개변수에 대해 현재 값을 검색합니다. 이 값들은 스트링의 길이가 판별되어 리턴되는 명령 스트링 안에 위치합니다.

다음 지침을 사용하여 명령 스트링이 올바로 정의되었는지를 확인하십시오.

- 명령 스트링에 대해 명령 행에서와 같은 키워드 형식을 사용하십시오.
- 명령 스트링 안에 명령어와 키 매개변수를 포함시키지 마십시오.
- 매개변수의 표시 방법과 CPP로 전달될 값을 정의하기 위해 선택 프롬트 문자를 각 키워드 앞에 놓으십시오. 선택 프롬트 문자 사용에 대해 알려면 199 페이지의 『CL 명령의 선택 프롬팅』 부분을 참조하십시오.

  선택 프롬트를 사용할 때는 다음과 같이 하십시오.

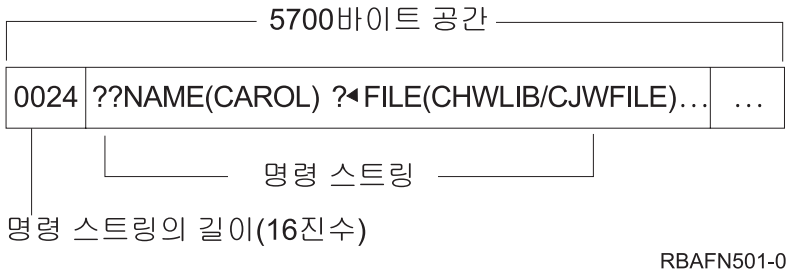
  – 매개변수가 명령 정의 소스에서 MIN(1)으로 정의되면(즉, 매개변수가 필수 매개변수임), 사용자는 프롬트 대체 프로그램으로부터의 명령 스트링에 있는 키워드에 선택 프롬트 문자 **??**를 사용해야 합니다.

– 프롬트 대체 프로그램 명령 스트링에는 선택 프롬트 문자 ?를 사용하지 마십시오.

다음 예는 프롬트 대체 프로그램으로부터 리턴되는 명령 스트링을 표시합니다.

```
??Number(123456) ?<Qualifier(CLIB/CFILE) ?<LIST(ITEM1 ITEM2 ITEM3) ?<TEXT('Carol's file')
```

- 프로그램이 전달하는 공간의 처음 2바이트에 지정된 값이 실제 16진 길이의 명령 스트링인지 확인하십시오.



- 명령이 프롬트될 때 표시되도록 하려는 매개변수만 현재 값의 명령 스트링에 포함시키십시오. 명령 스트링에 포함되지 않는 매개변수는 디폴트로 표시됩니다.
- 명령 스트링에 나타나는 숫자에 대해서는 문자 형식을 사용하십시오. 2진수나 팩 형식을 사용하지 마십시오. 명령 스트링에 16진수를 포함시키지 마십시오.
- 라이브러리와 규정자 또는 규정자와 오브젝트 사이에는 공백을 두지 마십시오. 예를 들어, 다음과 같은 경우가 있습니다.

  ??KWD1(라이브러리/오브젝트**)**
  유효하지 않음

  ??KWD1(라이브러리/오브젝트**)**
  유효하지 않음

  ??KWD1(라이브러리/오브젝트**)**
  유효함

  ??KWD1(라이브러리/오브젝트**)**
  유효함

- 특수 값이나 단일 값을 사용할 경우 그 값들이 명령 정의 소스에 정의된 From 값으로 변환되는지를 확인하십시오.

  예를 들면, 키워드가 명령 정의 소스에서 SPCVAL(*SPECIAL *)로 정의된 특수 값을 가집니다. 이때 *SPECIAL은 From 값이고 *는 To 값입니다. 이 키워드에 대해 현재 값이 검색되면 *는 검색된 값이지만 *SPECIAL은 프롬트 대체 프로그램으로부터 리턴되는 명령 스트링에 표시되어야 합니다. 둘 이상의 특수 값이나 단일 값이 동일한 To 값을 가질 수 있으므로 명령 스트링에는 올바른 From 값이 들어가야 합니다. 예를 들면, `KWD1 SPCVAL((*SPC *) (*SPECIAL *))`이 지정되면 프롬트 대체 프로그램은 *가 *SPC에 대한 To 값인지 또는 *SPECIAL에 대한 To 값인지를 판별해야 합니다.

- 텍스트를 검색하는 데 사용되는 필드의 길이를 다음과 같이 정의하십시오.

  (2*(명령 정의 소스에서 정의된 필드 길이))+ 2

  이 길이는 텍스트 필드에서 허용되는 인용 부호의 최대수입니다. 예를 들면, CHGxxx 명령의 길이는 TEXT 매개변수가 명령 정의 소스에서 LEN(50)으로 정의되면 매개변수는 매개변수의 프롬트 대체 프로그램에서 CHAR(102)로 선언됩니다. 텍스트 검색에 사용되는 필드의 길이를 정의하는 방법의 예는 386 페이지의 『프롬트 대체 프로그램을 사용한 CL의 샘플』 부분을 참조하십시오.

  텍스트 필드에 대한 매개변수가 프롬트 대체 프로그램에서 올바로 정의되지 않고, 프롬트 대체 프로그램에 의해 검색된 텍스트 스트링에 한 개의 인용 부호가 들어 있으면, 명령이 올바로 프롬트되지 않습니다.
- 내장 어포스트로피의 경우 다음 예와 같이 어포스트로피를 두 개 사용해야 합니다.

  ```
  ?<TEXT('Carol''s library')
  ```

어떤 명령들은 특정 모드(예:DEBUG) 또는 특정 작업 상태(예:*BATCH)에서만 수행될 수 있지만 여전히 다른 모드나 작업 상태로부터 프롬트될 수 있습니다. 명령이 프롬트되면 프롬트 대체 프로그램은 사용자의 환경에 상관없이 호출됩니다. 프롬트 대체 프로그램이 명령에 대해 유효하지 않은 모드나 환경에서 호출되면 명령에 대한 디폴트가 표시되고 길이에 0 값이 리턴됩니다. 이런 상태의 예로는 디버그 모드에 있지 않을 때 사용하는 디버그 명령인 CHGDBG(디버그 변경)와 ADDPGM(프로그램 추가)이 있습니다.

프롬트 대체 프로그램에서 오류에 대한 허용: 프롬트 대체 프로그램이 오류를 검출하면 다음을 수행해야 합니다.

- 명령이 프롬트될 때 현재 값이 아닌 디폴트가 표시되도록 명령 스트링 길이를 0으로 설정.
- 호출 스택의 이전 프로그램으로 진단 메세지 송신.
- 이탈 메세지 CPF0011 송신

예를 들면, ″라이브러리가 존재하지 않음″이라는 메세지가 필요하면 다음과 유사한 방법으로 메세지 설명을 추가하십시오.

```
ADDMSGD      MSG('Library &2 does not exist') +
             MSGID(USR0012) +
             MSGF(QGPL/ACTMSG) +
             SEV(40) +
             FMT((*CHAR 4) (*CHAR 10))
```

주: 대체 변수 &1은 메세지 안에 있지 않으나 4자로서 FMT 매개변수에 정의됩니다. &1은 시스템에서 사용되도록 예약되어 있으며 항상 4자여야 합니다. 대체 변수 &1이 메세지에서 정의된 유일한 대체 변수인 경우 메세지를 송신할 때 메세지 자료의

네 번째 바이트에 공백이 들어 있지 않도록 하십시오. 네 번째 바이트는 명령 처리 및 프롬트 작업 동안에 메세지를 관리하기 위해 시스템에서 사용합니다.

이 메세지는 프롬트 대체 프로그램에서 다음 명령을 지정하여 프롬트 대체 프로그램의 호출 프로그램으로 송신될 수 있습니다.

```
SNDPGMMSG    MSGID(USR0012) MSGF(QGPL/ACTMSG) +
             MSGDTA('0000' || &libname) MSGTYPE(*DIAG)
```

프롬트 대체 프로그램은 필요한 모든 진단 메세지를 송신한 후 메세지 CPF0011을 송신해야 합니다. 메세지 CPF0011을 송신하려면 다음과 같이 SNDPGMMSG(프로그램 메세지 송신) 명령을 사용하십시오.

```
SNDPGMMSG    MSGID(CPF0011) MSGF(QCPFMSG) +
             MSGTYPE(*ESCAPE)
```

메세지 CPF0011이 수신되면 메세지 CPD680A가 호출 프로그램으로 송신되고 오류가 발견되었음을 나타내기 위해 프롬트 화면상에 표시됩니다. 모든 진단 메세지는 사용자의 작업 기록부에 위치하게 됩니다.

명령의 작성 및 변경 시 프롬트 대체 프로그램 지정

작성 명령에 프롬트 대체 프로그램을 사용하려는 경우 CRTCMD(명령 작성) 명령을 사용할 때 프로그램명을 지정할 수 있습니다. 또한 CHGCMD(명령 변경) 명령을 사용하여 명령을 변경할 때도 프로그램명을 지정할 수 있습니다. 두 명령의 경우 PMTOVRPGM 매개변수에 프롬트 대체 프로그램명을 지정하십시오.

명령이 작성되거나 변경될 때 키 매개변수가 명령 정의 소스에 정의되었으나 프롬트 대체 프로그램이 지정되지 않았으면 경고 메세지 CPD029B가 발행됩니다. 명령이 프롬트되면 키 매개변수가 무시되고 명령 정의 소스에 지정된 디폴트를 사용해서 명령이 표시됩니다.

때때로 명령이 작성될 때 프롬트 대체 프로그램은 지정되나 명령 정의 소스에 키 매개변수가 정의되지 않을 때가 있습니다. 이 경우 명령이 프롬트되기 전에 프롬트 대체 프로그램이 호출됩니다. 명령이 작성되거나 변경될 때에는 정보 메세지 CPD029A가 송신됩니다.

프롬트 대체 프로그램을 사용한 CL의 샘플

다음 예는 명령과 프롬트 대체 프로그램의 명령 소스를 보여줍니다. 이 명령으로 라이브러리의 소유권과 텍스트 설명을 변경할 수 있습니다. 이 명령에 대한 프롬트 대체 프로그램이 라이브러리명을 수신합니다. 라이브러리 소유자의 현재 값과 텍스트 설명을 검색합니다. 그런 후 이 값들을 명령 스트링에 놓고 값들을 리턴시킵니다.

이 프롬트 대체 프로그램은 "??" 선택 프롬트 문자를 사용합니다.

명령 소스의 샘플

```
CHGLIBATR: CMD   PROMPT('Change Library Attributes')
           PARM  KWD(LIB) +
                   TYPE(*CHAR) MIN(1) MAX(1) LEN(10) +
                   KEYPARM(*YES) +
                   PROMPT('Library to be changed')
           PARM  KWD(OWNER) +
                   TYPE(*CHAR) LEN(10) MIN(0) MAX(1) +
                   KEYPARM(*NO) +
                   PROMPT('Library owner')
           PARM  KWD(TEXT) +
                   TYPE(*CHAR) MIN(0) MAX(1) LEN(50) +
                   KEYPARM(*NO) +
                   PROMPT('Text description')
```

프롬트 대체 프로그램의 샘플

```
   PGM PARM(&cmdname &keyparm1 &rtnstring)
/*****************************************************************/
/*                                                               */
/*  Declarations of parameters passed to the prompt override program */
/*                                                               */
/*****************************************************************/
 DCL  VAR(&cmdname)   TYPE(*CHAR) LEN(20)
 DCL  VAR(&keyparm1)   TYPE(*CHAR) LEN(10)
 DCL  VAR(&rtnstring) TYPE(*CHAR) LEN(5700)


/*****************************************************************/
/*                                                               */
/* Return command string structure declaration                   */
/*                                                               */
/*****************************************************************/

                      /* Length of command string generated     */
 DCL  VAR(&stringlen) TYPE(*DEC)  LEN(5 0) VALUE(131)
 DCL  VAR(&binlen)    TYPE(*CHAR) LEN(2)
                      /* OWNER keyword                          */
 DCL  VAR(&ownerkwd)  TYPE(*CHAR) LEN(8)  VALUE('?<OWNER(')
 DCL  VAR(&name)      TYPE(*CHAR) LEN(10)
                      /* TEXT keyword                           */
 DCL  VAR(&textkwd)   TYPE(*CHAR) LEN(8)  VALUE(' ?<TEXT(')
 DCL  VAR(&descript)  TYPE(*CHAR) LEN(102)


/*****************************************************************/
/*                                                               */
/* Variables related to command string declarations              */
/*                                                               */
/*****************************************************************/
 DCL  VAR(&quote)      TYPE(*CHAR) LEN(1) VALUE('''')
 DCL  VAR(&closparen)  TYPE(*CHAR) LEN(1) VALUE(')')
```

```
/*****************************************************************/
/*                                                               */
/*                  Start of operable code                       */
/*                                                               */
/*****************************************************************/
/*****************************************************************/
/*                                                               */
/* Monitor for exceptions                                        */
/*                                                               */
/*****************************************************************/
    MONMSG  MSGID(CPF0000) +
            EXEC(GOTO CMDLBL(error))


/*****************************************************************/
/*                                                               */
/* Retrieve the owner and text description for the library specified*/
/* on the LIB parameter.  Note:  This program assumes there are     */
/* no apostrophes in the TEXT description, such as (Carol's)  */
/*                                                               */
/*****************************************************************/
    RTVOBJD OBJ(&keyparm1) OBJTYPE(*LIB) OWNER(&name) TEXT(&descript)

    CHGVAR VAR(%BIN(&binlen)) VALUE(&stringlen)


/*****************************************************************/
/*                                                               */
/* Build the command string                                      */
/*                                                               */
/*****************************************************************/
    CHGVAR VAR(&rtnstring) VALUE(&binlen)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &ownerkwd)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &name)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &closparen)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &textkwd)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &quote)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &descript)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &quote)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &closparen)

    GOTO CMDLBL(pgmend)
    ERROR:
    VALUE(0)
    CHGVAR VAR(%BIN(&rtnstring 1 2)) VALUE(&stringlen)
    VALUE(&binlen)
```

```
/*****************************************************************/
/*                                                               */
/* Send error message(s)                                         */
/*                                                               */
/* NOTE: If you wish to send a diagnostic message as well as CPF0011*/
/*       you will have to enter a valid error message ID in the  */
/*       MSGID parameter and a valid message file in the MSGF    */
/*       parameter for the first SNGPGMMSG command listed below. */
/*       If you do not wish to send a diagnostic message, do not */
/*       include the first SNDPGMMSG in your program.  However, in */
/*       error conditions, you must ALWAYS send CPF0011 so the   */
/*       second SNDPGMMSG command must be included in your program. */
/*                                                               */
/*****************************************************************/
    SNDPGMMSG MSGID(XXXXXXX) MSGF(MSGLIB/MSGFILE) MSGTYPE(*DIAG)
    SNDPGMMSG MSGID(CPF0011) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)

PGMEND:
          ENDPGM
```

명령 작성

명령 정의문을 통해 명령을 정의한 후에는 CRTCMD(명령 작성) 명령을 사용하여 명령을 작성할 수 있습니다. CL 또는 고급 언어(HLL)에 대한 명령어, 라이브러리명 및 명령 처리 프로그램명을 정의하거나 또는 소스 멤버, 소스 파일, 명령 환경 및 REXX에 대한 나감 프로그램 외에도, 다음과 같은 명령 속성들을 정의할 수 있습니다.

- 명령에서 사용되는 유효성 검사
- 명령이 수행될 수 있는 모드
  - 실수행 모드
  - 디버그 모드
  - 서비스 모드
- 명령이 사용될 수 있는 곳
  - 일괄처리 작업
  - 대화식 작업
  - 일괄처리 작업 안의 ILE CL 모듈
  - 일괄처리 작업 안의 CL 프로그램
  - 대화식 작업 안의 ILE CL 모듈
  - 대화식 작업 안의 CL 프로그램
  - 일괄처리 작업 안의 REXX 프로시듀어
  - 대화식 작업 안의 REXX 프로시듀어

  - 시스템이 QCMDEXC 또는 QCAPCMD를 호출하여 해석해서 처리하는 명령. (QCMDEXC 및 QCAPCMD API에 대한 정보는 제 6 장 부분을 참조하십시오.)
- 위치별로 지정될 수 있는 매개변수의 최대수
- 프롬트 텍스트가 들어 있는 메세지 파일
- 프롬트할 수 있는 매개변수에 대한 도움말로서 사용되는 도움말 패널 그룹
- 이 명령에 사용되는 일반 도움말 모듈의 도움말 ID명
- DEP문에서 식별되는 메세지가 들어 있는 메세지 파일
- 명령이 처리되는 동안 활동할 현재 라이브러리
- 명령이 처리되는 동안 활동할 제품 라이브러리
- REPLACE(*YES)가 지정될 경우 동일한 이름, 유형 및 라이브러리로 된 기존 명령을 대체하는지의 여부
- 명령과 명령 설명에 대해 공용으로 부여된 권한
- 명령과 그 기능을 간략하게 설명하는 텍스트

REXX CPP가 있는 명령의 경우 다음을 지정할 수도 있습니다.

- 프로시듀어가 시작될 때 명령을 처리하는 초기 명령 환경
- 사용자의 프로시듀어 수행을 제어하기 위한 나감 프로그램

다음은 주문 입력 어플리케이션을 호출하는 ORDENTRY라는 명령을 정의하는 예입니다. CRTCMD 명령은 ORDENTRY에 대한 선행 속성을 정의하고 IBM 제공 소스 파일 QCMDSRC의 멤버 ORDENTRY에 들어 있는 매개변수 정의를 사용하여 명령을 작성합니다. ORDENTRY는 351 페이지의 『매개변수 정의의 예』의 예에서 사용된 PARM문을 포함합니다.

```
CRTCMD     CMD(DSTPRODLB/ORDENTRY) +
           PGM(*LIBL/ORDENT) +
           TEXT('Calls order entry application')
```

다음은 이에 대한 결과로 나온 명령입니다.

```
ORDENTRY  OETYPE(value)
```

여기에서 값은 DAILY, WEEKLY 또는 MONTHLY가 될 수 있습니다.

명령을 작성한 후 다음을 수행할 수 있습니다.

- DSPCMD(명령 표시) 명령을 사용하여 명령 속성을 표시
- CHGCMD(명령 변경) 명령을 사용하여 명령 속성을 변경
- DLTCMD(명령 삭제) 명령을 사용하여 명령을 삭제

명령 정의 소스 리스트

명령을 작성하면 소스 리스트가 작성됩니다. 다음은 샘플 소스 리스트를 보여줍니다. 번호는 리스트 아래에 있는 설명 번호를 가리킵니다.

```
5722SS1 V5R3M0  031231 1                Command Definition        DSTPRODLB/ORDENTRY   SYSNAME  11/20/98 14:53:32 2 Page 1 3

Command name  . . . . . . . . . . . . . . . . . :   ORDENTRY
  Library . . . . . . . . . . . . . . . . . . . :     DSTPRODLB
Command processing program  . . . . . . . . . . :   ORDENT      4
  Library . . . . . . . . . . . . . . . . . . . :     *LIBL
Source file . . . . . . . . . . . . . . . . . . :   QCMDSRC
  Library . . . . . . . . . . . . . . . . . . . :     QGPL
Source file member  . . . . . . . . . . . . . . :   ORDENTRY     11/20/98  14:54:32
Validity checker program  . . . . . . . . . . . :   *NONE
Mode in which valid . . . . . . . . . . . . . . :   *PROD
                                                    *DEBUG
                                                    *SERVICE
Environment allowed . . . . . . . . . . . . . . :   *IREXX
                                                    *BREXX
                                                    *BPGM
                                                    *IPGM
                                                    *EXEC
                                                    *INTERACT
                                                    *BATCH
                                                    *BMOD
                                                    *IMOD
Allow limited user  . . . . . . . . . . . . . . :   *NO
Max positional parameters . . . . . . . . . . . :   *NOMAX
Prompt file . . . . . . . . . . . . . . . . . . :   *NONE
Message file  . . . . . . . . . . . . . . . . . :   QCPFMSG
  Library . . . . . . . . . . . . . . . . . . . :     *LIBL
Authority . . . . . . . . . . . . . . . . . . . :   *LIBCRTAUT
Replace command . . . . . . . . . . . . . . . . :   *YES
Enable graphical user interface . . . . . . . . :   *NO
Threadsafe  . . . . . . . . . . . . . . . . . . :   *NO
Multithreaded job action  . . . . . . . . . . . :   *SYSVAL
Text  . . . . . . . . . . . . . . . . . . . . . :   Calls order entry application
Help book name  . . . . . . . . . . . . . . . . :   *NONE
Help bookshelf. . . . . . . . . . . . . . . . . :   *NONE
Help panel group  . . . . . . . . . . . . . . . :   *NONE
Help identifier . . . . . . . . . . . . . . . . :   *NONE
Help search index . . . . . . . . . . . . . . . :   *NONE
Current library . . . . . . . . . . . . . . . . :   *NOCHG
Product library . . . . . . . . . . . . . . . . :   *NOCHG
Prompt override program . . . . . . . . . . . . :   *NONE
Compiler  . . . . . . . . . . . . . . . . . . . :   IBM AS/400 Command Definition Compiler  5

                                  Command Definition Source
 6
SEQNBR  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+.  DATE  8
   100-              CMD       PROMPT('Order entry command')                                              11/20/98
7
   200-              PARM      KWD(OETYPE) TYPE(*CHAR) RSTD(*YES) +                                       11/20/98
   300                           VALUES(DAILY WEEKLY MONTHLY) MIN(1) +                                   11/20/98
   400                           PROMPT('Type of order entry:')                                          11/20/98
                                      * * * * *  E N D  O F  S O U R C E  * * * * *

5769SS1 V4R4M0  990521               Command Definition              DSTPRODLB/ORDENTRY     11/20/98 14:54:32        Page    2
                                       Cross Reference

Defined Keywords  9
Keyword              Number          Defined      References
OETYPE                 001             200
                    * * * * *  E N D  O F  C R O S S  R E F E R E N C E  * * * * *

5769SS1 V4R4M0  990521               Command Definition              DSTPRODLB/ORDENTRY     11/20/98 14:54:32        Page    3
                                       Final Messages

Message              Sequence
ID                    Number          Sev        Text  10

                                     Message Summary
Total         Info        Error
    0            0           0   11
* CPC0202                            00        Command ORDENTRY created in library DSTPRODLB.  12
                                 * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

제목:

1 OS/400의 프로그램 번호, 버전, 릴리스, 수정 레벨 및 날짜

2 수행 날짜 및 시간

3 리스트의 페이지 번호

프롤로그

4 CRTCMD 명령에 지정된 매개변수 값(지정되지 않은 경우에는 디폴트). 소스가 데이터베이스 파일에 없으면 멤버명, 날짜 및 시간은 생략됩니다.

5 명령 정의 작성 컴파일러의 이름.

소스:

6 소스 파일에서의 행(레코드)의 순번. 순번 뒤의 대시(-)는 소스문이 그 순번에서 시작됨을 나타냅니다. 대시(-)가 없으면 그 명령문이 이전 명령문에서 연속됨을 의미합니다. 예를 들어, PARM 소스문이 순번 200에서 시작하여 순번 300과 400에서 계속됩니다. (순번 200과 300의 PARM문에 계속 표시 문자 +가 있음에 주의하십시오.)

주석 소스문은 다른 소스문과 똑같이 처리되며 순번을 가집니다.

7 소스문

8 소스문이 변경되거나 추가된 최종일. 명령문이 변경되거나 추가되지 않으면 날짜가 표시되지 않습니다. 소스문이 데이터베이스 파일 안에 없으면 날짜가 생략됩니다.

명령 정의문을 처리하는 동안 오류가 발견되어 특정 소스문을 추적한 경우 오류 메세지가 소스문 바로 다음에 인쇄됩니다. 별표(*)는 이 오류 메세지가 들어 있는 행을 표시합니다. 행에는 메세지 ID, 심각도 및 메세지 텍스트가 포함되어 있습니다.

명령 정의 오류에 대해 자세히 알려면 393 페이지의 『명령 정의문 처리 시 발생하는 오류』 부분을 참조하십시오.

상호 참조:

9 키워드 표는 명령 정의에 유효하게 정의된 키워드들의 상호 참조 리스트입니다. 표에는 키워드, 명령 안의 키워드 위치, 키워드가 정의된 명령문의 순번 및 키워드를 참조하는 명령문의 순번이 들어 있습니다.

명령 정의에 유효한 레이블이 정의되어 있으면, 레이블(레이블 표)의 상호 참조 리스트가 제공됩니다. 이 표에는 레이블, 레이블이 정의되는 명령문의 순번, 레이블을 참조하는 명령문의 순번 등이 있습니다.

메세지:

10 명령 정의문을 처리하는 동안 발생하는 일반적인 오류 메세지(소스 섹션에서 나열되지 않는)의 리스트(있는 경우). 이 섹션에는 각 메세지에 대한 메세지 ID, 오류가 발생한 소스문의 순번, 심각도 및 메세지가 들어 있습니다.

메세지 요약:

11 명령 정의문을 처리하는 동안 발행된 메세지 갯수에 대한 요약. 총계가 심각도별 합계와 함께 표시됩니다.

12 메세지 요약 다음에 완료 메세지가 인쇄됩니다.

명령 정의문 처리 시 발생하는 오류

명령 정의문 처리 시 발견되는 오류의 유형에는 구문 오류, 정의되지 않은 키워드와 레이블에 대한 참조 및 누락 명령문이 있습니다. 명령 정의 컴파일러가 다음 유형의 오류를 검출하면 명령 작성이 중단됩니다(심각도 코드는 무시됨).

- 값 오류
- 구문 오류

명령 작성을 중단시킨 오류를 발견한 후에도, 명령 정의 컴파일러는 계속해서 다른 오류에 대해 소스문을 검사합니다. 구문 오류와 고정값 오류가 발생하면 사용자명에서의 오류 및 키워드나 레이블에 대한 값이나 참조에서의 오류를 식별하는 최종 검사가 수행되지 않습니다. 구문 오류와 고정값 오류에 대한 검사가 계속됩니다. 따라서 명령을 다시 작성하기 전에 가능한 한 많은 오류를 발견하여 정정하는 것이 좋습니다. 소스문에 있는 오류를 수정하기 위해 EDTF(파일 편집) 명령이나 소스 입력 유틸리티(SEU)를 사용할 수 있습니다. SEU에 대한 자세한 정보는 AS/400용 ADTS: 소스 입력 유틸리티  책을 참조하십시오.

명령 정의 소스 리스트에서 특정 소스문과 직접 관련된 오류 상태는 그 명령 다음에 나열됩니다. 이러한 인라인 메세지의 예를 보려면 391 페이지의 『명령 정의 소스 리스트』를 참조하십시오. 특정 소스문과 관계가 없는 보다 일반적인 메세지들은 소스문에 인라인으로 표시되지 않고, 리스트의 메세지 섹션에 나열됩니다.

명령 정의 표시

DSPCMD(명령 표시) 명령을 사용하면 CRTCMD 명령에 매개변수로 지정된 값을 표시하거나 인쇄할 수 있습니다. DSPCMD 명령은 사용자 명령이나 IBM 제공 명령에 대해 다음 정보를 표시합니다.

- 규정된 명령어. 라이브러리명은 표시될 명령이 들어 있는 라이브러리명입니다.
- 명령 처리 프로그램의 규정된 이름. 라이브러리명은 라이브러리명이 CRTCMD나 CHGCMD 명령에 지정되어 명령이 작성되었을 때 명령 처리 프로그램이 상주하는 라이브러리명입니다. 라이브러리명이 지정되지 않았으면, *LIBL이 라이브러리 규정자로 표시됩니다. CPP가 REXX 프로시듀어이면 *REXX가 표시됩니다.
- 소스 파일이 데이터베이스 파일인 경우의 규정된 소스 파일명. 라이브러리명은 CRTCMD 명령이 처리되었을 때 소스 파일이 위치한 라이브러리명입니다. 소스 파일이 데이터베이스 파일이 아니면 이 필드는 공백입니다.
- 소스 파일이 데이터베이스 소스 파일인 경우의 소스 파일 멤버명
- CPP가 REXX 프로시듀어이면 다음 정보가 표시됩니다.
  - REXX 프로시듀어 멤버명

  - REXX 프로시듀어가 있는 규정된 REXX 소스 파일명
  - REXX 명령 환경
  - REXX 나감 프로그램
- 유효성 검사 프로그램의 규정된 이름. 라이브러리명은 라이브러리명이 CRTCMD나 CHGCMD 명령에 지정되어 명령이 작성되었을 때 프로그램이 상주하는 라이브러리 명입니다. 라이브러리명이 지정되지 않았으면, *LIBL이 라이브러리 규정자로 표시됩니다.
- 유효한 조작 모드.
- 명령이 수행될 수 있는 유효한 환경
- 명령의 위치 한계. 명령에 대한 위치 한계가 존재하지 않으면 *NOMAX가 표시됩니다.
- 프롬트 메세지 파일의 규정된 이름. 라이브러리명은 CRTCMD 명령이 수행되었을 때 메세지 파일이 위치했던 라이브러리명입니다. 명령에 대한 프롬트 메세지 파일이 존재하지 않으면 *NONE이 표시됩니다.
- DEP문에 대한 메세지 파일의 규정된 이름. 명령이 작성되었을 때 메세지 파일에 대한 라이브러리명이 지정되었으면 그 라이브러리명이 표시됩니다. 명령이 작성되었을 때 라이브러리 리스트가 사용되었으면 *LIBL이 표시됩니다. 명령에 대한 DEP 메세지 파일이 존재하지 않으면 *NONE이 표시됩니다.
- 도움말 패널 그룹의 규정된 이름.
- 명령에 대한 도움말 ID명
- 프롬트 대체 프로그램의 규정된 이름
- 명령과 관련된 텍스트. 명령에 대한 텍스트가 존재하지 않으면 공백이 표시됩니다.
- 명령 프롬트가 그래픽 사용자 인터페이스로 변환될 수 있는지를 나타내는 인디케이터.
- 스레드세이프 인디케이터.
- 명령이 스레드세이프가 아닌 경우 복수 스레드 작업 활동.

QCDRCMDI(명령 정보 검색) API를 사용하면 명령 작성 시 CRTCMD(명령 작성) 명령에 지정되었던 명령 속성을 리턴할 수 있습니다. QCDRCMDD(명령 정의 검색) API를 사용하여 매개변수 정보, 내부 매개변수 종속성 정보 및 조건부 프롬팅 정보를 포함한 명령 정의 오브젝트의 구조를 검색할 수 있습니다. 이러한 API에 대한 자세한 정보는 iSeries Information Center **프로그래밍** 범주의 *API* 섹션을 참조하십시오.

프로시듀어나 프로그램 안의 명령에 대한 명령 정의 변경의 효과

CL 모듈이나 프로그램이 작성될 때 이 프로시듀어나 프로그램에 있는 명령들에 대한 명령 정의는 모듈이나 프로그램을 생성하는 데 사용됩니다. CL 프로시듀어나 프로그램이 실행될 때 명령 정의도 사용됩니다. CL 프로시듀어나 프로그램 안의 명령에 대해 라이브러리명을 지정하면 명령은 프로그램 작성 시 또는 프로그램 수행 시에 동일한 라이브러리에 있어야 합니다. CL 프로시듀어나 프로그램 안의 명령에 대해 *LIBL을 지정하면 프로시듀어의 작성 및 실행 시에 라이브러리 리스트(*LIBL)를 사용하여 명령을 찾습니다.

명령을 사용하는 모듈과 프로그램을 재작성하지 않고도 명령에 대한 명령 정의문을 다음과 같이 변경할 수 있습니다. 명령 정의문 소스에 대한 변경 중 일부는 명령을 재작성해야 합니다. 기타 변경은 CHGCMD(명령 변경) 명령을 통해 이루어집니다.

- 임의의 위치에 선택적 매개변수 추가. 위치 한계 앞에 선택적 매개변수를 추가하면 위치 형식으로 지정된 매개변수를 가진 프로시듀어, 프로그램 및 일괄처리 입력 스트림에 영향을 미칠 수 있습니다.
- REL 및 RANGE 검사가 보다 덜 제한적인 값을 갖도록 변경.
- 새로운 특수 값 추가. 그러나 이 값이 변경되기 전에 값이 지정될 수 있다면 프로시듀어나 프로그램의 조치를 변경할 수도 있습니다.
- 매개변수의 순서를 변경. 그러나 위치 한계 앞에 있는 매개변수 순서를 변경하면 위치 형식으로 지정된 매개변수를 가진 프로시듀어, 프로그램 및 일괄처리 입력 스트림에 영향을 *미칠 것입니다*.
- 단순 리스트 안의 선택적 요소의 수를 늘림
- 디폴트 값 변경. 그러나 이것은 프로시듀어나 프로그램의 조작에 영향을 미칠 수도 있습니다.
- 단순 리스트 안의 필수 리스트 항목의 수를 줄임
- 필수 매개변수를 선택적 매개변수로 변경
- RSTD를 *YES에서 *NO로 변경
- FULL(*NO)이 지정되면 길이를 늘림
- FULL을 *YES에서 *NO로 변경
- PROMPT 텍스트 변경
- ALLOW 값을 보다 덜 제한적인 값이 되도록 변경
- 새로운 명령 처리 프로그램에서 올바른 유형의 매개변수를 적절한 수만큼 허용하는 경우 명령 처리 프로그램명 변경
- 새로운 유효성 검사 프로그램에서 올바른 유형의 매개변수를 적절한 수만큼 허용하는 경우 유효성 검사 프로그램의 이름 변경

- CL 프로시듀어나 프로그램에서 사용되는 동일한 명령의 기존 모드에 영향을 미치지 않는 한 그 명령이 수행 가능한 새로운 모드로 변경
- TYPE을 호환성이 있고 덜 제한적인 값으로 변경. 예를 들면, TYPE을 *NAME에서 *CHAR로 변경합니다.
- MAX 값을 1보다 큰 값으로 변경
- PASSATR 및 VARY 값 변경

명령이 사용되는 CL 프로시듀어나 프로그램에서 무엇이 지정되었는지에 따라 명령 정의문에 대해 다음을 변경할 수 있습니다.

- 매개변수 제거
- RANGE 및 REL 값을 보다 더 제한적인 값으로 변경
- 특수 값 제거
- 리스트에 허용되는 요소의 수를 줄임
- TYPE 값을 보다 제한적이고 원래의 TYPE 값과 호환성이 없도록 변경. 예를 들어, TYPE 값을 *CHAR에서 *NAME으로 변경하거나 *PNAME에서 *CHAR로 변경
- 이전에 리스트 항목이었던 SNGVAL 매개변수를 추가
- 선택적 매개변수명 변경
- 값의 리스트에서 값을 제거
- 필수 리스트 항목의 수를 늘림
- SNGVAL 매개변수를 SPCVAL 매개변수로 변경
- 단순 리스트를 유사한 요소를 가진 혼합 리스트로 변경
- 선택적 매개변수를 상수로 변경
- RTNVAL을 *YES에서 *NO로 또는 *NO에서 *YES로 변경
- 대소문자 값을 *MIXED에서 *MONO로 변경

명령 정의문에 대해 다음을 변경할 수는 있지만 그로 인해 명령을 사용하는 프로시듀어나 프로그램의 기능이 달라질 수 있습니다.

- 값의 내용 변경
- 디폴트 값 변경
- SNGVAL 매개변수를 SPCVAL 매개변수로 변경
- 값을 SNGVAL 매개변수로 변경
- 리스트를 리스트 안의 리스트로 변경
- 대소문자 값을 *MIXED에서 *MONO로 변경

명령 정의문에 대한 다음과 같은 변경은 그 명령을 사용하는 프로시듀어나 프로그램을 재작성해야 합니다.

- 새로운 필수 매개변수의 추가

- 필수 매개변수의 제거
- 필수 매개변수명의 변경
- 필수 매개변수를 상수로 변경
- 명령 처리 프로그램을 *REXX로 변경하거나 *REXX로부터 변경

그 외에도 명령이 작성되거나 변경될 때 명령 처리 프로그램명 또는 유효성 검사 프로그램명에 *LIBL을 규정자로 지정하면 명령 정의문을 변경하지 않고도 명령 처리 프로그램이나 유효성 검사 프로그램을 라이브러리 리스트 안의 다른 라이브러리로 이동시킬 수 있습니다.

명령 디폴트 변경

CHGCMDDFT(명령 디폴트 변경) 명령으로 명령 키워드의 디폴트 값을 변경할 수 있습니다. 자세한 정보는 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오. 새로운 디폴트 값으로 변경하려면 키워드에 기존의 디폴트가 있어야 합니다. IBM 제공 명령이나 사용자 작성 명령을 변경할 수 있습니다. IBM 제공 명령의 디폴트를 변경할 경우에는 주의해야 합니다. 다음은 디폴트를 변경할 때 주의해야 할 사항입니다.

1. 사용자 라이브러리에서 변경하려는 IBM 제공 명령의 사본을 작성하려면 CRTDUPOBJ(오브젝트 사본 작성) 명령을 사용하십시오. 그러면 시스템상의 다른 사용자들이 IBM 제공 디폴트를 필요한 경우에 사용할 수 있습니다.

   라이브러리 리스트에서 앞부분에 QSYS가 오는 사용자 라이브러리나 다른 시스템 제공 라이브러리를 이동시키려면 CHGSYSLIBL(시스템 라이브러리 리스트 변경) 명령을 사용하십시오. 이 명령을 사용하면 라이브러리 규정자를 사용하지 않고도 변경된 명령을 사용할 수 있습니다.

   시스템 전반에 걸쳐 필요한 명령의 변경은 사용자 라이브러리에서 이루어져야 합니다. 또한 QSYS 앞의 QSYSLIBL 시스템 값에 사용자 라이브러리명을 추가해야 합니다. 변경된 명령은 시스템 전반에 걸쳐 사용됩니다. IBM 제공 디폴트를 사용하는 어플리케이션을 실행해야 할 경우 CHGSYSLIBL(시스템 라이브러리 리스트 변경) 명령을 사용하여 작업하십시오. 이와 같이 하면 영향을 받는 명령에 대한 특수 라이브러리나 라이브러리 규정이 제거됩니다.
2. 사용권 프로그램의 신규 릴리스를 설치하면 기계에 있는 사용권 프로그램을 위한 모든 IBM 제공 명령이 대체됩니다. 새로운 릴리스를 설치할 때는 CL 명령을 사용하여 명령에 대한 변경을 작성해야 합니다. 이와 같이 하면 CL 프로그램을 실행하여 새로운 키워드를 선별하는 신규 명령을 복제할 수 있으며 명령 디폴트를 변경시킬 수 있습니다.

   IBM 제공 명령에 새로운 키워드가 들어 있으면, 이전 릴리스의 명령 사본이 제대로 수행되지 않을 수도 있습니다.

다음은 이전 버전을 삭제하고 새로 변경된 명령을 작성하는 데 사용되는 CL 프로그램의 예입니다.

```
         PGM
DLTCMD USRQSYS/SIGNOFF
CRTDUPOBJ OBJ(SIGNOFF) FROMLIB(QSYS) OBJTYPE(*CMD) +
          TOLIB(USRQSYS)  NEWOBJ(*SAME)
CHGCMDDFT CMD(USRQSYS/SIGNOFF) NEWDFT('LOG(*LIST)')
 .
 .
Repeat the DLTCMD, CRTDUPOBJ and CHGCMDDFT for each
command you want changed
 .
 .
           ENDPGM
```

새로운 릴리스를 설치할 때 사용하기 위해 CL 명령 디폴트에 작성하는 변경사항을 추적할 수 있습니다. 변경사항을 추적하려면 나감점에 대한 나감 프로그램 QIBM_QCA_RTV_COMMAND를 등록하십시오. 나감 프로그램은 CHGCMDDFT 명령을 실행할 때 호출됩니다. 나감 프로그램으로 전달된 매개변수 중 하나는 실행되고 있는 명령 스트링입니다. 이 명령 스트링을 소스 파일에 저장한 후 이 소스 파일을 CL 프로그램으로 컴파일할 수 있습니다. 마지막으로, 이 프로그램을 사용하여 이전 릴리스 중에 명령 디폴트에 작성했던 변경사항을 재생성합니다. 자세한 정보는 iSeries Information Center의 **프로그래밍** 범주에 나오는 *CL* 섹션에서 명령 분석기 검색 나감 프로그램 설명을 참조하십시오.

다음은 CHGCMDDFT 명령에 대한 NEWDFT 명령 스트링을 만드는 단계입니다. 이 예에서는 USRQSYS/CRTCLPGM 명령이 사용됩니다.

1. 다음의 명령을 사용하여 사용자 라이브러리에서 변경될 명령의 사본을 작성하십시오.

   ```
   CRTDUPOBJ OBJ(CRTCLPGM) FROMLIB(QSYS) OBJTYPE(*CMD) +
             TOLIB(USRQSYS)  NEWOBJ(*SAME)
   ```

2. 소스 입력 유틸리티(SEU)에 의해 참조되는 소스 파일에서 변경될 명령어를 입력하십시오.
3. F4 키를 눌러 명령 프롬터를 호출하십시오.
4. 변경시킬 키워드의 신규 디폴트 값을 입력하십시오. 이 예에서는 `AUT(*EXCLUDE)`와 `TEXT('Isn''t this nice text')`가 입력됩니다.
5. 필수 키워드는 디폴트 값을 가질 수 없습니다. 그러나 소스 파일에 명령 스트링을 넣으려면 각 필수 키워드에 유효한 값을 지정해야 합니다. PGM 매개변수에 PGM1을 지정하십시오.
6. Enter 키를 눌러 명령 스트링을 소스 파일에 위치시키십시오. 리턴된 명령 스트링은 다음과 같습니다.

   ```
   USRQSYS/CRTCLPGM PGM(PGM1) AUT(*EXCLUDE) +
   TEXT('Isn''t this nice text')
   ```

7. 명령 스트링으로부터 필수 키워드를 제거하십시오.

```
USRQSYS/CRTCLPGM  AUT(*EXCLUDE) +
TEXT('Isn''t this nice text')
```

   기존의 디폴트 값을 가진 매개변수, 요소 또는 규정자만 변경될 수 있다는 것을 기억하십시오. 기존의 디폴트 값이 없는 매개변수, 요소 또는 규정자에 값이 지정되면 디폴트가 변경되지 않습니다.

8. 다음 예와 같이 시작 부분에 CHGCMDDFT를 삽입하십시오.

```
CHGCMDDFT USRQSYS/CRTCLPGM AUT(*EXCLUDE) +
TEXT('Isn''t this nice text')
```

9. NEWDFT 키워드에 대한 입력은 다음 예와 같이 인용 부호로 묶어야 합니다.

```
CHGCMDDFT USRQSYS/CRTCLPGM 'AUT(*EXCLUDE) +
TEXT('Isn''t this nice text')'
```

10. NEWDFT 값에 내장 어포스트로피가 있으므로 올바로 실행되도록 하기 위해서는 어포스트로피를 이중(″)으로 사용해야 합니다.

```
CHGCMDDFT USRQSYS/CRTCLPGM 'AUT(*EXCLUDE) +
TEXT(''Isn''''t this nice text'')'
```

11. F4 키를 눌러 명령 프롬터를 호출한 후 F11 키를 눌러 키워드 프롬팅을 요구하면 다음 화면이 표시됩니다.

```
Command . . . . . :   CMD         R   CRTCLPGM
  Library . . . . . :                   USRQSYS
New default parameter string:   NEWDFT      R   'AUT(*EXCLUDE)
TEXT(''Isn''''t this nice text'')'
```

12. 이제 Enter 키를 누르면, CHGCMDDFT 명령 스트링이 다음과 같이 표시됩니다.

```
CHGCMDDFT CMD(USRQSYS/CRTCLPGM) NEWDFT('AUT(*EXCLUDE) +
          TEXT(''Isn''''t  this nice text'')')
```

13. F1 키를 눌러 SEU에서 나가 CL 프로그램이나 프로시듀어를 작성하고 실행시키십시오.

14. USRQSYS/CRTCLPGM은 디폴트 값으로 AUT에는 *EXCLUDE, TEXT에는 'Isn't this nice text'를 가집니다.

예 1

CRTPF 명령의 MAXMBRS 키워드에 *NOMAX를 디폴트 값으로 제공하려면 다음과 같이 하십시오.

```
     CRTPF FILE(FILE1) RCDLEN(96) MAXMBRS(1)
    .
    .
CHGCMDDFT CMD(CRTPF) NEWDFT('MAXMBRS(*NOMAX)')
```

예 2

CRTPF 명령의 MAXMBRS 키워드에 10을 디폴트 값으로 제공하려면 다음과 같이 하십시오.

```
   CRTPF FILE(FILE1) RCDLEN(96) MAXMBRS(*NOMAX)
  .
  .
CHGCMDDFT CMD(CRTPF) NEWDFT('MAXMBRS(10)')
```

예 3

다음은 CRTCLPGM 명령의 SRCFILE 키워드에 대한 첫 번째 규정자의 디폴트 값으로 LIB001을, 두 번째 규정자의 디폴트 값으로 FILE001을 제공합니다. AUT 키워드는 현재 *EXCLUDE를 디폴트 값으로 가지고 있습니다.

```
   CRTCLPGM PGM(PROGRAM1) SRCFILE(*LIBL/QCMDSRC)
  .
  .
CHGCMDDFT CMD(CRTCLPGM) +
          NEWDFT('SRCFILE(LIB001/FILE001) AUT(*EXCLUDE)')
```

예 4

다음은 CHGJOB 명령의 PRTTXT 키워드에 디폴트 값 'Isn't this print text'를 제공합니다. NEWDFT 키워드에는 내장 어포스트로피가 있으므로 어포스트로피를 이중으로 사용해서는 안됩니다. 이중으로 사용할 경우 올바로 처리되지 않습니다.

```
 CHGJOB PRTTXT('Isn''t this print text')
  .
  .
CHGCMDDFT CMD(CHGJOB) +
          NEWDFT('PRTTXT(''Isn''''t this print text'')')
```

예 5

다음은 CRTLF 명령에 대한 DTAMBRS 키워드의 첫 번째 리스트 항목의 첫 번째 규정자(라이브러리명)에 QGPL을 디폴트 값으로 제공합니다. DTAMBRS 키워드의 두 번째 리스트 항목(멤버명)의 새로운 디폴트 값은 MBR1입니다.

```
   CRTLF FILE(FILE1) DTAMBRS(*ALL)
  .
  .
CHGCMDDFT CMD(CRTLF) +
          NEWDFT('DTAMBRS((QGPL/*N (MBR1)))')
```

전체 DTAMBRS 리스트에 대한 SNGVAL(단일 값)이 *ALL이므로 라이브러리명에 대한 디폴트인 *CURRENT와 멤버명에 대한 디폴트인 *NONE은 원래의 명령 프롬트 화면에 표시되지 않습니다. 디폴트 *CURRENT와 *NONE은 새로운 디폴트 값으로 변경될 수 있으나 전체 DTAMBRS 리스트에 대한 *ALL 단일 값 때문에 원래의 프롬트 화면상에 표시되지 않습니다.

예 6

작업의 스풀 파일을 표시할 명령을 작성하십시오.

```
   CRTDUPOBJ OBJ(WRKJOB) FROMLIB(QSYS) +
          TOLIB(MYLIB) NEWOBJ(WRKJOBSPLF)
WRKJOBSPLF  OPTION(*SPLF)
```

```
.
.
CHGCMDDFT CMD(MYLIB/WRKJOBSPLF) +
          NEWDFT('OPTION(*SPLF)')
```

명령 처리 프로그램 또는 프로시듀어 작성

명령 처리 프로그램(CPP)으로는 CL이나 HLL 프로그램 또는 REXX 프로시듀어가 될 수 있습니다. CL이나 HLL로 작성된 프로그램은 CALL CL 명령을 사용하여 직접 호출할 수 있습니다. STRREXPRC(REXX 프로시듀어 시작) 명령을 사용하여 REXX 프로시듀어를 직접 호출할 수 있습니다. CRTCMD(명령 작성) 명령이 수행될 때 명령 처리 프로그램이 반드시 존재할 필요는 없습니다. 라이브러리 규정자로서 *LIBL이 사용되면 작성된 명령이 수행될 때 라이브러리 리스트를 사용하여 명령 처리 프로그램을 찾습니다.

명령 처리 프로그램의 수행 결과로 발행되는 메세지는 작업 메세지 대기행렬로 송신되어 자동으로 표시 또는 인쇄될 수 있습니다. 사용자는 요구한 표시장치로 화면을 송신할 수 있습니다.

주:

1. 명령에 정의된 매개변수는 정의된 순서(PARM문의 순서)에 따라 개별적으로 전달됩니다.
2. 10진값은 PARM문에서 지정된 길이의 팩 10진값으로서 HLL과 CL 프로그램에 전달됩니다.
3. 문자, 이름 및 논리값은 PARM문에서 정의된 길이의 문자 스트링으로서 HLL과 CL 프로그램에 전달됩니다.

CL 또는 HLL 명령 처리 프로그램 작성

402 페이지의 그림 15는 CRTCMD(명령 작성) 명령, 명령 정의문 및 명령 처리 프로그램 간의 관계를 보여줍니다.

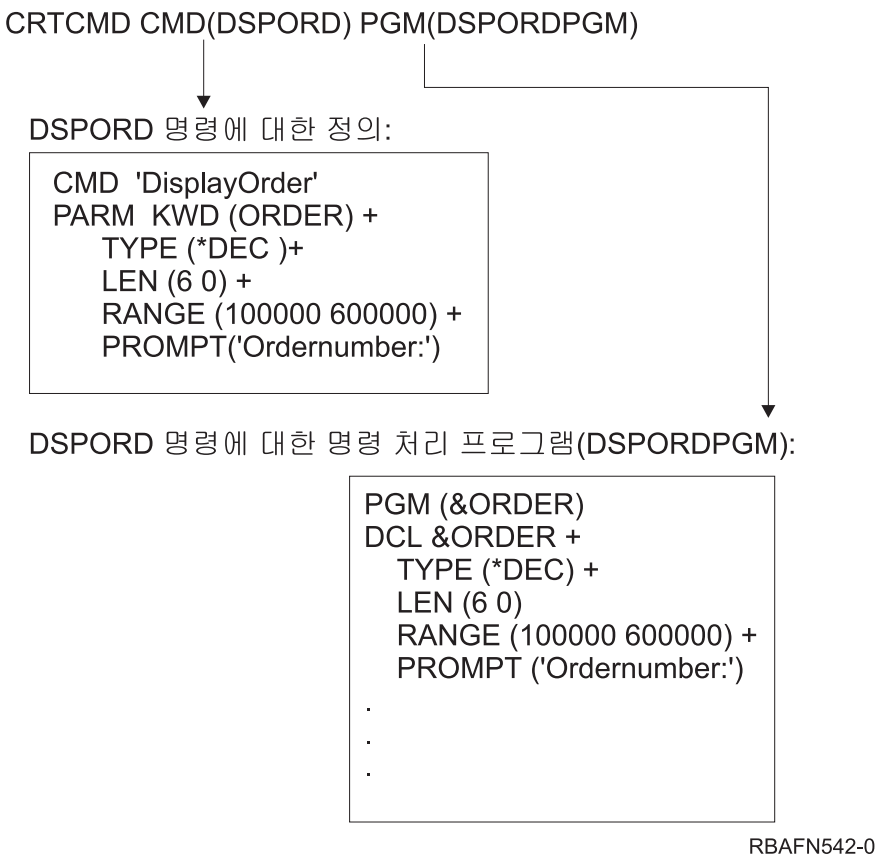


그림 15. CL과 HLL에 대한 명령 관계

명령 처리 프로그램이 CL로 작성된 프로그램인 경우 매개변수 값을 수신하는 변수는 각 PARM문에 지정된 유형과 길이에 일치하도록 선언되어야 합니다. 다음은 이러한 대응 관계를 나타내는 표입니다. (그림 15의 매개변수 ORDER의 선언에 유의하십시오.)

PARM문 유형	**PARM문 길이**	**선언된 변수 유형**	**선언된 가변 길이**
*DEC	x y[1]	*DEC	x y[1]
*LGL	1	*LGL	1
*CHAR	n	*CHAR	≤n[2]
*NAME	n	*CHAR	≤n[2]
*CNAME	n	*CHAR	≤n[2]
*SNAME	n	*CHAR	≤n[2]
*GENERIC	n	*CHAR	≤n[2]
*CMDSTR	n	*CHAR	≤n[2]
*DATE	7	*CHAR	7
*TIME	6	*CHAR	6
*INT2	n	*INT 또는 *CHAR	2
*INT4	n	*INT 또는 *CHAR	4
*UINT2	n	*UINT 또는 *CHAR	2
*UINT4	n	*UINT 또는 *CHAR	4

PARM문 유형	PARM문 길이	선언된 변수 유형	선언된 가변 길이
:			

[1] x는 길이이며 y는 소수 자릿수의 갯수입니다.

[2] 문자 변수인 경우 전달된 값의 길이가 선언된 길이보다 크면 값은 선언된 길이에 맞게 절단됩니다. RTNVAL(*YES)이 지정되면 선언된 길이는 PARM문에 정의된 길이와 같아야 합니다.

명령 처리 프로그램으로서 사용된 CL로 작성된 프로그램은 2진값(*INT2 또는 *INT4와 같은)을 처리할 수 있습니다. 프로그램은 이러한 값을 문자 필드 형태로 수신할 수 있습니다. 이 경우 2진 내장 기능(%BINARY)을 사용하여 해당 값을 10진 값으로 변환할 수 있습니다. 그렇지 않은 경우 CL 프로그램은 해당 값을 정수 변수로 선언할 수 있습니다.

*INT2 또는 *INT4와 *UINT2 또는 *UINT4 간의 차이점은 *INT 2와 *INT4 유형은 부호화 정수이고 *UINT2와 *UINT4 유형은 부호 없는 정수라는 것입니다. 모든 정수 유형의 디폴트 값은 0입니다. *UINT2 및 *UINT4 유형에는 *INT 및 *INT4 유형과 같은 제한사항이 있습니다.

주: %BINARY 내장 기능은 부호화 정수와 함께 사용하기 위한 것입니다. 비부호화 정수의 경우 해당 함수가 없습니다.

명령 처리 프로그램의 예에 대해서는 405 페이지의 『명령 정의 및 작성의 예』를 참조하십시오.

REXX 명령 처리 프로시듀어 작성

그림 16은 CRTCMD(명령 작성) 명령, 명령 정의문 및 REXX의 명령 처리 프로시듀어 간의 관계를 보여줍니다.

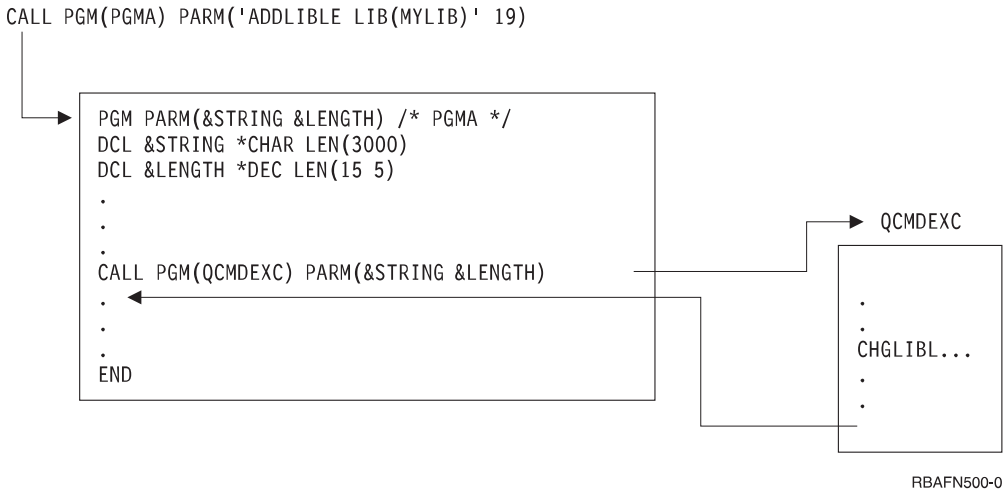


그림 16. REXX에 대한 명령 관계

유효성 검사 프로그램 작성

사용자의 명령에 대해 유효성 검사 프로그램을 작성하는 경우 CRTCMD 명령의 VLDCKR 매개변수에 유효성 검사 프로그램의 이름을 지정하십시오. CRTCMD 명령이 수행될 때 그 프로그램이 반드시 존재하지 않아도 됩니다. 라이브러리 규정자로서 *LIBL이 사용되면 작성된 명령이 수행될 때 라이브러리 리스트를 사용하여 유효성 검사 프로그램을 찾습니다.

다음은 유효성 검사 프로그램을 위한 두 가지 고려사항입니다.

- 유효성 검사 프로그램은 명령 구문이 올바를 때에만 호출됩니다. 모든 매개변수는 그들이 명령 처리 프로그램으로 전달될 때와 같이 유효성 검사 프로그램으로 전달됩니다.
- 변경된 값들은 항상 명령 처리 프로그램으로 전달되지 않으므로 매개변수 값을 변경하기 위해 유효성 검사 프로그램을 사용해서는 안됩니다.

이 섹션의 나머지 부분에서는 CL로 작성된 유효성 검사 프로그램으로부터 메세지를 송신하는 방법에 관해 설명합니다.

유효성 검사 프로그램이 오류를 발견하면 프로그램은 진단 메세지를 이전 호출 프로그램으로 송신한 뒤 이탈 메세지 CPF0002를 송신합니다. 예를 들면, 계정 번호가 더 이상 유효하지 않음을 알리는 메세지가 필요한 경우 다음과 유사한 메세지 설명을 메세지 파일에 추가하십시오.

```
ADDMSGD      MSG('Account number &2 no longer valid') +
             MSGID(USR0012) +
             MSGF(QGPL/ACTMSG) +
             SEV(40) +
             FMT((*CHAR 4) (*CHAR 6))
```

대체 변수 &1은 메세지 안에 있지 않으나 FMT 매개변수 안에서 4자로 정의되어 있음을 주의하십시오. &1은 시스템에서 사용되도록 예약되어 있으며 항상 4자여야 합니다. 대체 변수 &1이 메세지에서 정의된 유일한 대체 변수인 경우 메세지를 송신할 때 메세지 자료의 네 번째 바이트에 공백이 들어 있지 않도록 하십시오.

유효성 검사에 다음을 지정하면 시스템이 이 메세지를 송신할 수 있습니다.

```
SNDPGMMSG    MSGID(USR0012) MSGF(QGPL/ACTMSG) +
             MSGDTA('0000' || &ACCOUNT) MSGTYPE(*DIAG)
```

유효성 검사는 필요한 모든 진단 메세지를 송신한 후 메세지 CPF0002를 송신해야 합니다. 메세지 CPF0002를 송신하는 SNDPGMMSG(프로그램 메세지 송신) 명령은 다음과 같습니다.

```
SNDPGMMSG    MSGID(CPF0002) MSGF(QCPFMSG) +
             MSGTYPE(*ESCAPE)
```

시스템이 메세지 CPF0002를 수신한 후 호출 프로그램에 메세지 CPF0001을 송신해서 오류가 발견되었음을 알립니다.

메세지 CPD0006은 사용자 정의 유효성 검사 프로그램에서 사용하기 위해 정의된 메세지입니다. 즉시 메세지가 메세지 자료로 송신될 수 있습니다. 다음 예에서 메세지 앞에 4개의 0이 있어야 함에 주의하십시오.

다음은 유효성 검사의 예입니다.

```
PGM PARM(&PARM01)
DCL VAR(&PARM01) TYPE(*CHAR) LEN(10)
IF COND(&PARM01 *EQ 'ERROR') THEN(DO)
SNDPGMMSG MSGID(CPD0006) MSGF(QCPFMSG) +
          MSGDTA('0000 DIAGNOSTIC MESSAGE FROM USER-DEFINED +
          VALIDITY CHECKER INDICATING THAT PARM01 IS IN ERROR.') +
          MSGTYPE(*DIAG)
SNDPGMMSG MSGID(CPF0002) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)
 ENDDO
        ELSE
 .
 .
 .
           ENDPGM
```

명령 정의 및 작성의 예

이 섹션에는 명령 정의 및 작성의 예가 있습니다.

어플리케이션 프로그램 호출

사용자는 어플리케이션 프로그램을 호출하는 명령을 작성할 수 있습니다. 어플리케이션 프로그램을 호출하는 명령을 작성하면 OS/400은 프로그램으로 전달된 매개변수에 대해 유효성 검사를 수행합니다. 그러나 어플리케이션 프로그램을 호출하는 데 CALL 명령을 사용하면 어플리케이션 프로그램은 반드시 유효성 검사를 실행해야 합니다.

예를 들면, 레이블 기록 프로그램(LBLWRT)은 1 또는 2부분 형식으로 특정 고객의 레이블 수를 기록합니다. LBLWRT 프로그램이 실행될 경우 이 프로그램은 세 개의 매개변수(고객 번호, 레이블 번호 및 사용되어야 하는 양식 유형(ONE 또는 TWO))를 필요로 합니다.

프로그램이 화면으로부터 직접 호출되었으면 두 번째 매개변수가 프로그램에 대해 틀린 형식을 갖게 됩니다. CALL 명령의 숫자 상수는 15자리(소수 5자릿수)이며 LBLWRT 프로그램은 소수 자릿수가 없는 3자릿수를 기대합니다. 프로그램에서 요구하는 형식으로 자료를 제공하는 명령을 작성할 수 있습니다.

다음은 LBLWRT 프로그램을 호출하는 명령에 대한 명령 정의문입니다.

```
CMD PROMPT('Label Writing Program')
PARM  KWD(CUSNBR) TYPE(*CHAR) LEN(5) MIN(1) +
      PROMPT('Customer Number')
```

```
PARM  KWD(COUNT) TYPE(*DEC) LEN(3) DFT(20) RANGE(10 150) +
      PROMPT('Number of Labels')
PARM  KWD(FRMTYP) TYPE(*CHAR) LEN(3) DFT('TWO') RSTD(*YES) +
      SPCVAL(('ONE') ('TWO') ('1' 'ONE') ('2' 'TWO')) +
      PROMPT('Form Type')
```

두 번째 매개변수 COUNT에는 디폴트 값 20이 지정되며 RANGE 매개변수는 레이블 수에 입력될 수 있는 10에서 150 사이의 값만 허용합니다.

SPCVAL 매개변수는 표시장치 사용자가 세 번째 매개변수 FRMTYP에 'ONE', 'TWO', '1' 또는 '2'를 입력할 수 있도록 허용합니다. 프로그램은 값 'ONE' 또는 'TWO'를 예상합니다. 그러나 표시장치 사용자가 '1' 또는 '2'를 입력하면 명령은 FRMTYP 매개변수 값을 대체합니다.

이 명령의 명령 처리 프로그램은 어플리케이션 프로그램 LBLWRT입니다. 어플리케이션 프로그램이 RPG OS/400 프로그램이면 매개변수를 수신하기 위해 프로그램 안에 다음 스펙이 작성됩니다.

```
*ENTRY   PLIST
         PARM    CUST   5
         PARM    COUNT  30
         PARM    FORM   3
```

다음은 CRTCMD 명령입니다.

```
CRTCMD CMD(LBLWRT) PGM(LBLWRT) SRCMBR(LBLWRT)
```

디폴트 값 대체

사용자는 IBM 제공 명령에 디폴트를 제공하여 표시장치 사용자가 해야 하는 입력을 줄여주는 명령을 작성할 수 있습니다. 예를 들면, 테이프를 초기화하여 테이프 장치 TAPE1에 라이브러리를 저장하는 SAVLIBTAP(테이프에 라이브러리 저장) 명령을 작성할 수 있습니다. 이 명령은 표준 SAVLIB(라이브러리 저장) 명령 매개변수에 대한 디폴트를 제공하며 표시장치 사용자는 라이브러리명만 지정하도록 요구합니다.

SAVLIBTAP 명령에 대한 명령 정의문은 다음과 같습니다.

```
CMD PROMPT('Save Library to Tape')
PARM KWD(LIB) TYPE(*NAME) LEN(10) MIN(1) +
  PROMPT('Library Name')
```

명령 처리 프로그램은 다음과 같습니다.

```
PGM PARM(&LIB)
DCL &LIB TYPE(*CHAR) LEN(10)
INZTAP DEV(TAPE1) CHECK(*NO)
SAVLIB LIB(&LIB) DEV(TAPE1)
           ENDPGM
```

다음은 CRTCMD 명령입니다.

```
CRTCMD CMD(SAVLIBTAP) PGM(SAVLIBTAP) SRCMBR(SAVLIBTAP)
```

출력 대기행렬 표시

사용자는 출력 대기행렬 PGMR을 표시하기 위해 디폴트인 출력 대기행렬을 표시하는 명령을 작성할 수 있습니다. 또한 표시장치 사용자는 DSPOQ 명령을 사용하여 라이브러리 리스트상의 대기행렬을 표시할 수 있습니다. DSPOQ 명령은 인쇄 옵션도 제공합니다.

다음은 DSPOQ 명령에 대한 명령 정의문입니다.

```
         CMD PROMPT('WRKOUTQ.-Default to PGMR')
PARM  KWD(OUTQ) TYPE(*NAME) LEN(10) DFT(PGMR) +
      PROMPT('Output queue')
PARM  KWD(OUTPUT) TYPE(*CHAR) LEN(6) DFT(*) RSTD(*YES)
      VALUES(* *PRINT) PROMPT('Output')
```

두 번째 PARM문의 RSTD 매개변수는 항목이 값의 리스트 중 하나가 되도록 지정합니다.

다음은 DSPOQ 명령에 대한 명령 처리 프로그램입니다.

```
PGM PARM(&OUTQ &OUTPUT)
DCL &OUTQ TYPE(*CHAR) LEN(10)
DCL &OUTPUT TYPE(*CHAR) LEN(6)
WRKOUTQ OUTQ(*LIBL/&OUTQ) OUTPUT(&OUTPUT)
           ENDPGM
```

다음은 CRTCMD 명령입니다.

```
CRTCMD CMD(DSPOQ) PGM(DSPOQ) SRCMBR(DSPOQ)
```

다음의 DSPOQ1 명령은 앞에 나온 명령을 변환한 것입니다. 워크스테이션 사용자는 이 명령을 사용하여 출력 대기행렬명으로 규정된 이름을 입력할 수 있습니다. 라이브러리명에 대한 명령의 디폴트는 *LIBL입니다.

다음은 DSPOQ1 명령에 대한 명령 정의문입니다.

```
                 CMD PROMPT('WRKOUTQ.-Default to PGMR')
         PARM    KWD(OUTQ) TYPE(QUAL1) +
                 PROMPT('Output queue:')
         PARM    KWD(OUTPUT) TYPE(*CHAR) LEN(6) RSTD(*YES) +
                 VALUES(* *PRINT) DFT(*) +
                 PROMPT('Output')
QUAL1:   QUAL TYPE(*NAME) LEN(10) DFT(PGMR)
         QUAL TYPE(*NAME) LEN(10) DFT(*LIBL) +
         SPCVAL(*LIBL)
```

QUAL문은 OUTQ 매개변수에 입력할 수 있는 규정된 이름을 정의하는 데 사용됩니다. 사용자가 이름을 입력하지 않으면 *LIBL/PGMR이 사용됩니다. 모든 라이브러리명이 유효한 명명 규칙(예를 들면, A에서 Z까지의 문자로 시작)을 따라야 하는데, 값 *LIBL이 이 규칙을 어기므로 SPCVAL 매개변수가 사용됩니다. SPCVAL 매개변수는 *LIBL이 입력되면 OS/400 유효명 검사 규칙을 무시하도록 지정합니다.

DSPOQ1 명령에 대한 명령 처리 프로그램은 다음과 같습니다.

```
PGM PARM(&OUTQ &OUTPUT)
DCL &OUTQ TYPE(*CHAR) LEN(20)
DCL &OBJNAM TYPE(*CHAR) LEN(10)
DCL &LIB TYPE(*CHAR) LEN(10)
DCL &OUTPUT TYPE(*CHAR) LEN(6)
CHGVAR &OBJNAM %SUBSTRING(&OUTQ 1 10)
CHGVAR &LIB %SUBSTRING(&OUTQ 11 10)
WRKOUTQ OUTQ(&LIB/&OBJNAM) OUTPUT(&OUTPUT)
           ENDPGM
```

규정된 이름이 명령으로부터 20자 변수로 전달되므로 이 프로그램에서는 규정된 이름을 적합한 CL 구문으로 입력하기 위해 서브스트링 내장 기능(%SUBSTRING 또는 %SST)을 사용해야 합니다.

IBM 명령으로부터의 메세지 재표시

CLROUTQ 명령은 삭제된 항목의 수, 삭제되지 않은 항목의 수 및 출력 대기행렬의 이름을 설명하는 완료 메세지 CPF3417을 발행합니다. CLROUTQ 명령이 CPP 안에서 수행되면 메세지는 여전히 발행되나 CPP에 의해 직접 발행되지는 않으므로 상세 메세지가 됩니다. 예를 들어, 사용자 정의 CLROUTQ 명령이 프로그래머 메뉴로부터 발행된 경우 메세지는 표시되지 않습니다. 그러나 IBM 메세지를 수신하여 사용자의 CPP로부터 다시 발행할 수 있습니다.

예를 들면, 출력 대기행렬 QPRINT2를 지우기 위해 명령 CQ2를 작성합니다.

다음은 CQ2 명령에 대한 명령 정의문입니다.

```
CMD PROMPT ('Clear QPRINT2 output queue')
```

다음은 CRTCMD 명령입니다.

```
CRTCMD CMD(CQ2) PGM(CQ2)
```

다음은 완료 메세지를 수신하고 이를 표시하는 CPP입니다.

```
PGM /* Clear QPRINT2 output queue CPP */
DCL &MSGID TYPE(*CHAR) LEN(7)
DCL &MSGDTA TYPE(*CHAR) LEN(100)
CLROUTQ QPRINT2
RCVMSG MSGID(&MSGID) MSGDTA(&MSGDTA) MSGTYPE(*COMP)
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) MSGTYPE(*COMP)
           ENDPGM
```

메세지 CPF3417에 대한 MSGDTA 길이는 28바이트입니다. 그러나 변수 &MSGDTA를 100바이트로 정의하여 사용되지 않은 위치를 무시할 수 있으므로 대부분의 메세지에 대해 동일한 접근을 사용할 수 있습니다.

약어 명령 작성

예 1

IBM이 제공한 명령들을 단순화시키거나 사용자들을 위해 허용된 매개변수들을 제한하기 위하여 사용자 자신의 약어 명령을 작성할 수 있습니다. 예를 들어, 사용자들에게 단지 인쇄 장치 매개변수만 변경하도록 허용하기 위해 사용자는 자신의 CJ(작업 변경) 명령을 작성할 수 있습니다. 다음은 사용자 자신의 CJ 명령을 작성하고 구축하기 위한 세 가지 단계입니다.

- 1단계: 명령 정의 소스문

```
          CMD   PROMPT('Change Job')

      PARM  KWD(PRTDEV) +
            TYPE(*NAME) +
            LEN(10)  +
            SPCVAL(*SAME *USRPRF *SYSVAL *WRKSTN) +
            PROMPT('Printer Device')
```

- 2단계: 처리 프로그램

```
          PGM PARM(&PRTDEV)
      DCL VAR(&PRTDEV)  TYPE(*CHAR)  LEN(10)
      CHGJOB  PRTDEV(&PRTDEV)
             ENDPGM
```

- 3단계: CRTCMD 명령

```
          CRTCMD  CMD(CJ)  PGM(CJ)  SRCMBR(CJ)
```

예 2

사용자는 프린터 출력기 W1을 시작하기 위해 DW1이라는 약어 명령을 작성할 수 있습니다.

명령 정의문은 다음과 같습니다.

```
CMD /* Start printer writer command */
```

명령 처리 프로그램은 다음과 같습니다.

```
          PGM
STRPRTWTR DEV(QSYSPRT) OUTQ(QPRINT) WTR(W1)
            ENDPGM
```

다음은 CRTCMD 명령입니다.

```
CRTCMD CMD(DW1) PGM(DW1) SRCMBR(DW1)
```

파일 및 소스 멤버 삭제

사용자는 파일과 QDDSSRC에서 파일에 상응하는 소스 멤버를 삭제하기 위한 명령을 작성할 수 있습니다.

다음은 명령 DFS에 대한 명령 정의문입니다.

```
CMD PROMPT('Delete File and Source')
PARM KWD(FILE) TYPE(*NAME) LEN(10) PROMPT('File Name')
```

이 명령 처리 프로그램은 파일명과 소스 파일 멤버가 동일하다는 가정하에 작성됩니다. 또한 이 프로그램은 파일과 소스 파일이 모두 라이브러리 리스트에 있다고 가정합니다. 프로그램이 파일을 삭제할 수 없으면, 정보 메세지가 송신되며, 명령은 소스 멤버의 제거를 시도합니다. 소스 멤버가 존재하지 않으면 이탈 메세지가 송신됩니다.

명령 처리 프로그램은 다음과 같습니다.

```
PGM PARM(&FILE)
DCL &FILE TYPE(*CHAR) LEN(10)
DCL &MSGID TYPE(*CHAR) LEN(7)
DCL &MSGDTA TYPE(*CHAR) LEN(80)
DCL &SRCFILE TYPE(*CHAR) LEN(10)
MONMSG MSGID(CPF0000) EXEC(GOTO ERROR) /* CATCH ALL */
DLTF &FILE
MONMSG MSGID(CPF2105) EXEC(DO) /* NOT FOUND*/
RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*INFO) +
            MSGDTA(&MSGDTA)
GOTO TRYDDS
 ENDDO
RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
    /* DELETE FILE COMPLETED */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
          MSGDTA(&MSGDTA) /* TRY IN QDDSSRC FILE */
TRYDDS:   CHKOBJ QDDSSRC OBJTYPE(*FILE) MBR(&FILE)
          RMVM QDDSSRC MBR(&FILE)
          CHGVAR &SRCFILE 'QDDSSRC'
         GOTO END
END:      RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
             /* REMOVE MEMBER COMPLETED */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
             MSGDTA(&MSGDTA)
          RETURN
ERROR:    RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
             /* ESCAPE MESSAGE */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
             MSGDTA(&MSGDTA)
           ENDPGM
```

프로그램 오브젝트 삭제

사용자는 HLL 프로그램과 이 프로그램에 상응하는 소스 멤버를 삭제하기 위한 명령을 작성할 수 있습니다.

다음은 명령 DPS에 대한 명령 정의문입니다.

```
CMD PROMPT ('Delete Program and Source')
PARM KWD(PGM) TYPE(*NAME) LEN(10) PROMPT('Program Name')
```

이 명령 처리 프로그램은 프로그램명과 소스 파일 멤버가 동일하다는 가정하에 작성된 것입니다. 또한 반드시 QCLSRC, QRPGSRC, QCBLSRC의 IBM 제공 소스 파일을

사용해야 합니다. 그리고 프로그램은 프로그램과 소스 파일이 모두 라이브러리 리스트에 있다고 가정한 것입니다. 프로그램을 열지 못할 경우 시스템이 정보 메세지를 송신하며 명령이 소스 멤버를 제거하려 시도합니다. 소스 멤버가 없으면, 시스템이 이탈 메세지를 송신합니다. 명령 처리 프로그램은 다음과 같습니다.

```
PGM PARM(&PGM)
DCL &PGM TYPE(*CHAR) LEN(10)
DCL &MSGID TYPE(*CHAR) LEN(7)
DCL &MSGDTA TYPE(*CHAR) LEN(80)
DCL &SRCFILE TYPE(*CHAR) LEN(10)
MONMSG MSGID(CPF0000) EXEC(GOTO ERROR) /* CATCH ALL */
DLTPGM &PGM
MONMSG MSGID(CPF2105) EXEC(DO) /* NOT FOUND*/
RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*INFO) +
            MSGDTA(&MSGDTA)
GOTO TRYCL  /* TRY TO DELETE SOURCE MEMBER */
 ENDDO
RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
    /* DELETE PROGRAM COMPLETED */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
     MSGDTA(&MSGDTA) /* TRY IN QCLSRC */
TRYCL:   CHKOBJ QCLSRC OBJTYPE(*FILE) MBR(&PGM)
         MONMSG MSGID(CPF9815) EXEC(GOTO TRYRPG) /* NO CL MEMBER */
         RMVM QCLSRC MBR(&PGM)
         CHGVAR &SRCFILE 'QCLSRC'
         GOTO END
TRYRPG:   /* TRY IN QRPGSRC FILE */
         CHKOBJ QRPGSRC OBJTYPE(*FILE) MBR(&PGM)
         MONMSG MSGID(CPF9815) EXEC(GOTO TRYCBL) /* NO RPG MEMBER */
         RMVM QRPGSRC MBR(&PGM)
         CHGVAR &SRCFILE 'QRPGSRC'
         GOTO END
TRYCBL:   /* TRY IN QCBLSRC FILE */
         CHKOBJ QCBLSRC OBJTYPE(*FILE) MBR(&PGM)
         /* ON LAST SOURCE FILE LET CPF0000 OCCUR FOR A NOT FOUND +
         CONDITION */
         RMVM QCBLSRC MBR(&PGM)
         CHGVAR &SRCFILE 'QCBLSRC'
         GOTO END
TRYNXT:   /* INSERT ANY ADDITIONAL SOURCE FILES */
          /* ADD MONMSG AFTER CHKOBJ IN TRYCBL AS WAS +
            DONE IN TRYCL AND TRYRPG */
END:      RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
               /*REMOVE MEMBER COMPLETED */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
             MSGDTA(&MSGDTA)
          RETURN
ERROR:    RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
              /* ESCAPE MESSAGE */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
             MSGDTA(&MSGDTA)
           ENDPGM
```

제 10 장 명령 문서화

사용자 고유의 CL 명령을 정의하고 작성하도록 선택한 경우 339 페이지의 제 9 장『명령 정의』부분에 설명된 대로 명령을 설명하는 데 도움이 되는 온라인 명령 도움말을 작성할 수 있습니다. 이 장에서는 사용자 고유의 명령에 대한 온라인 도움말 작성 방법 및 온라인 명령 도움말을 사용한 HTML 명령 문서 작성 방법에 대해 설명합니다.

명령 및 명령 도움말

명령 프롬팅 및 온라인 명령 도움말은 CL 명령의 강력한 기능입니다. 하나 이상의 사용자 CL 명령을 개발한 경우 IBM CL 명령이 사용하는 모든 명령 프롬팅 기능을 사용자 명령에 사용할 수 있습니다. 명령 도움말의 경우에도 마찬가지입니다. 즉, 사용자의 명령을 설명하는 도움말을 작성할 수 있습니다.

첫 번째 단계는 명령과 명령 도움말 간의 연결 작동 방법을 이해하는 것입니다.

- 명령 도움말 정보는 패널 그룹 오브젝트에 저장됩니다. 패널 그룹의 기호 오브젝트 유형은 *PNLGRP입니다. 도움말 패널 그룹은 도움말 모듈로 구성되어 있습니다. 각 도움말 모듈에는 도움말 모듈명이 있습니다.
- 명령(*CMD) 오브젝트와 온라인 도움말 패널 그룹을 연결하는 CRTCMD(명령 작성) 명령에는 **HLPID**(도움말 ID)와 **HLPPNLGRP**(도움말 패널 그룹)의 두 매개변수가 사용됩니다.
- 하나의 온라인 도움말 패널 그룹에는 4가지 유형의 명령 도움말 모듈이 있습니다.
  1. 명령 레벨 도움말 모듈
  2. 매개변수 레벨 도움말 모듈
  3. 명령 예 도움말 모듈
  4. 명령 오류 메세지 도움말 모듈

CL 명령의 도움말을 보려고 할 때(예: 명령이 프롬트된 상태에서 F1(도움말) 키를 누름) OS/400은 명령에 해당 명령과 연관된 도움말 패널 그룹이 있는지 여부를 판별합니다. HLPPNLGRP 매개변수에 패널 그룹명을 지정하는 명령이 작성되었거나 CHGCMD(명령 변경) 명령을 사용하여 도움말 패널 그룹이 해당 명령과 연관되도록 변경된 경우 패널 그룹에 저장된 도움말이 검색, 형식화 및 표시됩니다. OS/400은 도움말 패널 그룹에 있는 다음 도움말 모듈에서 도움말을 검색하려고 시도합니다.

- 명령을 작성하거나 변경할 때 HLPID 매개변수에 지정된 값과 이름이 같은 명령 레벨 도움말 모듈. 예를 들어, HLPID(STRPAY)를 지정하는 STRPAY 명령이 작성된 경우 OS/400은 이름이 STRPAY인 도움말 모듈을 찾습니다.

- 상수 매개변수를 제외하고 각 명령 매개변수에 대한 매개변수 레벨 도움말 모듈. 도움말 모듈명은 뒤에 슬래시 문자와 매개변수 키워드명이 차례로 나오는 명령의 HLPID 값이어야 합니다. 예를 들어, STRPAY 명령에 STRPAY의 HLPID 값과 이름이 TITLE인 매개변수가 있는 경우 OS/400은 이름이 STRPAY/TITLE인 매개변수 레벨 도움말 모듈을 찾습니다.
- 명령을 사용하는 하나 이상의 예를 포함하는 도움말 모듈. 도움말 모듈명은 뒤에 슬래시 문자와 COMMAND/EXAMPLES가 차례로 나오는 명령의 HLPID 값이어야 합니다. 예를 들어, STRPAY 명령에 STRPAY의 HLPID 값이 있는 경우 OS/400은 이름이 STRPAY/COMMAND/EXAMPLES인 명령 예 도움말 모듈을 찾습니다.
- 명령에서 부호화될 수 있는 모니터 가능한 메세지 리스트가 있는 도움말 모듈. 도움말 모듈명은 뒤에 슬래시 문자와 ERROR/MESSAGES가 차례로 나오는 명령의 HLPID 값이어야 합니다. 예를 들어, STRPAY 명령에 STRPAY의 HLPID 값이 있는 경우 OS/400은 이름이 STRPAY/ERROR/MESSAGES인 명령 예 도움말 모듈을 찾습니다.

5250 단말기에서 확장 CL 명령 도움말을 볼 때 또는 5250 에뮬레이터 소프트웨어를 사용할 때 명령 레벨 및 매개변수 레벨 도움말 섹션이 필요하며 예 및 오류 메세지 도움말 섹션은 선택사항입니다. 온라인 도움말 패널 그룹에서 명령 레벨 도움말 섹션이나 매개변수 레벨 도움말 섹션을 찾을 수 없는 경우 도움말 화면의 하단에 진단 메세지 CPF6E01(도움말 정보가 불완전합니다)이 표시됩니다. 예 또는 오류 메세지 도움말 모듈이 없으면 진단 메세지가 표시되지 않습니다.

명령 도움말 작성

명령(*CMD) 오브젝트를 온라인 도움말 패널 그룹(*PNLGRP) 오브젝트에 연결하는 방법을 이해하고 나면 명령에 대한 네 가지 유형의 도움말 모듈에 포함되는 도움말 텍스트를 실제로 작성할 수 있습니다. 명령의 iSeries 온라인 도움말은 사용자 인터페이스 관리 프로그램(UIM)으로 알려진 태그 언어로 작성됩니다. UIM 소스는 CRTPNLGRP(패널 그룹 작성) 명령을 사용하여 컴파일되어 *PNLGRP 오브젝트를 작성합니다. 도움말 패널 그룹 및 UIM 패널 그룹 정의 언어에 대한 자세한 정보는 Application Display Programming  책을 참조하십시오.

명령 도움말의 UIM 소스 생성

UIM 구문을 보다 쉽게 배울 수 있는 한 방법으로 GENCMDDOC(명령 문서 생성) 명령을 사용할 수 있습니다. 이 명령을 사용하면 UIM 소스가 포함된 파일을 작성할 수 있습니다. 이 파일은 온라인 명령 도움말에 대한 템플리트를 제공합니다. UIM 소스를 작성하려면 GENOPT(생성 옵션) 매개변수에 대해 *UIM을 지정해야 합니다. 템플리

트 작성에 사용된 정보는 사용자가 지정한 명령 오브젝트(*CMD)에서 검색됩니다. GENCMDDOC를 사용하여 UIM 소스를 작성하면 CL 명령의 온라인 도움말을 쉽게 작성할 수 있습니다.

GENCMDDOC에 대한 자세한 정보는, iSeries Information Center의 CL 주제에 있는 명령 문서를 참조하십시오.

다음은 GENCMDDOC 명령을 사용하여 UIM 템플리트를 생성하는 예입니다.

```
GENCMDDOC   CMD(MYLIB/MYCMD)
            TODIR('/QSYS.LIB/MYLIB.LIB/QPNLSRC.FILE')
            TOSTMF(*CMD)  GENOPT(*UIM)
```

위 예에서, 명령은 라이브러리 MYLIB에서 이름이 MYCMD인 명령 오브젝트로부터 정보를 검색하고 라이브러리 MYLIB에서 소스 파일 QPNLSRC의 멤버 MYCMD로 UIM 소스를 생성합니다. 템플리트 UIM 소스를 생성한 후 다음과 같이 UIM 소스를 편집해야 합니다.

- 명령 레벨 도움말 모듈은 명령의 목적을 설명해야 합니다. 첫 번째 구문의 시작 부분이 제공되며 사용자가 적절한 텍스트로 대체해야 하는 <...> 마커가 뒤에 나옵니다. 먼저 실행해야 할 명령이나 필요한 특수 권한과 같이 명령 실행에 적용되는 제한사항이 설명되어야 합니다. 일부 제한사항 예가 제공되며 이러한 예를 사용자 명령의 제한사항에 맞게 편집해야 합니다.
- 각 매개변수 레벨 도움말 모듈은 매개변수의 목적을 설명해야 합니다. 첫 번째 구문의 시작 부분이 제공되며 사용자가 적절한 텍스트로 대체해야 하는 <...> 마커가 뒤에 나옵니다. 내부 매개변수 종속성이나 제한사항을 설명하려면 이러한 매개변수 레벨 제한사항을 위해 NT. 및 :ENT.(주의 끝) UIM 태그를 사용할 수 있습니다.

  매개변수 레벨 도움말 모듈은 매개변수에서 사용할 수 있는 선택사항에 대한 설명을 제공해야 합니다. 각 특수 값이나 단일 값의 머리말이 제공되지만 <...> 마커를 매개변수 값 설명으로 대체해야 합니다.
- 필요하면 예를 제공해야 합니다. 명령 예 도움말 모듈에는 두 개의 예에 대한 머리말이 제공됩니다. 명령에 매개변수가 없는 경우 하나의 예만을 가지도록 이 도움말 모듈을 편집할 수 있습니다. 명령에 여러 매개변수가 있거나 명령이 여러 개의 고유한 기능을 제공하는 경우 머리말을 복사하여 추가 명령 예를 작성할 수 있습니다. 사용자 고유 명령의 매개변수 키워드 및 매개변수 값을 삽입하고 <...> 마커가 명령 예의 수행 사항에 대한 설명으로 대체되도록 명령 예를 편집해야 합니다.
- 필요하면 오류 메세지 텍스트를 제공해야 합니다. 명령 오류 메세지 도움말 모듈에는 &MSG 내장 기능을 사용하여 명령이 송신한 오류 메세지의 메세지 텍스트를 삽입하는 방법을 보여주는 머리말이 제공됩니다. 메세지 설명을 포함하는 메세지 파일과 함께, 명령에서 신호한 메세지의 실제 메세지 ID가 포함되도록 메세지 리스트를 편집해야 합니다.

사용자 명령에 맞게 UIM 소스를 편집한 후에는 CRTPNLGRP(패널 그룹 작성) 명령을 사용하여 온라인 도움말 패널 그룹을 작성할 수 있습니다. 다음은 CRTPNLGRP 명령을 사용하는 예입니다.

```
CRTPNLGRP   PNLGRP(MYLIB/MYCMD)
            SRCFILE(MYLIB/QPNLSRC)  SRCMBR(MYCMD)
```

이 명령은 라이브러리 MYLIB에서 소스 실제 파일 QPNLSRC에 있는 멤버 MYCMD의 UIM 소스에서 패널 그룹을 작성하려고 시도합니다. UIM 소스를 컴파일할 때 심각한 오류가 발생하지 않으면 라이브러리 MYCMD에 이름이 MYCMD인 패널 그룹 (*PNLGRP) 오브젝트가 작성됩니다. 명령은 스풀 파일을 생성합니다. 이 파일은 UIM 컴파일러에서 찾은 정보, 경고 및 심각한 오류를 확인하는 경우 볼 수 있습니다.

공통 도움말 공유

이전 섹션에서는 오직 하나의 명령에 대한 도움말 모듈을 포함하는 패널 그룹을 작성하는 방법에 대해 설명했습니다. 명령 온라인 도움말 모듈을 명령과 연결하는 데 사용되는 명명 구조 때문에 다수의 명령에 대한 도움말 모듈을 하나의 패널 그룹에 저장할 수 있습니다. 여러 개의 관련된 명령에 대한 도움말 모듈이 하나의 패널 그룹에 있으면 :IMHELP.(삽입된 도움말) UIM 태그를 사용하여 공통 도움말 정보를 공유할 수 있습니다. 예를 들어, 각 명령에 대한 일반적인 설명이 동일한 OUTPUT이라는 매개변수를 갖는 명령이 여러 개 있을 수 있습니다. 동일한 명령 도움말 정보를 공유하면 이 매개변수를 갖는 모든 명령들 사이에서 일관성을 유지할 수 있습니다.

온라인 도움말 텍스트를 공유하기 위한 또 다른 옵션은 :IMPORT.(가져오기) UIM 태그를 사용하는 것입니다. :IMHELP 태그를 사용하여 동적으로 삽입할 수 있는 또 다른 패널 그룹의 도움말 모듈을 하나 이상 정의할 수 있습니다.

:IMHELP. 및 :IMPORT. UIM 태그에 대한 자세한 정보는 Application Display Programming 부록을 참조하십시오.

도움말 모듈에 도움말 텍스트 구성

사용자 패널 그룹의 도움말 모듈에 포함되는 도움말 텍스트를 구성하는 방법을 선택할 수 있습니다. UIM 기능을 통해 :IMHELP. 태그를 사용하여 도움말을 하나의 도움말 모듈에서 하나 이상의 다른 도움말 모듈로 삽입하면 둘 이상의 도움말 모듈들 중에서 이전에 정의한 네 가지 유형의 명령 도움말 모듈에 대한 도움말 텍스트를 나눌 수 있습니다. 도움말 텍스트를 더 작은 도움말 모듈로 분할하는 가장 일반적인 이유는 공통 도움말 텍스트의 공유를 용이하게 하기 위함입니다.

다른 도움말 모듈에 삽입될 도움말 모듈을 정의할 때 온라인 명령 도움말 모듈의 네 가지 유형 중 하나와 혼동되지 않을 도움말 모듈명을 선택해야 합니다.

명령 문서의 HTML 소스 생성

사용자 명령에 대한 도움말 정보를 작성하는 것 외에 iSeries 시스템에 연결되지 않았을 때 보거나 인쇄할 수 있는 명령 문서를 작성할 수도 있습니다. OS/400을 사용하면 인터넷 브라우저를 사용하여 표시하거나 여러 워드 프로세싱 프로그램으로 가져올 수 있는 HTML 소스로 된 명령 문서를 생성할 수 있습니다. 이를 수행하려면 GENOPT(생성 옵션) 매개변수에 대해 *HTML을 지정하는 GENCMDDOC(명령 문서 생성) 명령을 사용할 수 있습니다. 이 명령은 사용자가 지정한 명령 오브젝트(*CMD) 및 명령에 이미 존재하는 명령 도움말 패널 그룹(*PNLGRP) 오브젝트에서 검색되는 정보가 포함된 파일을 생성합니다.

GENCMDDOC 사용에 대한 자세한 정보는, iSeries Information Center의 CL 주제에 있는 명령 문서를 참조하십시오.

HTML 출력 모양과 이러한 모양이 온라인 도움말과 일치하는지 확인하려면 임의의 명령에 대한 도움말과 iSeries Information Center에서 제공하는 CL 명령 문서의 도움말을 비교하십시오. IBM은 온라인 명령 도움말로부터 Information Center 명령 문서를 빌드합니다.

다음은 GENCMDDOC 명령을 사용하여 HTML 소스를 생성하는 예입니다.

```
GENCMDDOC   CMD(MYLIB/MYCMD)
```

이 예에서 명령은 라이브러리 MYLIB에서 이름이 MYCMD인 명령 오브젝트에서 정보를 검색하고 작업의 현재 작업 디렉토리에서 HTML 소스를 스트림 파일 MYLIB_MYCMD.HTML로 생성합니다. 생성된 스트림 파일은 DSPF(파일 표시)를 사용하여 HTML 소스 형식이나 표준 인터넷 브라우저 소프트웨어를 사용하여 HTML 브라우저 형식으로 보거나 명령을 사용하여 볼 수 있습니다.

제 11 장 프로그램 디버깅

ILE 프로그램 디버깅

디버깅을 통해 프로그램의 오류를 검출, 진단 및 제거할 수 있습니다. 사용자가 ILE 소스 디버거를 통해 자신의 ILE 프로그램을 디버그할 수 있습니다. 이 장에서는 ILE 소스 디버거의 사용법을 설명합니다.

다음은 이 장에서 설명될 내용입니다.

- 디버깅을 위한 ILE 프로그램 준비
- 디버그 세션 시작
- 디버그 세션에서의 프로그램 추가 및 제거
- 디버그 세션에서의 프로그램 소스 보기
- 조건부 및 무조건부 중단점의 설정 및 제거
- 프로그램의 단계화
- 변수 값 표시
- 변수 값 변경
- 변수 속성 표시
- 단축명을 변수, 표현식 또는 디버그 명령과 같게 함

프로그램을 디버깅하고 테스트하는 동안, 테스트 자료가 들어 있는 테스트 라이브러리로 프로그램을 지시하여 기존 실제 자료에 아무런 영향을 주지 않도록 사용자의 라이브러리 리스트가 변경되어 있는지 확인하십시오.

다음 중 하나의 명령을 사용하면 제품 라이브러리의 데이터베이스 파일이 뜻하지 않게 수정되는 것을 방지할 수 있습니다.

- STRDBG(디버그 시작) 명령을 사용하고 UPDPROD 매개변수에 대해 디폴트 *NO를 보유하십시오.
- CHGDBG(디버그 변경) 명령을 사용하십시오.

제시한 정보는 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

ILE 소스 디버거(프로그램이나 서비스 프로그램을 디버그하는 데 필요한 권한과 최적화 레벨의 영향 포함)에 대한 자세한 정보는 ILE 개념 책의 디버깅 고려사항 장을 참조하십시오.

ILE 소스 디버거

ILE 소스 디버거는 프로그램 오브젝트와 서비스 프로그램 안의 오류를 검출하고 제거하는 데 사용됩니다. 소스 디버거를 사용하여 다음을 수행할 수 있습니다.

- ILE CL 또는 혼합 ILE 언어 어플리케이션 디버그
- 프로그램 실행 시 디버그 명령을 사용하여 프로그램의 흐름 모니터
- 프로그램 소스 보기
- 조건부 및 무조건부 중단점의 설정 및 제거
- 지정된 명령문 숫자의 단계화
- 변수 값 표시 또는 변경
- 변수 속성 표시

프로그램이 중단점이나 단계 명령에 의해 중단되면 적용 가능한 모듈 오브젝트의 보기가 프로그램이 중단된 지점에서 화면에 표시됩니다. 이 지점에서 더 많은 디버그 명령을 입력할 수 있습니다.

소스 디버거를 사용하기 전에 CRTCLMOD(CL 모듈 작성) 또는 CRTBNDCL(바인드된 CL 작성)을 사용하여 모듈 오브젝트나 프로그램 오브젝트를 작성할 때 디버그 옵션(DBGVIEW)을 사용해야 합니다. 중단점이나 다른 ILE 소스 디버거 옵션을 설정한 후 프로그램을 호출할 수 있습니다.

디버그 명령

ILE 소스 디버거의 사용을 위해 여러 가지 디버그 명령을 사용할 수 있습니다. 디버그 명령 및 그 매개변수는 '모듈 소스 표시' 및 '표현식 평가' 화면의 맨 아래에 표시된 디버그 명령 행에 입력됩니다. 이 명령들은 대소문자 또는 혼합 문자로 입력할 수 있습니다.

주: 소스 디버거 명령 행에 입력된 디버그 명령은 CL 명령이 아닙니다.

표 10에 이러한 디버그 명령들이 요약되어 있습니다. ILE 소스 디버거의 온라인 도움말에서 디버그 명령 및 허용된 약어를 설명하고 있습니다.

표 10. ILE 소스 디버거 명령

디버그 명령	설명
ATTR	사용자가 변수 속성을 표시할 수 있도록 합니다. 속성은 디버그 기호 표에 기록된 것과 같은 변수 *크기*와 유형입니다.
BREAK	테스트 중인 프로그램의 위치에 조건부 또는 무조건부 중단점을 입력할 수 있도록 합니다. 조건부 중단점을 입력하려면 BREAK *위치* WHEN *표현식*을 사용하십시오.
SBREAK	테스트 중인 프로그램의 한 위치에 서비스 입력점을 입력할 수 있도록 합니다. 서비스 입력점은 시스템 디버거가 파생 작업의 제어를 확보할 수 있도록 프로그램에 지정하는 중단점 유형입니다. 중단점은 서비스 입력점에 맞는 작업이 현재 디버그 상태에 없을 경우에만 신호를 보냅니다.

표 10. ILE 소스 디버거 명령 (계속)

디버그 명령	설명
CLEAR	조건부 및 무조건부 중단점을 제거할 수 있도록 합니다.
DISPLAY	EQUATE 명령을 사용하여 할당한 이름과 정의를 표시할 수 있도록 합니다. 또한 현재 '모듈 소스 표시' 화면에 표시된 것과 다른 소스 모듈을 표시할 수 있도록 합니다. 모듈 오브젝트는 현재 프로그램 오브젝트에 존재해야 합니다.
EQUATE	단축형으로 사용할 이름으로 표현식, 변수 또는 디버그 명령을 할당할 수 있도록 합니다.
EVAL	변수 값을 표시하거나 변경하고 표현식 값을 표시할 수 있도록 합니다.
QUAL	후속 EVAL 명령에 나오는 변수 범위를 정의할 수 있도록 합니다.
STEP	디버그 중인 프로그램 중 하나 이상의 명령문을 실행할 수 있도록 합니다.
FIND	지정된 행 번호나 스트링 또는 텍스트용으로 현재 표시된 모듈을 탐색합니다.
UP	소스의 표시된 창을 입력된 양만큼 따라 보기의 처음으로 이동합니다.
DOWN	소스의 표시된 창을 입력된 양만큼 따라 보기의 끝으로 이동합니다.
LEFT	소스의 표시된 창을 입력된 문자 수만큼 왼쪽으로 이동합니다.
RIGHT	소스의 표시된 창을 입력된 문자 수만큼 오른쪽으로 이동합니다.
TOP	첫 번째 행이 보이도록 보기를 놓습니다.
맨 아래	마지막 행이 보이도록 보기를 놓습니다.
NEXT	보기를 현재 표시된 소스의 다음 번 중단점에 놓습니다.
PREVIOUS	보기를 현재 표시된 소스의 이전 중단점에 놓습니다.
HELP	사용 가능한 소스 디버거 명령에 대한 온라인 도움말 정보를 보여줍니다.
SET™	현재 디버그 세션에서 모든 FIND 요구에 대해 대소문자 구분 여부를 지정합니다. 또한 갱신 실행 파일 값을 변경할 수 있도록 해줍니다.
WATCH	현재 활동 중인 감시 조건의 리스트를 표시합니다.

디버그 세션을 위한 프로그램 오브젝트 준비

ILE 소스 디버거를 사용하기 전에 CRTCLMOD 또는 CRTBNDCL 명령을 사용하여 DBGVIEW 옵션을 지정해야 합니다.

사용자가 디버그하려는 각각의 ILE CL 모듈 오브젝트에 대해, 다음 세 개의 보기 중 하나를 작성할 수 있습니다.

- 루트 소스 보기
- 리스팅 보기
- 명령문 보기

루트 소스 보기 사용

루트 소스 보기에는 소스 멤버의 소스문이 들어 있습니다.

ILE CL 컴파일러는 ILE 소스 디버거와 함께 루트 소스 보기를 사용하기 위해 모듈 오브젝트(*MODULE)가 작성되는 동안 루트 소스 보기를 작성합니다.

주: 모듈 오브젝트는 소스문을 보기로 복사하는 대신 루트 소스 멤버의 소스문 위치에 대한 참조를 사용하여 작성됩니다. 따라서 모듈 작성과 이 멤버에서 작성된 모듈의 디버깅 간 루트 소스 멤버를 수정, 이름 변경 또는 이동해서는 안됩니다.

루트 소스 보기를 사용하여 ILE CL 모듈 오브젝트를 디버그하려면 CRTCLMOD 또는 CRTBNDCL 명령에 대한 DBGVIEW 매개변수에서 *SOURCE 또는 *ALL 옵션을 사용하십시오.

루트 소스 보기를 작성하는 한 가지 방법은 다음과 같습니다.

```
CRTCLMOD
MODULE(MYLIB/MYPGM) SRCFILE(MYLIB/QCLLESRC) SRCMBR(MYPGM) TEXT('CL Program')
DBGVIEW(*SOURCE)
```

DBGVIEW 매개변수에 대한 *SOURCE와 함께 CRTCLMOD(CL 모듈 작성) 명령이 모듈 오브젝트 *MYPGM*에 대한 루트 소스 보기를 작성합니다.

리스팅 보기 사용

리스팅 보기는 ILE CL 컴파일러가 작성한 컴파일 리스트나 스풀 파일의 소스 코드 부분과 유사합니다.

리스팅 보기를 사용하여 ILE CL 모듈 오브젝트를 디버그하려면 모듈을 작성할 때 CRTCLMOD 또는 CRTBNDCL 명령에 대한 DBGVIEW 매개변수에서 *LIST 또는 *ALL 옵션을 사용하십시오.

리스팅 보기를 작성하는 한 가지 방법은 다음과 같습니다.

```
CRTCLMOD
MODULE(MYLIB/MYPGM) SRCFILE(MYLIB/QCLLESRC) SRCMBR(MYPGM) TEXT('CL Program')
DBGVIEW(*LIST)
```

명령문 보기 사용

명령문 보기에는 어떤 CL 소스 자료도 포함되어 있지 않습니다. 그러나 컴파일러 리스트에서 발견된 프로시듀어명과 명령문 숫자를 사용하여 중단점을 추가할 수 있습니다. 명령문 보기를 사용하여 ILE CL 모듈 오브젝트를 디버그하려면 컴파일러 리스트의 사본이 필요합니다.

주: ILE CL 모듈 오브젝트를 디버그하는 데 명령문 보기를 사용할 경우 어떤 자료도 '모듈 소스 표시' 화면에 표시되지 않습니다.

명령문 보기를 사용하여 ILE CL 모듈 오브젝트를 디버그하려면 모듈을 작성할 때 CRTCLMOD 또는 CRTBNDCL 명령에 대한 DBGVIEW 매개변수에서 *STMT, *SOURCE, *LIST 또는 *ALL 옵션을 사용하십시오.

명령문 보기를 작성하는 한 가지 방법은 다음과 같습니다.

```
CRTCLMOD
MODULE(MYLIB/MYPGM) SRCFILE(MYLIB/QLSRC) SRCMBR(MYPGM) TEXT('CL Program')
DBGVIEW(*STMT)
```

ILE 소스 디버거 시작

디버그 보기를 작성한 후에는 어플리케이션 디버깅을 시작할 수 있습니다.

ILE 소스 디버거를 시작하려면 STRDBG(디버그 시작) 명령을 사용하십시오. 일단 디버거가 시작되면 사용자가 ENDDBG(디버그 종료) 명령을 입력할 때까지 그 프로그램이 활동합니다.

초기에 디버그 세션에 20개의 프로그램 오브젝트와 20개의 서비스 프로그램을 추가할 수 있습니다. 이것은 STRDBG 명령에 프로그램(PGM) 및 서비스 프로그램(SRVPGM) 매개변수를 사용하여 수행할 수 있습니다. 프로그램 오브젝트들은 ILE나 원래 프로그램 모델(OPM: original program model) 프로그램의 임의의 조합일 수 있습니다. 세 개의 프로그램 오브젝트와 함께 디버그 세션을 시작하려면 다음을 입력하십시오.

```
STRDBG PGM(*LIBL/MYPGM1 *LIBL/MYPGM2 *LIBL/MYPGM3) SRVPGM(*LIBL/SRVPGM1 *LIBL/SRVPGM2)
DBGMODSRC(*YES)
```

주: 프로그램 오브젝트를 디버그 세션에 추가하려면 그 프로그램 오브젝트에 대한 *CHANGE 권한이 있어야 합니다.

STRDBG 명령을 입력한 후 ILE 프로그램 오브젝트에 대한 '모듈 소스 표시' 화면이 나옵니다. 디버그 자료를 사용하여 프로그램 오브젝트에 바인드된 첫 번째 모듈 오브젝트를 볼 수 있습니다.

OPM 프로그램을 디버그하기 위해 ILE 소스 디버거가 사용하는 옵션이 있습니다. OPM 프로그램에는 작성 시의 소스 디버그 자료가 포함되어 있습니다. CRTCLPGM(CL 프로그램 작성) 명령에 OPTION(*SRCDBG) 또는 OPTION(*LSTDBG) 매개변수를 지정하여 이 작업을 수행하십시오. 소스 디버그 자료는 실제로 프로그램 오브젝트의 일부입니다.

소스 디버그 자료가 들어 있는 OPM 프로그램을 작성시켜 ILE 소스 디버거에 추가하려면 STRDBG 명령에 프로그램(PGM) 및 OPM 소스 레벨 디버그(OPMSRC) 매개변수를 사용하십시오. 소스 디버그 자료와 함께 작성된 OPM 프로그램으로 디버그 세션을 시작하려면 다음을 입력하십시오.

```
STRDBG PGM(*LIBL/MYOPMPGM) OPMSRC(*YES) DSPMODSRC(*YES)
```

프로그램 오브젝트를 디버그 세션에 추가

세션을 시작한 후에도 추가의 프로그램 오브젝트를 디버그 세션에 추가할 수 있습니다.

디버그 세션에 ILE 프로그램 오브젝트와 서비스 프로그램을 추가하려면 옵션 1(프로그램 추가)을 사용하고 '모듈 리스트에 대한 작업' 화면의 첫 번째 행에서 프로그램 오브

젝트명을 입력하십시오. ILE 소스 디버거 명령 리스트를 보려면 420 페이지의 표 10을 참조하십시오. F14(모듈 리스트에 대한 작업) 키를 눌러 '모듈 소스 표시' 화면에서 '모듈 리스트에 대한 작업' 화면에 액세스할 수 있습니다. 서비스 프로그램을 추가하려면 디폴트 프로그램 유형을 *PGM에서 *SRVPGM으로 변경하십시오. 주어진 시간에 디버그 세션에 포함할 수 있는 ILE 프로그램 오브젝트와 서비스 프로그램의 수에는 제한이 없습니다.

그림 17. ILE 프로그램 오브젝트를 디버그 세션에 추가. Enter 키를 누르면, 프로그램 WEEKDAY2가 디버그 세션에 추가됩니다.

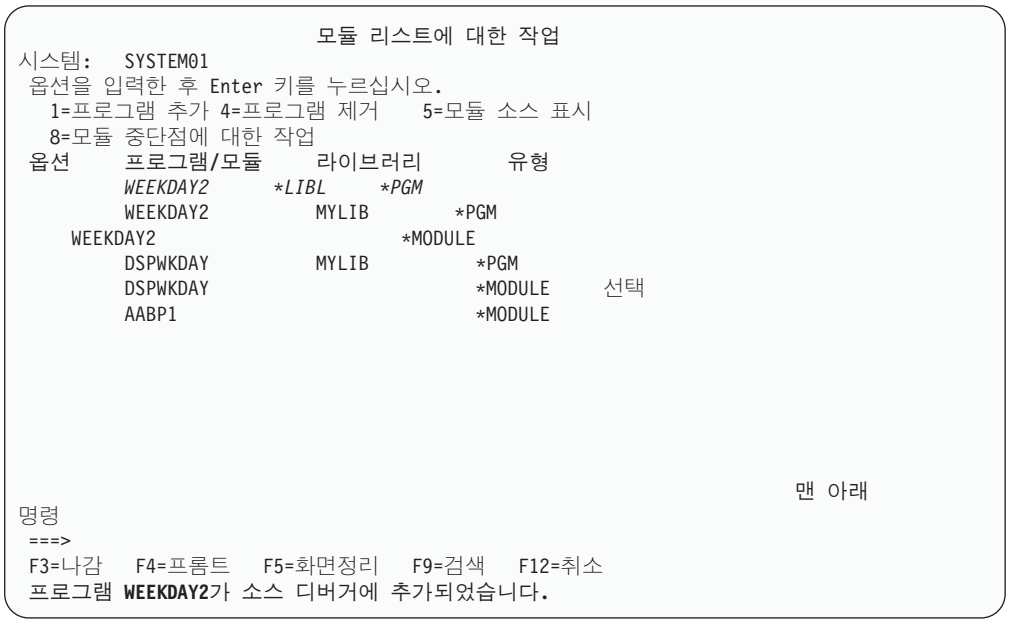


그림 18. ILE 프로그램 오브젝트를 디버그 세션에 추가. 화면의 맨 아래에 있는 정보 메세지는 프로그램 WEEKDAY2가 디버그 세션에 추가되었음을 보여줍니다.

디버그 세션에 프로그램 오브젝트를 추가하는 것이 완료되면 '모듈 리스트에 대한 작업' 화면에서 F3(나감) 키를 눌러 '모듈 소스 표시' 화면으로 리턴하십시오. 옵션 5(모듈 소스 표시)를 사용하여 모듈을 선택하고 표시할 수도 있습니다.

디버그 세션에 OPM 프로그램을 추가하려면 ADDPGM(프로그램 추가) 명령을 사용하십시오. 주어진 시간에서 디버그 세션에는 최대 20개의 OPM 프로그램이 포함될 수 있습니다. '모듈 리스트에 대한 작업' 화면에서 옵션 1(프로그램 추가)을 사용하면 소스 디버그 자료가 포함된 OPM 프로그램을 디버그 세션에 추가할 수 있습니다. (이것은 제공되는 디버그 세션에서 OPM 소스 레벨 디버깅을 허용할 경우에만 해당됩니다.) OPM 소스 레벨 디버깅을 허용하려면 디버그 세션을 시작하여 STRDGB 명령에 OPMSRC 매개변수를 사용하십시오. OPMSRC 매개변수가 STRDBG 명령에 지정되지 않은 경우에는 OPM 소스 레벨 디버깅을 활성화하십시오. 이것은 CHGDBG(디버그 변경) 명령에 OPM 소스 레벨 디버그(OPMSRC) 매개변수를 사용하여 실행할 수 있습니다. SET 디버그 명령을 사용하여 OPM 소스 디버그 지원 옵션의 값을 변경해서 수행할 수도 있습니다.

디버그 세션에서 프로그램 오브젝트 제거

세션을 시작한 후 디버그 세션에서 프로그램 오브젝트를 제거할 수 있습니다.

ILE 프로그램 오브젝트와 서비스 프로그램을 디버그 세션에서 제거하려면 제거하려는 프로그램 오브젝트 옆의 옵션 4(프로그램 제거)를 '모듈 리스트에 대한 작업' 화면에서 사용하십시오. 426 페이지의 그림 19를 참조하십시오. F14(모듈 리스트에 대한 작업) 키를 눌러 '모듈 소스 표시' 화면에서 '모듈 리스트에 대한 작업' 화면에 액세스할 수 있습니다. 서비스 프로그램을 제거하려면 디폴트 프로그램 유형을 *PGM에서 *SRVPGM으로 변경하십시오.

```
                          모듈 리스트에 대한 작업
시스템:   SYSTEM01
 옵션을 입력한 후 Enter 키를 누르십시오.
   1=프로그램 추가 4=프로그램 제거    5=모듈 소스 표시
   8=모듈 중단점에 대한 작업
 옵션     프로그램/모듈     라이브러리        유형
               *LIBL     *PGM
  4         WEEKDAY2          MYLIB        *PGM
     WEEKDAY2                          *MODULE
          DSPWKDAY          MYLIB          *PGM
          DSPWKDAY                         *MODULE     선택
          AABP1                            *MODULE
                                                                  맨 아래
명령
 ===>
 F3=나감    F4=프롬트    F5=화면정리    F9=검색    F12=취소
```

그림 19. 디버그 세션에서 ILE 프로그램 오브젝트 제거. Enter 키를 누르면, WEEKDAY2가 디버그 세션에서 제거됩니다.

```
                          모듈 리스트에 대한 작업
시스템:   SYSTEM01
 옵션을 입력한 후 Enter 키를 누르십시오.
   1=프로그램 추가 4=프로그램 제거    5=모듈 소스 표시
   8=모듈 중단점에 대한 작업
 옵션     프로그램/모듈     라이브러리        유형
               *LIBL     *PGM
          DSPWKDAY          MYLIB          *PGM
          DSPWKDAY                         *MODULE     선택
          AABP1                            *MODULE
                                                                  맨 아래
명령
 ===>
 F3=나감    F4=프롬트    F5=화면정리    F9=검색    F12=취소
 프로그램 WEEKDAY2가 소스 디버거에서 제거되었습니다.
```

그림 20. 디버그 세션에서 ILE 프로그램 오브젝트 제거

디버그 세션에서 프로그램 오브젝트를 제거하는 것이 완료되면 '모듈 리스트에 대한 작업' 화면에서 F3(나감) 키를 눌러 '모듈 소스 표시' 화면으로 리턴하십시오.

주: 디버그 세션에서 그것을 제거하기 위해서는 사용자에게 프로그램에 대한 *CHANGE 권한이 있어야 합니다.

디버그 세션에서 OPM 프로그램을 제거하려면 RMVPGM(프로그램 제거) 명령을 사용하십시오. OPM 소스 레벨 디버깅이 활동 중인 경우에는 소스 디버그 자료와 함께 작

성된 OPM 프로그램이 '모듈 리스트에 대한 작업' 화면에 나올 수 있습니다. '모듈 리스트에 대한 작업' 화면에서 옵션 4(프로그램 제거)를 사용하면 이 프로그램들을 디버그 세션에서 제거할 수 있습니다.

프로그램 소스 보기

'모듈 소스 표시' 화면은 한번에 하나씩 모듈 오브젝트의 프로그램 오브젝트 소스를 보여줍니다. 다음 디버그 보기 옵션 중 하나를 사용하여 모듈 오브젝트가 컴파일된 경우 모듈 오브젝트의 소스를 볼 수 있습니다.

- DBGVIEW(*ALL)
- DBGVIEW(*SOURCE)
- DBGVIEW(*LISTING)

'모듈 소스 표시' 화면에서 보여지는 내용을 변경하는 데는 두 가지 방법이 있습니다.

- 보기 변경
- 모듈 변경

보기를 변경할 경우 ILE 소스 디버거는 변경할 보기 내의 동등한 위치에 맵핑됩니다. 모듈을 변경할 경우 표시된 보기상의 실행 가능한 명령문이 메모리에 저장되어 모듈이 다시 표시될 때 보여집니다. 중단점을 설정했던 행 번호가 강조표시됩니다. 중단점, 단계 또는 메세지 때문에 프로그램이 중단되고 화면이 표시되는 경우 이벤트가 발생한 소스 행이 강조표시됩니다.

모듈 오브젝트 변경

'모듈 리스트에 대한 작업' 화면에서 옵션 5(모듈 소스 표시)를 사용하여 '모듈 소스 표시' 화면에 나오는 모듈 오브젝트를 변경할 수 있습니다. F14(모듈 리스트에 대한 작업) 키를 눌러 '모듈 소스 표시' 화면에서 '모듈 리스트에 대한 작업' 화면에 액세스할 수 있습니다. '모듈 소스 표시' 화면은 428 페이지의 그림 21에 있습니다.

모듈 오브젝트를 선택하려면 표시하려는 모듈 오브젝트 옆에 5(모듈 오브젝트 표시)를 입력하십시오.

```
                          모듈 소스 표시

프로그램:   DSPWKDAY      라이브러리:   MYLIB       모듈:   DSPWKDAY
    24      500-             CALL       PGM(WEEKDAY2) PARM(&DAYOFWK)
    25      600-             IF         COND(&DAYOFWK *EQ 1) THEN(CHGVAR +
    26      700                           VAR(&WEEKDAY) VALUE('Sunday'))
    27      800-             ELSE       CMD(IF COND(&DAYOFWK *EQ 2) THEN(CHGV
    28      900                           VAR(&WEEKDAY) VALUE('Monday')))
    29     1000-             ELSE       CMD(IF COND(&DAYOFWK *EQ 3) THEN(CHGV
    30     1100                          VAR(&WEEKDAY) VALUE('Tuesday')))
    31     1200-             ELSE       CMD(IF COND(&DAYOFWK *EQ 4) THEN(CHGV
    32     1300                          VAR(&WEEKDAY) VALUE('Wednesday')))
    33     1400-             ELSE       CMD(IF COND(&DAYOFWK *EQ 5) THEN(CHGV
    34     1500                          VAR(&WEEKDAY) VALUE('Thursday')))
    35     1600-             ELSE       CMD(IF COND(&DAYOFWK *EQ 6) THEN(CHGV
    36     1700                          VAR(&WEEKDAY) VALUE('Friday')))
    37     1800-             ELSE       CMD(IF COND(&DAYOFWK *EQ 7) THEN(CHGV
    38     1900                          VAR(&WEEKDAY) VALUE('Saturday')))
                                                                    계속...
디버그. . .

F3=프로그램 종료   F6=중단점 추가/지우기   F10=단계화   F11=변수 표시
F12=재개   F17=감시 변수   F18=감시 변수에 대한 작업   F24=추가 키
```

그림 21. 모듈 보기 표시

보려는 모듈 오브젝트를 선택한 후 Enter 키를 누르십시오. 선택된 모듈 오브젝트가 '모듈 소스 표시' 화면에 표시됩니다.

모듈 오브젝트를 변경하는 또 다른 방법은 DISPLAY 디버그 명령을 사용하는 것입니다. 디버그 명령 행에서 다음을 입력하십시오.

```
DISPLAY MODULE module-name
```

그러면, 모듈 오브젝트 *module-name*이 표시됩니다. 모듈 오브젝트는 디버그 세션에 추가되었던 프로그램이나 서비스 프로그램 오브젝트에 존재해야 합니다.

모듈 오브젝트의 보기 변경

사용자가 ILE CL 모듈 오브젝트를 작성할 때 지정한 값에 따라 ILE CL 모듈 오브젝트의 몇 가지 보기가 사용될 수 있습니다. 이들 보기는 다음과 같습니다.

- 루트 소스 보기
- 리스팅 보기
- 명령문 보기

'보기 선택' 화면을 사용하여 '모듈 소스 표시' 화면에서 표시된 모듈 오브젝트의 보기를 변경할 수 있습니다. F15(보기 선택) 키를 눌러 '모듈 소스 표시' 화면에서 '보기 선택' 화면에 액세스할 수 있습니다. '보기 선택' 화면은 429 페이지의 그림 22에 있습니다. 현재 보기가 창의 맨 위에 나열되고 사용 가능한 다른 보기는 아래에 표시됩니다. 프로그램 오브젝트 안의 각 모듈 오브젝트는 그것을 작성하는 데 사용된 디버그 옵션에 따라 사용 가능한 상이한 보기 세트를 가질 수 있습니다.

보기를 선택하려면 표시하려는 보기 옆에 1(선택)을 입력하십시오.

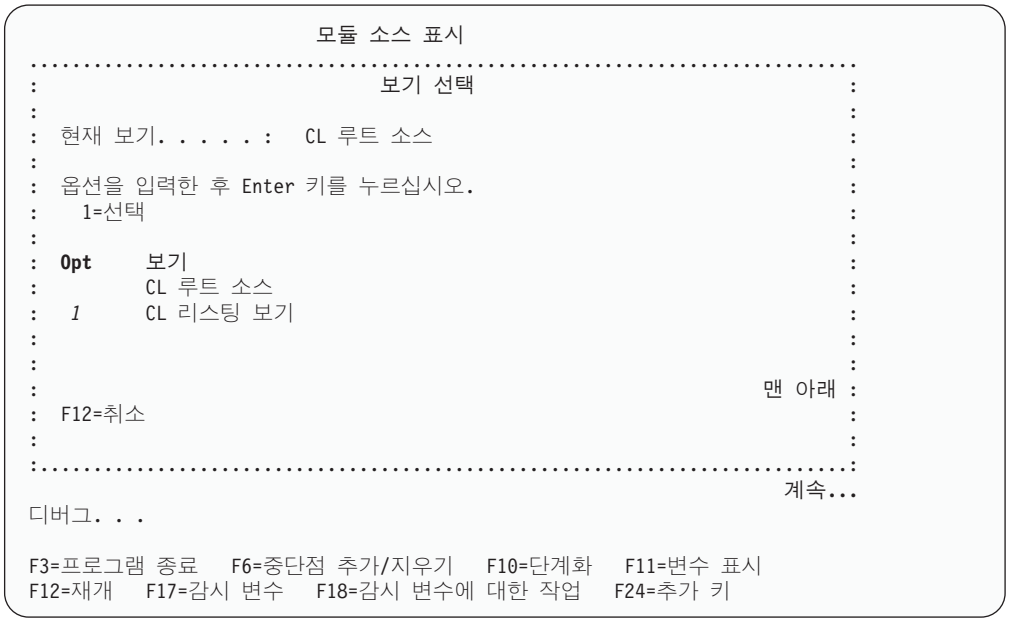


그림 22. 모듈 오브젝트의 보기 변경

보려는 모듈 오브젝트의 보기를 선택한 후 Enter 키를 누르면 선택한 모듈 오브젝트의 보기가 '모듈 소스 표시' 화면에 나옵니다.

중단점 설정 및 제거

프로그램 오브젝트가 실행 중일 때 중단점을 사용하여 특정 지점에서 프로그램 오브젝트를 중단할 수 있습니다. **무조건부 중단점**은 특정 명령문에서 프로그램 오브젝트를 중단합니다. **조건부 중단점**은 특정 명령문의 특정 조건이 충족될 때 프로그램 오브젝트를 중단합니다.

프로그램 오브젝트가 중단되면 '모듈 소스 표시' 화면이 표시됩니다. 중단점이 발생한 행의 소스와 함께 해당 모듈 오브젝트가 표시됩니다. 이 행은 강조표시됩니다. 이 지점에서 변수 평가, 추가 중단점 설정 및 디버그 명령을 수행할 수 있습니다.

중단점을 사용하기 전에 중단점에 대한 다음 특성들을 알아야 합니다.

- 예를 들어, GOTO문의 경우 중단점이 바이패스되면 그 중단점은 처리되지 않습니다.
- 중단점이 명령문에 설정되면 그 명령문이 처리되기 전에 중단점이 발생합니다.
- 조건부 중단점이 있는 명령문에 도달하면 명령문이 처리되기 전에 중단점과 연관된 조건식 이 계산됩니다.
- 중단점 기능은 디버그 명령을 통해 지정됩니다.

  그 기능들은 다음과 같습니다.

  – 프로그램 오브젝트에 중단점 추가

  – 프로그램 오브젝트에서 중단점 제거

– 중단점 정보 표시
– 중단점에 도달한 후 프로그램 오브젝트의 실행 재개

무조건부 중단점 설정 및 제거

다음을 사용하여 무조건부 중단점을 설정하거나 제거할 수 있습니다.

- '모듈 소스 표시' 화면에서 F6(중단점 추가/지우기) 키
- '모듈 소스 표시' 화면에서 F13(모듈 중단점에 대한 작업) 키
- 중단점을 설정하는 BREAK 디버그 명령
- 중단점을 제거하는 CLEAR 디버그 명령

무조건부 중단점을 설정하고 제거하는 가장 간단한 방법은 '모듈 소스 표시' 화면에서 F6(중단점 추가/지우기) 키를 사용하는 것입니다. F6 키를 사용하여 무조건부 중단점을 설정하려면 중단점을 추가하려는 행에 커서를 놓고 F6 키를 누르십시오. 그러면 무조건부 중단점이 그 행에 설정됩니다. 무조건부 중단점을 제거하려면 중단점을 제거하려는 행에 커서를 놓고 F6 키를 누르십시오. 그러면 중단점이 그 행에서 제거됩니다.

설정하려는 각각의 무조건부 중단점에 대해 이전 단계들을 반복하십시오.

주: 중단점을 설정하려는 행이 실행 가능한 명령문이 아닌 경우에는 실행 가능한 다음 명령문에 중단점이 설정됩니다.

중단점이 설정된 후에는 F3(나감) 키를 눌러 '모듈 소스 표시' 화면에서 나가십시오. '모듈 소스 표시' 화면에서 F21(명령 행) 키를 사용하여 명령 행에서 프로그램을 호출할 수도 있습니다.

프로그램 오브젝트를 호출하십시오. 중단점에 도달하면 프로그램이 중단되고 '모듈 소스 표시' 화면이 다시 표시됩니다. 이 지점에서 변수 평가, 추가 중단점 설정 및 디버그 명령을 수행할 수 있습니다.

무조건부 중단점을 설정하고 제거하는 또 다른 방법은 BREAK와 CLEAR 디버그 명령을 사용하는 것입니다.

BREAK 디버그 명령을 사용하여 무조건부 중단점을 설정하려면 디버그 명령 행에

```
BREAK line-number
```

를 입력하십시오. *Line-number*는 중단점을 설정하려는 현재 표시된 모듈 오브젝트 보기의 행 번호입니다.

CLEAR 디버그 명령을 사용하여 무조건부 중단점을 제거하려면 디버그 명령 행에

```
CLEAR line-number
```

를 입력하십시오. *Line-number*는 중단점을 제거하려는 현재 표시된 모듈 오브젝트의 보기의 행 번호입니다.

명령문 보기를 사용하는 경우에는 어떤 행 번호도 표시되지 않습니다. 명령문 보기에 무조건부 중단점을 설정하려면 디버그 명령 행에

```
BREAK procedure-name/statement-number
```

를 입력하십시오. *Procedure-name*은 CL 모듈명입니다. *Statement-number*는 (컴파일러 리스팅에서) 중단시키려는 명령문의 갯수입니다.

조건부 중단점 설정 및 제거

다음을 사용하여 조건부 중단점을 설정하거나 제거할 수 있습니다.

- '중단점에 대한 작업' 화면
- 중단점을 설정하는 BREAK 디버그 명령
- 중단점을 제거하는 CLEAR 디버그 명령

'중단점에 대한 작업' 화면 사용:

주: 조건부 중단점에 지원되는 관계 연산자는 <, >, =, <=, >= 및 <>(같지 않음)입니다.

조건부 중단점을 설정하거나 제거할 수 있는 한 가지 방법은 '모듈 중단점에 대한 작업' 화면을 사용하는 것입니다. F13(모듈 중단점에 대한 작업) 키를 눌러 '모듈 소스 표시' 화면에서 '모듈 중단점에 대한 작업' 화면에 액세스할 수 있습니다. '모듈 중단점에 대한 작업' 화면은 432 페이지의 그림 23에 있습니다. 조건부 중단점을 설정하려면 경우 다음과 같이 입력하십시오.

- 1(추가)을 *Opt* 필드에
- 중단점을 설정하려는 디버거 행 번호를 *행* 필드에
- 조건식을 *조건* 필드에

그리고나서 Enter 키를 누르십시오. 예를 들어, 432 페이지의 그림 23에 나오는 것처럼 디버거 행 35에 조건부 중단점을 설정하려면 다음과 같이 입력하십시오.

- 1(추가)을 *Opt* 필드에
- 35를 *행* 필드에
- &I=21을 *조건* 필드에

그리고나서 Enter 키를 누르십시오.

조건부 중단점을 제거하려면 제거하려는 중단점 옆의 *옵션(opt)* 필드에 4(지우기)를 입력한 후 Enter 키를 누르십시오. 이러한 방법으로 무조건부 중단점도 제거할 수 있습니다.

```
                        모듈 중단점에 대한 작업

시스템:   SYSTEM01
 프로그램 . . . :   MYPGM               라이브러리 . . :   MYLIB
   모듈 . . . . :     MYMOD             유형 . . . . . :   *PGM

옵션을 입력한 후 Enter 키를 누르십시오.
   1=추가  4=지우기

 Opt     행         조건
  1      35____     &I=21_____________________
  _      ______     __________________________
```

그림 23. 조건부 중단점 설정

설정하거나 제거하려는 각 조건부 중단점에 대해 이전 단계들을 반복하십시오.

주: 중단점을 설정하려는 행이 실행 가능한 명령문이 아닌 경우에는 실행 가능한 다음 명령문에 중단점이 설정됩니다.

설정하거나 제거하려는 모든 중단점을 지정한 후에는 F3(나감) 키를 눌러 '모듈 소스 표시' 화면으로 리턴하십시오.

그런 후 F3(나감) 키를 눌러 '모듈 소스 표시' 화면에서 나가십시오. '모듈 소스 표시' 화면에서 F21(명령 행) 키를 눌러 명령 행에서 프로그램 오브젝트를 호출할 수 있습니다.

프로그램 오브젝트를 호출하십시오. 조건부 중단점이 있는 명령문에 도달하면 명령문이 수행되기 전에 평가된 중단점과 연관된 조건식이 평가됩니다. 결과가 거짓이면, 프로그램 오브젝트는 계속 수행됩니다. 결과가 참이면, 프로그램 오브젝트는 중단되고 '모듈 소스 표시' 화면이 표시됩니다. 이 지점에서 변수 평가, 추가 중단점 설정 및 디버그 명령을 수행할 수 있습니다.

BREAK 및 CLEAR 디버그 명령 사용: 조건부 중단점을 설정하고 제거하는 또 다른 방법은 BREAK와 CLEAR 디버그 명령을 사용하는 것입니다.

BREAK 디버그 명령을 사용하여 조건부 중단점을 설정하려면 디버그 명령 행에

```
BREAK line-number WHEN expression
```

를 입력하십시오. *Line-number*는 중단점을 설정하려는 현재 표시된 모듈 오브젝트 보기의 행 번호입니다. *expression*은 중단점을 만났을 때 평가되는 조건식입니다. 조건부 중단점에 지원되는 관계 연산자는 <, >, =, <=, >= 및 <>(같지 않음)입니다.

비숫자 조건부 중단점 표현식에서 더 짧은 표현식은 비교되기 전에 내재적으로 공백으로 채워집니다. 내재적 채움은 자국어 정렬 순서(NLSS) 변환 전에 발생합니다. NLSS에 대해 자세히 알려면 433 페이지의 『자국어 정렬 순서(NLSS)』를 참조하십시오.

CLEAR 디버그 명령을 사용하여 조건부 중단점을 제거하려면 디버그 명령 행에

```
CLEAR line-number
```

를 입력하십시오. *Line-number*는 중단점을 제거하려는 현재 표시된 모듈 오브젝트 보기의 행 번호입니다.

명령문 보기에서는 행 번호가 표시되지 않습니다. 명령문 보기에 조건부 중단점을 설정하려면 디버그 명령 행에

```
BREAK procedure-name/statement-name WHEN expression
```

를 입력하십시오. *Procedure-name*은 CL 모듈명입니다. *Statement-number*는 (컴파일러 리스팅에서) 중단시키려는 명령문의 갯수입니다.

자국어 정렬 순서(NLSS): 비숫자 조건부 중단점 표현식은 다음 두 가지 유형으로 구분됩니다.

- Char-8: 각 문자가 8비트입니다.
- Char-16: 각 문자가 16비트입니다(DBCS).

NLSS는 Char-8이라는 비숫자 조건부 중단점 표현식에만 적용됩니다. 비숫자 조건부 중단점 표현식의 가능한 조합에 대해서는 434 페이지의 표 11을 참조하십시오.

Char-8 유형의 표현식에 대해 소스 디버거가 사용하는 정렬 순서 표는 CRTCLMOD 또는 CRTBNDCL 명령에서 SRTSEQ 매개변수용으로 지정된 정렬 순서 표입니다.

해결된 정렬 순서 표가 *HEX인 경우 정렬 순서 표는 사용되지 않습니다. 따라서 소스 디버거는 16진 문자 값을 사용하여 정렬 순서를 결정합니다. 그렇지 않으면 비교하기 전에 지정된 정렬 순서 표를 사용하여 각 바이트에 가중치를 할당합니다. SO/SI 문자들 사이 및 이 문자들을 포함한 바이트에는 가중치가 할당되지 **않습니다**.

주: 정렬 순서 표 이름은 컴파일 시 저장됩니다. 디버그시, 소스 디버거는 컴파일로부터 저장된 이름을 사용하여 정렬 순서 표에 액세스합니다. 컴파일 시에 지정된 정렬 순서 표가 *HEX 또는 *JOBRUN 이외의 어떤 것에 대해 해결하는 경우 디버깅이 시작되기 전에 정렬 순서 표를 변경하지 않는 것은 중요합니다. 표가 손상되었거나 삭제되어 표에 액세스할 수 없는 경우 소스 디버거는 *HEX 정렬 순서 표를 사용합니다.

표 11. 비숫자 조건부 중단점 표현식

유형	가능성
Char-8	• 문자 변수 대 문자 변수 비교 • 문자 변수 대 문자 리터럴 비교 [1] • 문자 변수 대 16진 리터럴 비교 [2] • 문자 리터럴 [1] 대 문자 변수 비교 • 문자 리터럴 [1] 대 문자 리터럴 [1] 비교 • 문자 리터럴 [1] 대 16진 리터럴 [2] 비교 • 16진 리터럴 [2] 대 문자 변수 [1] 비교 • 16진 리터럴 [2] 대 문자 리터럴 [1] 비교 • 16진 리터럴 [2] 대 16진 리터럴 [2] 비교
Char 16	• DBCS 문자 변수 대 DBCS 문자 변수 비교 • DBCS 문자 변수 대 그래픽 리터럴 [3] 비교 • DBCS 문자 변수 대 16진 리터럴 [2] 비교 • 그래픽 리터럴 [3] 대 DBCS 문자 변수 비교 • 그래픽 리터럴 [3] 대 그래픽 리터럴 [3] 비교 • 그래픽 리터럴 [3] 대 16진 리터럴 [2] 비교 • 16진 리터럴 [2] 대 DBCS 문자 변수 비교 • 16진 리터럴 [2] 대 그래픽 리터럴 [3] 비교

:

1 'abc' 양식의 문자 리터럴.

2 X'16진수' 양식의 16진 리터럴.

3 그래픽 리터럴은 G'<so>DBCS 자료<si>' 양식으로 되어 있습니다. SO는 <so>로 표시하고 SI는 <si>로 표시합니다.

조건부 중단점의 예:

```
CL 선언:          DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(1)
                  DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(2)
                  DCL    VAR(&DEC1) TYPE(*DEC) LEN(3 1)
                  DCL    VAR(&DEC2) TYPE(*DEC) LEN(4 1)

디버그 명령:      BREAK 31 WHEN &DEC1 = 48.1

디버그 명령:      BREAK 31 WHEN &DEC2 > &DEC1

디버그 명령:      BREAK 31 WHEN &CHAR2 <> 'A'

주석:             'A'는 비교되기 전에 하나의
                  공백 문자와 함께 오른쪽에 내재적으로
                  채워집니다.

디버그 명령:      BREAK 31 WHEN %SUBSTR(&CHAR2 2 1) <= X'F1'
```

```
디버그 명령:       BREAK 31 WHEN %SUBSTR(&CHAR2 1 1) >= &CHAR1
디버그 명령:       BREAK 31 WHEN %SUBSTR(&CHAR2 1 1) < %SUBSTR(&CHAR2 2 1)
```

%SUBSTR 내장 기능을 사용하여 문자 스트링 변수를 서브스트링화할 수 있습니다. 첫 번째 인수는 스트링 ID, 두 번째 인수는 시작 위치 그리고 세 번째 인수는 1바이트나 2바이트 문자로 된 숫자여야 합니다. 인수는 하나 이상의 빈 칸을 사용하여 분리할 수 있습니다.

모든 중단점 제거

CLEAR PGM 디버그 명령을 사용하여 '모듈 소스 표시' 화면에서 표시된 모듈 오브젝트가 있는 프로그램 오브젝트에서 조건부 및 무조건부 중단점을 모두 제거할 수 있습니다. 디버그 명령을 사용하려면 디버그 명령 행에

```
CLEAR PGM
```

를 입력하십시오. 프로그램이나 서비스 프로그램의 바인드된 모든 모듈에서 중단점이 제거됩니다.

프로그램 오브젝트의 단계화

중단점을 만난 후 프로그램 오브젝트에 지정된 갯수의 명령문을 실행 시킨 다음, 프로그램을 다시 중단하고 '모듈 소스 표시' 화면으로 리턴할 수 있습니다. 프로그램 오브젝트는 프로그램이 중단된 모듈 오브젝트의 다음 번 명령문에서 실행을 시작합니다. 일반적으로, 중단점은 프로그램 오브젝트를 중단시키는 데 사용됩니다.

다음을 사용하여 프로그램 오브젝트를 단계화할 수 있습니다.

- '모듈 소스 표시' 화면에서 F10(단계화) 키 또는 F22(내부 단계화) 키
- STEP 디버그 명령

'소스 표시' 화면에서 F10 키 또는 F22 키 사용

한번에 한 명령문씩 프로그램 오브젝트를 단계화하는 가장 간단한 방법은 '모듈 소스 표시' 화면에서 F10(단계화) 또는 F22(내부 단계화) 키를 사용하는 것입니다. F10(단계화) 또는 F22(내부 단계화) 키를 누르면, '모듈 소스 표시' 화면에서 표시된 모듈 오브젝트의 다음 번 명령문이 실행되고 프로그램 오브젝트가 다시 중단됩니다.

주: F10(단계화) 또는 F22(내부 단계화) 키를 사용할 경우 단계화할 명령문 갯수를 지정할 수 없습니다. F10(단계화) 또는 F22(내부 단계화) 키를 누르면 하나의 단계가 수행됩니다.

프로그램 오브젝트를 단계화하는 또 다른 방법은 STEP 디버그 명령을 사용하는 것입니다. STEP 디버그 명령으로 사용자는 한 단계에서 둘 이상의 명령문을 실행할 수 있습니다.

STEP 디버그 명령 사용

STEP 디버그 명령을 사용하여 실행할 명령문의 디폴트 숫자는 1입니다. STEP 디버그 명령을 사용하여 프로그램 오브젝트를 단계화하려면 디버그 명령 행에

```
STEP number-of-statements
```

를 입력하십시오. *Number-of-statements*는 프로그램 오브젝트가 다시 중단되기 전에 다음 단계에서 사용자가 실행할 프로그램 오브젝트의 명령문 수입니다. 예를 들어, 디버그 명령 행에 다음을 입력한 경우:

```
STEP 5
```

이 경우 프로그램 오브젝트의 다음 5개의 명령문이 수행된 후 프로그램 오브젝트가 다시 중단되고 '모듈 소스 표시' 화면이 표시됩니다.

외부 단계화(step over)와 내부 단계화(step into)

다른 프로그램 오브젝트에 대한 CALL문을 디버그 세션에서 만날 경우 다음 중 하나를 수행할 수 있습니다.

- 호출된 프로그램 오브젝트의 외부 단계화
- 호출된 프로그램 오브젝트의 내부 단계화

호출된 프로그램 오브젝트의 **외부 단계화**를 선택하면 CALL문과 호출된 프로그램 오브젝트가 하나의 단계로 수행됩니다. 따라서 호출 프로그램 오브젝트가 다음 단계에서 중단되기 전에 호출된 프로그램 오브젝트가 실행되어 완료됩니다. 외부 단계화는 디폴트 단계 모드입니다.

호출된 프로그램 오브젝트의 **외부 단계화**를 선택하면 호출된 프로그램 오브젝트의 각 명령문이 한 단계로 수행됩니다. 실행 중인 프로그램 오브젝트가 중단될 다음 단계가 호출된 프로그램 오브젝트 내에 있는 경우 호출된 프로그램 오브젝트는 이 지점에서 중단됩니다. 호출된 프로그램 오브젝트가 디버그 자료로 컴파일되었으며 사용자에게 디버그에 대한 올바른 권한이 있는 경우에는 '모듈 소스 표시' 화면에 호출된 프로그램 오브젝트가 표시됩니다.

프로그램 오브젝트의 외부 단계화

다음을 사용하여 프로그램 오브젝트의 외부 단계화를 수행할 수 있습니다.

- '모듈 소스 표시' 화면에서 F10(단계화) 키
- STEP OVER 디버그 명령

F10(단계화) 키 사용

'모듈 소스 표시' 화면에서 F10(단계화) 키를 사용하여 디버그 세션에서 호출된 프로그램 오브젝트를 외부로 단계화할 수 있습니다. 수행할 다음 명령문이 다른 프로그램 오브젝트에 대한 CALL문인 경우 F10(단계화) 키를 누르면 호출 프로그램 오브젝트가 다시 중단되기 전에 호출된 프로그램 오브젝트가 실행되어 완료됩니다.

외부 단계화(step over) 디버그 명령 사용

다른 방법으로는 STEP OVER 디버그 명령을 사용하여 디버그 세션에서 호출된 프로그램 오브젝트의 외부 단계화를 수행할 수 있습니다. STEP OVER 디버그 명령을 사용하려면 디버그 명령 행에

```
STEP number-of-statements OVER
```

를 입력하십시오. *Number-of-statements*는 프로그램 오브젝트가 다시 정지되기 전에 다음 단계에서 사용자가 실행시키려는 프로그램 오브젝트의 명령문 갯수입니다. 실행되는 명령문 중 하나에 다른 프로그램 오브젝트에 대한 CALL문이 있는 경우 ILE 소스 디버거가 호출된 프로그램 오브젝트의 외부 단계화를 수행할 수 있습니다.

프로그램 오브젝트의 내부 단계화

다음을 사용하여 프로그램 오브젝트의 내부 단계화(step into)를 수행할 수 있습니다.

- '모듈 소스 표시' 화면에서 F22(내부 단계화) 키
- STEP INTO 디버그 명령

F22(내부 단계화) 키 사용

'모듈 소스 표시' 화면에서 F22(내부 단계화) 키를 사용하여 디버그 세션에서 호출된 프로그램 오브젝트의 내부로 단계화를 수행할 수 있습니다. 수행할 다음 명령문이 다른 프로그램 오브젝트에 대한 CALL문인 경우 F22(내부 단계화) 키를 누르면 호출된 프로그램 오브젝트의 첫 번째 명령문이 수행됩니다. 그러면 호출된 프로그램 오브젝트가 '모듈 소스 표시' 화면에서 표시됩니다.

주: 호출된 프로그램 오브젝트를 '모듈 소스 표시' 화면에서 표시하려면 그것과 연관된 디버그 자료가 있어야 합니다.

내부 단계(step into)화 디버그 명령 사용

다른 방법으로는 STEP INTO 디버그 명령을 사용하여 디버그 세션에서 호출된 프로그램 오브젝트의 내부 단계화를 수행할 수 있습니다. STEP INTO 디버그 명령을 사용하려면 디버그 명령 행에

```
STEP number-of-statements INTO
```

를 입력하십시오. *Number-of-statements*는 프로그램 오브젝트가 다시 중단되기 전에 다음 단계에서 사용자가 실행할 프로그램 오브젝트의 명령문 수입니다. 실행되는 명령문 중 하나에 다른 프로그램 오브젝트에 대한 CALL문이 있는 경우 디버거가 호출된 프로그램 오브젝트의 내부 단계화를 수행합니다. 호출된 프로그램 오브젝트의 각 명령문은 이 단계에서 계산됩니다. 이 단계가 호출된 프로그램 오브젝트에서 종료하면 호출된 프로그램 오브젝트가 '모듈 소스 표시' 화면에서 표시됩니다. 예를 들어, 디버그 명령 행에 다음을 입력한 경우:

```
STEP 5 INTO
```

프로그램 오브젝트에 있는 다음 5개의 명령문이 실행됩니다. 세 번째 명령문이 다른 프로그램 오브젝트에 대한 CALL문인 경우 호출 프로그램 오브젝트에 있는 두 개의 명령문이 수행되고 호출된 프로그램 오브젝트에 있는 첫 번째 세 개의 명령문이 수행됩니다.

변수 표시

다음을 사용하여 변수 값을 표시할 수 있습니다.

- '모듈 소스 표시' 화면에서 F11(변수 표시) 키
- EVAL 디버그 명령

EVAL 명령에서 사용되는 변수 범위는 QUAL 명령을 사용하여 정의됩니다. 그러나 CL 모듈에 들어 있는 변수는 모두 전영역 범위이므로 변수의 범위를 특별히 정의할 필요는 없습니다.

F11(변수 표시) 키 사용

F11(변수 표시) 키를 사용하여 변수를 표시하려면 커서를 변수 위에 놓고 F11 키를 누르십시오. 변수의 현재 값이 '모듈 소스 표시' 화면의 맨 아래에 있는 메세지 행에 표시됩니다.

```
                          모듈 소스 표시

프로그램:   DSPWKDAY      라이브러리:   MYLIB       모듈:   DSPWKDAY
     4              DCL        VAR(&MSGTEXT) TYPE(*CHAR) LEN(20)
     5              CALL       PGM(WEEKDAY2) PARM(&DAYOFWK)
     6              IF         COND(&DAYOFWK *EQ 1) THEN(CHGVAR +
     7                           VAR(&WEEKDAY) VALUE('Sunday'))
     8              ELSE       CMD(IF COND(&DAYOFWK *EQ 2) THEN(CHGVAR +
     9                           VAR(&WEEKDAY) VALUE('Monday')))
    10              ELSE       CMD(IF COND(&DAYOFWK *EQ 3) THEN(CHGVAR +
    11                           VAR(&WEEKDAY) VALUE('Tuesday')))
    12              ELSE       CMD(IF COND(&DAYOFWK *EQ 4) THEN(CHGVAR +
    13                           VAR(&WEEKDAY) VALUE('Wednesday')))
    14              ELSE       CMD(IF COND(&DAYOFWK *EQ 5) THEN(CHGVAR +
    15                           VAR(&WEEKDAY) VALUE('Thursday')))
    16              ELSE       CMD(IF COND(&DAYOFWK *EQ 6) THEN(CHGVAR +
    17                           VAR(&WEEKDAY) VALUE('Friday')))
    18              ELSE       CMD(IF COND(&DAYOFWK *EQ 7) THEN(CHGVAR +
                                                                  계속...
디버그. . .

F3=프로그램 종료 F6=중단점 추가/지우기     F10=단계화   F11=변수 표시
F12=재개    F17=감시 변수    F18=감시 변수에 대한 작업    F24=추가 키
&DAYOFWK = 3.
```

그림 24. F11(변수 표시) 키를 사용하여 변수 표시. EVAL 디버그 명령 사용

EVAL 디버그 명령을 사용하여 변수 값을 결정할 수도 있습니다. EVAL 디버그 명령을 사용하여 변수 값을 표시하려면 디버그 명령 행에

```
EVAL variable-name
```

를 입력하십시오. *Variable-name*은 표시하려는 변수명입니다. '모듈 소스 표시' 화면에서 EAVL 디버그 명령을 입력하고 값이 하나의 행에 표시될 수 있으면 변수 값이 메세지 행에 표시됩니다. 값이 하나의 행에 표시될 수 없으면 '표현식 평가' 화면에 표시됩니다.

예를 들어, 438 페이지의 그림 24에 나오는 모듈 오브젝트의 7행에 변수 *&DAYOFWK*; 의 값을 표시하려면 다음과 같이 입력하십시오.

```
EVAL &DAYOFWK
```

'모듈 소스 표시' 화면의 메세지 행에서 438 페이지의 그림 24처럼 &DAYOFWK = 3.이 표시됩니다.

논리 변수 표시 예

```
CL 선언:          DCL    VAR(&LGL1) TYPE(*LGL) VALUE('1')

디버그 명령:      EVAL &LGL1
결과:             &LGL1 = '1'
```

문자 변수 표시 예

```
CL 선언:    DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(10) VALUE('EXAMPLE')
디버그 명령:      EVAL &CHAR1
결과:             &CHAR1 = 'EXAMPLE   '

디버그 명령:      EVAL %SUBSTR(&CHAR1 5 3)
결과:             %SUBSTR(&CHAR1 5 3) = 'PLE'

디버그 명령:      EVAL %SUBSTR(&CHAR1 7 4)
결과:             %SUBSTR(&CHAR1 7 4) = 'E   '
```

%SUBSTR 내장 기능을 사용하여 문자 스트링 변수를 서브스트링화할 수 있습니다. 첫 번째 인수는 스트링 ID, 두 번째 인수는 시작 위치 그리고 세 번째 인수는 1바이트나 2바이트 문자로 된 숫자여야 합니다. 인수는 하나 이상의 빈 칸을 사용하여 분리할 수 있습니다.

10진 변수 표시 예

```
CL 선언:    DCL    VAR(&DEC1) TYPE(*DEC) LEN(4 1) VALUE(73.1)
CL 선언:    DCL    VAR(&DEC2) TYPE(*DEC) LEN(3 1) VALUE(12.5)
디버그 명령:      EVAL &DEC1
```

```
결과:              &DEC1 = 073.1

디버그 명령:       EVAL &DEC2

결과:              &DEC2 = 12.5
```

16진값으로 변수 표시

EVAL 디버그 명령을 사용하여 변수 값을 16진 형식으로 표시할 수 있습니다. 16진 형식으로 변수를 표시하려면 디버그 명령 행에 다음을 입력하십시오.

```
EVAL variable-name: x  number-of-bytes
```

*Variable-name*은 16진 형식으로 표시하려는 변수명입니다. 'x'는 변수가 16진 형식으로 표시되도록 지정하고 *number-of-bytes*는 표시될 바이트 수를 나타냅니다. 'x' 다음에 길이가 지정되지 않으면 변수 크기가 길이로 사용됩니다. 최소한 16바이트는 항상 표시됩니다. 변수의 길이가 16바이트 미만이면, 나머지 공간은 16바이트 경계에 도달할 때까지 0으로 채워집니다.

```
CL 선언:         DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(10) VALUE('ABC')
                 DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(10) VALUE('DEF')

디버그 명령:     EVAL &CHAR1:X 32

결과:
00000     C1C2C340 40404040 4040C4C5 C6404040 ABC       DEF
00010     40404040 00000000 00000000 00000000     ............
```

변수 값 변경

할당된 연산자(=)를 EVAL 명령과 함께 사용하여 변수 값을 변경할 수 있습니다.

EVAL 명령에서 사용되는 변수 범위는 QUAL 명령을 사용하여 정의됩니다. 그러나 CL 모듈에 들어 있는 변수는 모두 전영역 범위(global scope)이므로 변수의 범위를 특별히 정의할 필요는 없습니다.

변수가 변수 정의에 일치하는 경우에는 EVAL 디버그 명령을 사용하여 숫자, 문자 및 16진 자료를 변수에 할당할 수 있습니다.

변수 값을 변경하려면 디버그 명령 행에

```
EVAL variable-name = value
```

를 입력하십시오. *Variable-name*은 변경하려는 변수명이며, *value*는 *variable-name*에 할당하려는 ID 또는 리터럴 값입니다. 예를 들면,

```
EVAL &COUNTER = 3.0
```

은 *&COUNTER;*의 값을 3.0으로 변경하고

```
&COUNTER = 3.0 = 3.0
```

을 '모듈 소스 표시' 화면의 메세지 행에 표시합니다. 결과 앞에 사용자가 변경 중인 변수명과 값이 나옵니다.

문자 변수에 값을 할당할 때에는 다음 규칙이 적용됩니다.

- 소스 표현식의 길이가 목표 표현식의 길이보다 짧으면, 자료가 목표 표현식에서 좌측 정렬되며 나머지 자리는 공백으로 채워집니다.
- 소스 표현식의 길이가 목표 표현식의 길이보다 길면, 자료가 목표 표현식에서 좌측 정렬되며 목표 표현식의 길이가 절단됩니다.

주: 다음 어느 것이든지 DBCS 변수를 할당할 수 있습니다.

- 다른 DBCS 변수
- G'<so>DBCS 자료<si>' 형식의 그래픽 리터럴
- X'16진수' 양식의 16진 리터럴

논리 변수 변경 예

```
CL 선언:          DCL    VAR(&LGL1) TYPE(*LGL) VALUE('1')
                  DCL    VAR(&LGL2) TYPE(*LGL)

디버그 명령:      EVAL &LGL1

결과:             &LGL1 = '1'

디버그 명령:      EVAL &LGL1 = X'F0'

결과:             &LGL1 = X'F0' = '0'

디버그 명령:      EVAL &LGL2 = &LGL1

결과:             &LGL2 = &LGL1 = '0'
```

문자 변수 변경 예

```
CL 선언:          DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(1) VALUE('A')
                  DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(10)

디버그 명령:      EVAL &CHAR1 = 'B'

결과:             &CHAR1 = 'B' = 'B'

디버그 명령:      EVAL &CHAR1 = X'F0F1F2F3'

결과:             &CHAR1 = 'F0F1F2F3' = '0'

디버그 명령:      EVAL &CHAR2 = 'ABC'

결과:             &CHAR2 = 'ABC' = 'ABC       '
```

```
디버그 명령:       EVAL %SUBSTR(CHAR2 1 2) = %SUBSTR(&CHAR2 3 1)

결과:              %SUBSTR(CHAR2 1 2) = %SUBSTR(&CHAR2 3 1) = 'C '

주석:              변수 &CHAR에는 'C C       '가 들어 있습니다.
```

%SUBSTR 내장 기능을 사용하여 문자 스트링 변수를 서브스트링화할 수 있습니다. 첫 번째 인수는 스트링 ID, 두 번째 인수는 시작 위치 그리고 세 번째 인수는 1바이트나 2바이트 문자로 된 숫자여야 합니다. 인수는 하나 이상의 빈 칸을 사용하여 분리할 수 있습니다.

10진 변수 변경 예

```
CL 선언:           DCL    VAR(&DEC1) TYPE(*DEC) LEN(3 1) VALUE(73.1)
                   DCL    VAR(&DEC2) TYPE(*DEC) LEN(2 1) VALUE(3.1)

디버그 명령:       EVAL &DEC1 = 12.3

결과:              &DEC1 = 12.3 = 12.3

디버그 명령:       EVAL &DEC1 = &DEC2

결과:              &DEC1 = &DEC2 = 03.1
```

변수 속성 예

속성(ATTR) 디버그 명령은 사용자가 변수의 속성을 표시할 수 있도록 합니다. 속성은 420 페이지의 표 10의 디버그 기호 표에 기록된 것과 같은 변수 *크기*(바이트)와 유형입니다.

다음은 ATTR 디버그 명령을 사용한 예입니다.

```
CL 선언:          DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(10)

디버그 명령:       ATTR &CHAR2

결과:              TYPE = FIXED LENGTH STRING, LENGTH = 10 BYTES

CL 선언:           DCL    VAR(&DEC) TYPE(*DEC) LEN(3 1)

디버그 명령:       ATTR &DEC

결과:              TYPE = PACKED(3,1), LENGTH = 2 BYTES
```

이름을 변수, 표현식 또는 명령과 같게 함

EQUATE 디버그 명령을 사용하여 이름을 단축하여 사용하도록 변수, 표현식 또는 디버그 명령과 같게 할 수 있습니다. 그런 다음, 그 이름을 단독 또는 다른 표현식 안에

서 사용할 수 있습니다. 그것을 다른 표현식 안에서 사용하는 경우 표현식이 평가 계산되기 전에 이름값이 결정됩니다. 이 이름들은 디버그 세션이 종료되거나 이름이 제거될 때까지 활동 상태로 있습니다.

이름을 변수, 표현식 또는 디버그 명령과 같게 하려면 디버그 명령 행에서

```
EQUATE shorthand-name definition
```

를 입력하십시오. *shorthand-name*은 변수, 표현식 또는 디버그 명령과 같게 하려는 이름이며, *definition*은 이름과 같도록 하는 변수, 표현식 또는 디버그 명령입니다.

예를 들어, *&COUNTER*라는 변수의 내용을 표시하는 단축명 *DC*를 정의하려면 디버그 명령행에

```
EQUATE DC EVAL &COUNTER
```

를 입력하십시오. 그러면, 디버그 명령행에 *DC*가 입력될 때마다, EVAL *&COUNTER* 명령이 수행됩니다.

EQUATE 명령에 입력 가능한 최대 문자 수는 144입니다. 정의가 제공되지 않고 이전 EQUATE 명령이 이름을 정의했다면, 이전 정의는 제거됩니다. 이름이 미리 정의되지 않았다면 오류 메세지가 표시됩니다.

디버그 세션 동안 EQUATE 디버그 명령으로 정의되었던 이름을 참조하려면 디버그 명령 행에

```
DISPLAY EQUATE
```

를 입력하십시오. 활동명 리스트가 '표현식 평가' 화면에 표시됩니다.

ILE CL에 대한 소스 디버그 자국어 지원

ILE CL에 대한 소스 디버그 자국어 지원에 대한 작업에는 다음 조건이 존재합니다.

- '모듈 소스 표시' 화면에 보기가 표시될 때 소스 디버거가 모든 자료를 디버그 작업의 코드화 문자 세트 ID(CCSID)로 변환합니다.
- 변수에 리터럴을 할당할 때 소스 디버거는 인용된 리터럴(예: 'abc')에 대한 CCSID 변환을 수행하지 않습니다. 또한 인용된 리터럴은 대소문자를 구분합니다.

*SOURCE 보기에 대한 작업

다음 조건은 사용자가 CL 루트 소스 보기에 대해 작업할 경우에만 해당됩니다.

- 소스 파일 CCSID가 모듈 CCSID와 **다를** 경우 소스 디버거는 변형 문자(#, @, $)가 들어 있는 CL 식별자를 인식하지 못할 수 있습니다.

DSPMOD(모듈 표시) CL 명령을 사용하여 모듈의 CCSID를 찾을 수 있습니다. CL 루트 소스 보기에 대해 작업해야 하고 소스 파일 CCSID가 모듈 CCSID와 다른 경우에는 다음 중 하나의 조치를 수행할 수 있습니다.

- CL 소스의 CCSID가 컴파일 시 작업의 CCSID와 같은지를 확인하십시오.
- 컴파일 시 작업의 CCSID를 65 535로 변경하고 컴파일하십시오.
- 앞의 두 개의 옵션을 사용할 수 없을 경우 CL 리스팅 보기를 사용하십시오.

ILE 개념  , 제 10 장 『디버깅 고려사항』을 참조하십시오.

디버깅 중 COPY, SAVE, RESTORE, CRTDUPOBJ, CHKOBJITG 사용

중단점이나 단계들은 디버깅 중에 라이브러리나 프로그램을 지정하기 위해 CL 명령을 사용할 때 프로그램으로부터 **일시적으로 제거**될 수 있습니다. CL 명령이 실행을 완료하게 되면 중단점 및 단계들이 다시 복원됩니다. 중단점이나 단계들이 제거될 때 작업 기록부에는 CPD190A 메세지가 놓이게 되고 중단점이나 단계들이 다시 복원되면 또 다른 CPD190A 메세지가 작업 기록부에 놓입니다.

다음은 중단점이나 단계들을 일시적으로 제거시키는 CL 명령들입니다.

CHKOBJITG	CPY	CPROBJ	RSTLIB	SAVLIB
	CPYLIB	CRTDUPOBJ	RSTOBJ	SAVOBJ
				SAVSYS
				SAVCHGOBJ

주: CL 명령이 프로그램에서 실행될 때 BREAK나 STEP 명령을 발행하면 CPF7102 오류 메세지를 수신합니다.

OPM 프로그램 디버깅

테스트 기능은 어플리케이션의 작성 및 유지보수에 도움을 주기 위해 설계된 것입니다. 테스트 기능을 통해 사용자들은 프로그램이 테스트 환경에서 어떻게 처리되는지를 관찰 및 제어하면서 동시에 특별한 테스트 환경에서 프로그램을 실행할 수 있습니다. 이 장에서 설명하는 테스트 기능을 통해 사용자들은 자신의 프로그램과 대화하게 됩니다. 이 기능들은 대화식 또는 일괄처리 작업에서 사용되는 명령 세트를 통해서도 사용이 가능합니다. 이 기능을 사용하여 실행할 수 있는 작업은 다음과 같습니다.

- 프로그램의 처리 순서를 추적하고, 순서에 따라 각 위치에서 처리된 명령문과 프로그램 변수의 값을 표시할 수 있습니다.
- 프로그램 안의 어떤 명령문에서나 중단하고(중단점이라고 함), 변수 값을 표시 또는 변경하거나 다른 사용자 정의 프로그램을 호출하는 등의 기능을 수행하기 위해 제어를 수신할 수 있습니다.

테스트되는 프로그램 안에 테스트를 위한 특별한 명령이 들어 있는 것은 아닙니다. 테스트되고 있는 같은 프로그램은 변경 없이 실행시킬 수 있습니다. 모든 테스트 명령들은 테스트되는 프로그램의 영구적인 부분으로서가 아닌, 프로그램이 테스트되는 작업 안

에 지정됩니다. 테스트 명령을 통해 사용자들은 고급 언어(HLL: High Level Language) 프로그램으로 작성된 것과 동일한 용어로 프로그램과 부호식으로 대화할 수 있습니다. 사용자는 그 이름으로 변수를 참조하고, 번호로 명령문을 참조합니다. (이 번호는 프로그램의 소스 리스트에서 사용된 번호입니다.) 또한 테스트 기능은 테스트 기능이 설정되어 있는 작업에만 적용할 수 있습니다. 같은 프로그램의 경우 설정된 테스트 기능에 영향을 받지 않고도 동시에 다른 작업에서 사용이 가능합니다.

디버그 모드

테스트를 시작하려면 프로그램이 디버그 모드에 있어야 합니다. 디버그 모드는 일반적인 시스템 기능에 추가하여 테스트 기능이 사용될 수 있는 특수 환경입니다. 테스트 기능은 디버그 모드에서만 사용될 수 있습니다. 디버그 모드를 시작하려면 STRDBG(디버그 시작) 명령을 반드시 사용해야 합니다. STRDBG 명령은 프로그램을 디버그 모드에 있게 하는 것은 물론, 디버그될 프로그램 등의 테스트 정보도 지정할 수 있게 합니다. 프로그램은 ENDDBG(디버그 종료) 명령 또는 RMVPGM(프로그램 제거) 명령을 발견하거나 사용 중인 현재 라우팅 단계를 마칠 때까지 디버그 모드 상태에 있습니다.

주: iSeries Navigator에서 시스템을 검사하기 위해 디버그를 선택할 때 시스템 디버그 관리자 기능을 사용한 경우 STRDBG(디버그 시작) 명령을 발행하면 그래픽 인터페이스가 나타납니다. 이 경우 사용자 리스트에 지정된 사용자 중 한 사람이 STRDBG 명령을 발행할 때마다 명령 입력 화면 대신 iSeries Navigator 시스템 디버그 관리자가 나타납니다. ENDDBG 명령을 입력한 후 시스템 디버그 관리자에서 *디버그*를 선택하지 않은 경우 STRDBG 명령을 발행하면 명령 입력 화면에서 시스템 디버거를 다시 호출합니다.

다음의 STRDBG 명령은 작업을 디버그 모드에 놓고, 디버그될 프로그램으로서 프로그램 CUS310을 추가합니다.

```
STRDBG  PGM(CUS310)
```

ILE 소스 디버거를 사용하여 OPM 프로그램을 디버그할 수 있습니다. 소스 디버그 자료가 포함된 OPM 프로그램을 작성하려면 CRTCLPGM(CL 프로그램 작성) 명령에 OPTION(*SRCDBG) 매개변수나 OPTION(*LSTDBG) 매개변수를 지정하십시오. 소스 디버그 자료는 실제로 프로그램 오브젝트의 일부입니다.

소스 디버그 자료가 들어 있는 OPM 프로그램을 작성하여 ILE 소스 디버거에 추가하려면 STRDBG 명령에 프로그램(PGM) 및 OPM 소스 레벨 디버그(OPMSRC) 매개변수를 사용하십시오. 소스 디버그 자료와 함께 작성된 OPM 프로그램으로 디버그 세션을 시작하려면 다음을 입력하십시오.

```
STRDBG PGM(*LIBL/MYOPMPGM) OPMSRC(*YES) DSPMODSRC(*YES)
```

ILE 소스 디버깅에 대한 추가 정보는 419 페이지의 『ILE 프로그램 디버깅』 부분을 참조하십시오.

디버그 모드에 프로그램 추가

어떤 프로그램이나 디버그 모드에서 수행될 수 있지만, 디버그를 위해서는 먼저 프로그램을 디버그 모드에 놓아야 합니다. 프로그램을 디버그 모드에 놓기 위해서는 STRDBG 명령의 PGM 매개변수에 프로그램을 지정하거나 ADDPGM(프로그램 추가) 명령을 사용하여 프로그램을 디버그 세션에 추가하면 됩니다. 한 작업에서 동시에 20개의 프로그램이 디버그되도록 지정할 수도 있습니다. 디버그 모드에 프로그램을 추가하려면 *CHANGE 권한이 있어야 합니다.

20개의 프로그램을 디버그 모드로 지정하고 나서(STRDBG나 ADDPGM 명령 또는 두 명령을 모두 사용하여) 더 많은 프로그램을 디버그 작업에 추가하려는 경우에는 이전에 지정했던 프로그램들 중 일부를 제거해야 합니다. 이 경우 RMVPGM(프로그램 제거) 명령을 사용하십시오. 디버그 모드가 종료되면 모든 프로그램이 자동으로 디버그 모드에서 제거됩니다.

디버그 모드를 시작할 때 한 프로그램을 디폴트 프로그램으로 지정할 수 있습니다. 디폴트 프로그램을 지정하면 명령이 사용될 때마다 프로그램명을 지정하지 않고도 PGM 매개변수를 가진 디버그 명령을 사용할 수 있습니다. 이는 하나의 프로그램만 디버그할 경우에 유용합니다. 예를 들어, ADDBKP(중단점 추가) 명령의 경우 중단점을 추가할 프로그램으로 디폴트 프로그램이 가정되기 때문에 PGM 매개변수에 프로그램명을 지정하지 않아도 됩니다. 디버그해야 할 프로그램 리스트(PGM 매개변수)에는 디폴트 프로그램명이 반드시 지정되어야 합니다. 리스트 안에 디버그할 프로그램이 둘 이상이면, 사용자가 DFTPGM 매개변수에 디폴트 프로그램을 지정할 수 있습니다. 이를 지정하지 않으면 STRDBG 명령의 PGM 매개변수에 지정된 리스트의 첫 번째 프로그램이 디폴트 프로그램으로 간주됩니다.

CHGDBG(디버그 변경) 또는 ADDPGM 명령을 사용하여 테스트하는 동안 디폴트 프로그램은 언제라도 변경이 가능합니다.

주: 디버그 모드 상태의 프로그램이 삭제 또는 재작성되거나 기억장치가 해제되어 저장된 경우 이 프로그램이 참조되면(RMVPGM 명령 제외), 기능 검사를 초래하게 됩니다. RMVPGM 명령을 사용하여 프로그램을 제거하거나 ENDDBG 명령을 사용하여 디버그 모드를 종료해야 합니다. 프로그램을 변경하고 나서 그것을 디버그하려면 프로그램을 디버그 모드에서 제거하여 재작성한 후 다시 디버그 모드로 추가해야 합니다(ADDPGM 명령 사용).

실행(production) 라이브러리의 데이터베이스 파일 갱신 방지

디버그 모드에 있는 동안에는 실행 라이브러리 안의 파일들을 사용할 수 있습니다. 실행 라이브러리 안의 데이터베이스 파일이 잘못 변경되지 않게 하려면 UPDPROD(*NO)를 지정하거나 STRDBG 명령의 디폴트로 *NO를 지정하십시오. 그러면 신규 레코드의 갱신 및 추가를 위해 테스트 라이브러리 안에 있는 파일만 열리게 됩니다. 신규 레코드를 갱신 또는 추가하기 위해 실행 라이브러리 안의 데이터베이스 파일을 열거나 실행 실제 파일로부터 멤버를 삭제하려면 UPDPROD(*YES)를 지정하십시오.

라이브러리 리스트에 대해서도 이 기능을 사용할 수 있습니다. 디버그 작업의 대상이 되는 라이브러리 리스트 안의 실행 라이브러리 앞에 테스트 라이브러리를 위치시킬 수 있습니다. 실행 파일은 테스트 라이브러리에서 디버그되는 프로그램에 의해 갱신될 수도 있으므로 사본을 보관하도록 하십시오. 그러면, 프로그램 수행 시, 테스트 라이브러리 안의 파일이 사용됩니다. 따라서 실행 파일이 잘못 갱신되는 것을 막을 수 있습니다.

호출 스택

DSPDBG(디버그 표시) 명령을 통해 사용자는 다음을 나타내는 호출 스택을 표시할 수 있습니다.

- 현재 디버그 중인 프로그램
- 호출 명령어의 명령어 번호 또는 프로그램 처리가 중단된 각 중단점의 명령어 번호
- 프로그램 순환 레벨(program recursion level)
- 디버그 모드에 있으나 호출되지 않았던 프로그램명

프로그램 호출은 프로그램에 *자동* 기억장치를 할당하고, 그 프로그램으로 기계 처리를 전송하는 것입니다. 일련의 호출이 호출 스택에 놓입니다. 프로그램이 처리를 종료하거나 제어를 전송하면 호출 스택에서 제거됩니다. 호출 스택에 대해 자세히 알려면 제 3 장을 참조하십시오.

프로그램의 첫 번째 호출이 호출 스택에 그대로 있는 동안에는 그 프로그램을 여러 번 호출할 수 있습니다. 프로그램을 호출하는 것 이 각각 그 프로그램에서 하나의 순환 레벨입니다.

호출이 종료되면(프로그램 리턴 또는 제어 전송) 자동 기억장치가 시스템으로 리턴됩니다.

주:

1. CL 프로그램은 순환될 수 있는 것으로서, CL 프로그램은 자신이 호출했던 프로그램을 사용하여 스스로를 직접 또는 간접으로 호출할 수 있습니다.
2. 고급 언어(HLL) 중 일부는 프로그램의 순환 호출을 허용하지 않습니다. 반면, 일부 다른 언어는 프로그램의 순환 호출뿐만 아니라, 프로그램 안의 프로시듀어까지 순환적으로 호출할 수 있습니다.(이 책에서 *순환 레벨*이란 프로그램이 호출 스택에서 호출되는 횟수를 가리킵니다. 프로시듀어의 순환 레벨은 명시적으로 프로시듀어 순환 레벨을 말합니다.)
3. 모든 CL 명령과 화면은 프로그램의 규정된 이름 순환 레벨만 사용합니다.

프로그램 활성화

프로그램을 활성화하는 것은 프로그램에 정적 기억장치를 할당하는 것입니다. 다음 중 하나가 발생할 경우 활성화가 종료됩니다.

- 현재 라우팅 단계가 종료된 경우.
- 프로그램을 활성화시킨 요구가 취소된 경우.
- RCLRSC(자원 재생) 명령이 수행되어 프로그램의 최종(또는 유일한) 호출이 종료된 경우.

이 외에도 활성화는 프로그램을 호출하는 중에 수행되는 조치에 의해서도 취소될 수 있습니다. 이러한 조치는 프로그램 작성 시 사용된 언어(HLL 또는 CL)에 따라 결정됩니다.

프로그램이 비활성화되면 정적 기억장치가 시스템으로 리턴됩니다. 작성된 프로그램 언어(HLL 또는 CL)에 따라 프로그램이 정상적으로 비활성화되는 시기가 결정됩니다. 프로그램이 종료되면 CL 프로그램은 항상 비활성화됩니다.

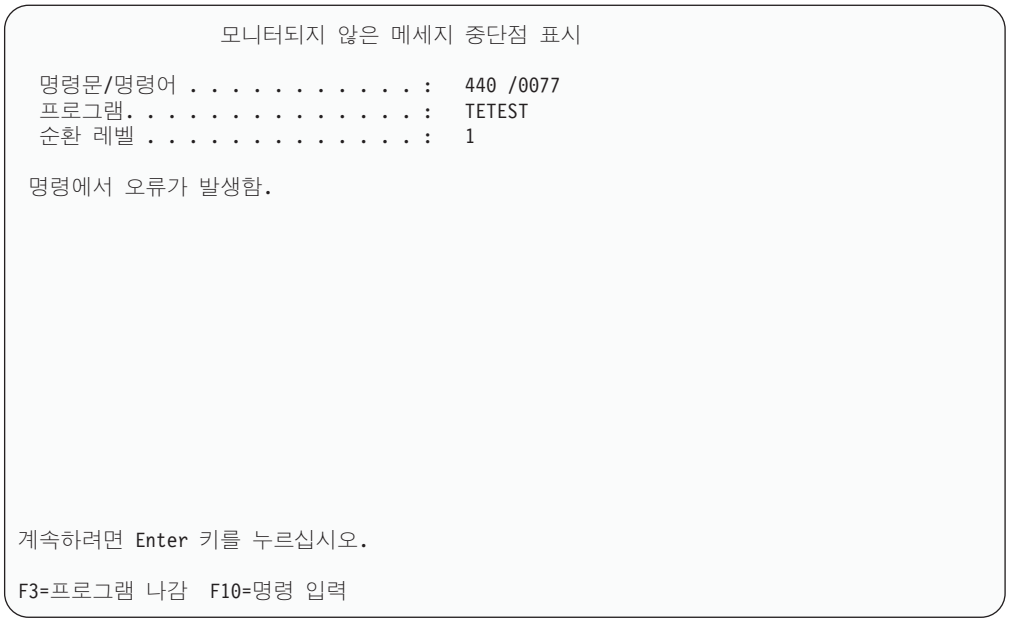
RPG/400® 프로그램은 프로그램 종료에 앞서 최종 레코드 인디케이터(LR)가 켜질 때 비활성화됩니다. 리턴 작업이 있고, LR이 꺼진 경우에는 프로그램이 비활성화되지 않습니다.

모니터되지 않은 메세지 처리

보통 프로그램이 모니터되지 않은 이탈 메세지를 수신하는 경우 시스템이 기능 검사 메세지(CPF9999)를 프로그램의 프로그램 메세지 대기행렬로 송신하고 프로그램이 처리를 중단합니다. 그러나 HLL 프로그램 컴파일러에서는 기능 검사 메세지나 프로그램에서 발생할 수 있는 메세지의 모니터를 삽입하기도 합니다. (조회 메세지는 프로그램 메세지 화면으로 송신됩니다.) 이것은 사용자가 원하는 방법으로 프로그램을 종료시킬 수 있도록 합니다. 대화식 디버그 작업에서 기능 검사가 발생하면 시스템은 디폴트 처리를 통해 프로그램을 중단시키는 대신 사용자에게 제어를 넘깁니다. 시스템은 모니터되지 않은 메세지 화면에 다음을 표시합니다.

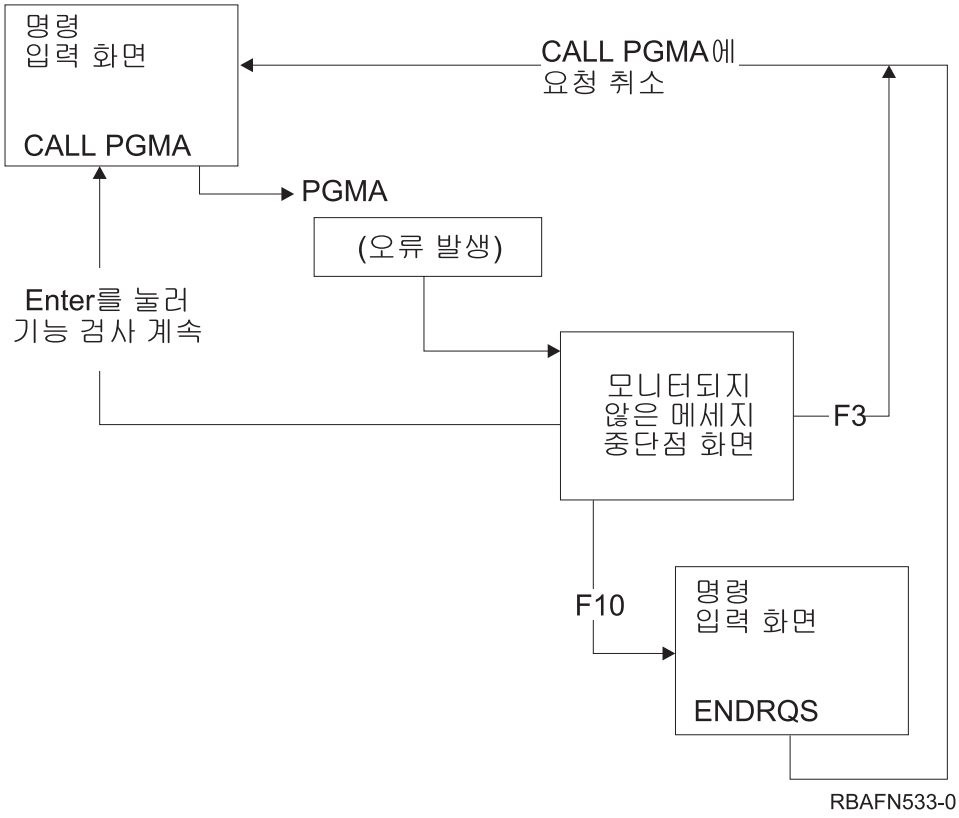
- 메세지
- 메세지가 송신된 MI 명령어 번호 및 HLL 명령문 ID(가능한 경우)
- 메세지가 송신된 프로그램의 이름과 순환 레벨

다음은 모니터되지 않은 메세지 중단점 화면의 예입니다.



사용자는 테스트 기능을 사용하여 오류가 발생된 소스를 분리하기 위한 시도를 할 수 있습니다. 그러나 오류가 있는 원래의 요구는 오류가 발생한 위치에서 중단된 상태로 있습니다. 오류가 발생한 요구를 호출 스택에서 제거하려면 ENDRQS(요구 종료) 명령을 사용하거나 모니터되지 않은 메세지 중단점 화면이 나올 때 F3 키를 누르십시오. '모니터되지 않은 메세지 중단점' 화면이 나올 때 Enter 키를 눌러 기능 검사 처리가 계속되도록 할 수도 있습니다. '명령 입력' 화면을 호출하려고 F10 키를 누른 경우에는 F3 키를 눌러 '모니터되지 않은 메세지 중단점' 화면으로 리턴하십시오.

다음은 ENDRQS 명령의 처리 과정입니다.



ENDRQS 명령이 입력되면 프로그램 호출이 취소됩니다. (이전 다이어그램에서 PGMA의 프로그램 호출이 취소되었습니다.)

중단점

중단점은 시스템이 프로그램 처리를 중단하고 표시장치(대화식 모드)에 있는 사용자 또는 ADDBKP(중단점 추가) 명령(일괄처리 모드)의 BKPPGM 매개변수에 지정된 프로그램으로 제어를 부여하는 프로그램에서의 한 위치입니다.

프로그램에 중단점 추가

디버그할 프로그램에 중단점을 추가하려는 경우 ADDBKP 명령을 사용하십시오. 하나의 ADDBKP 명령에 명령문 ID를 10개까지 지정할 수 있습니다. ADDBKP 명령에 지정된 프로그램 변수는 그 명령에 지정된 중단점에서만 적용됩니다. 하나의 ADDBKP 명령에는 변수를 10개까지 지정할 수 있습니다.

또한 중단점을 추가할 프로그램명을 지정할 수도 있습니다. 중단점을 추가할 프로그램명을 지정하지 않으면 중단점은 STRDBG, CHGDBG 또는 ADDPGM 명령에 지정된 디폴트 프로그램에 추가됩니다.

중단점 명령에 관한 자세한 내용은 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.

프로그램에 중단점을 추가하려면 아래와 같이 명령문 ID를 지정하십시오.

- 명령문 레이블
- 명령문 번호
- 기계 인터페이스(MI: Machine Interface) 명령어 번호

중단점을 프로그램에 추가할 경우 중단점에 도달할 때 표시할 값 또는 부분적인 값의 프로그램 변수를 지정할 수 있습니다. 이 변수는 문자나 16진 형식으로 표시됩니다.

프로그램 처리는 명령어가 처리되기 전에 중단점에서 중단됩니다. 대화식 작업인 경우 시스템은 프로그램이 중단되는 중단점과 프로그램 변수의 값(요구될 경우)을 표시합니다.

고급 언어(HLL) 프로그램에서는 서로 다른 명령문과 레이블들이 동일한 내부 명령어에 맵핑될 수 있습니다. 한 프로그램에서 실행 불가 명령문 다음에 또 다른 실행 불가 명령문(예: DO, ENDDO)이 올 때 이런 일이 발생합니다. IRP 리스트를 사용하여 동일한 명령어에 맵핑되는 것이 어떤 명령문 또는 레이블인지를 판별할 수 있습니다.

동일한 명령어에 여러 명령문들을 맵핑시키면, 추가되는 중단점이 다른 명령문에 추가되었던 이전의 중단점을 재정의하는 결과가 됩니다. 그러면 새로운 중단점은 이전에 추가되었던 중단점을 대체하게 되므로 이전의 중단점은 제거되고 새로운 중단점이 추가됩니다. 이러한 정보가 표시되면 다음 중 하나를 수행할 수 있습니다.

- F3 키를 눌러 가장 최근의 요구를 종료합니다.
- Enter 키를 눌러 프로그램 처리를 계속합니다.
- F10 키를 눌러 다음 요구 단계의 '명령 입력' 화면으로 갑니다. 이 화면에서는 다음 작업이 가능합니다.
  - 대화식 디버그 환경에서 사용될 수 있는 CL 명령을 입력합니다. 프로그램 안의 변수 값을 표시하거나 변경할 수 있으며, 디버그 모드에 프로그램을 추가하거나 디버그 모드에서 제거할 수 있고, 다른 디버그 명령을 수행할 수도 있습니다.
  - RSMBKP(중단점 재개) 명령을 입력하여 프로그램 처리를 계속합니다.
  - F3 키를 눌러 '중단점' 화면으로 리턴합니다.
  - ENDRQS(요구 종료) 명령을 입력하여 이전 요구 레벨의 '명령 입력' 화면으로 리턴합니다.

일괄처리 작업의 경우 중단점에 도달할 때 중단점 프로그램을 호출할 수 있습니다. 중단점 정보를 처리하려면 이러한 중단점 프로그램을 작성해야 합니다. 중단점 정보가 이 중단점 프로그램으로 전달됩니다. 중단점 프로그램은 대화식 대화식으로 작업에서 입력할 때 사용하는 명령(기능 요구)들이 들어 있는 CL 프로그램과 같은 또 하나의 프로그램입니다. 예를 들어, 변수를 표시하고 변경하거나 중단점을 추가하고 제거할 수 있는 프로그램입니다. 일괄처리 작업에서 유효한 기능이면 어떤 기능도 요구될 수 있습니다. 중단점 프로그램이 처리를 완료하면 디버그되고 있는 프로그램이 계속 처리됩니다.

디버그 작업의 각 중단점에 대한 메세지가 작업 기록부에 기록됩니다.

다음 ADDBKP 명령들은 프로그램 CUS310에 중단점을 추가합니다. CUS310은 디폴트 프로그램이므로 지정할 필요가 없습니다. 변수 &ARBAL의 값은 두 번째 중단점에 도달할 때 나옵니다.

```
ADDBKP  STMT(900)
ADDBKP  STMT(2200)  PGMVAR('&ARBAL')
```

주: CL 변수는 어포스트로피로 묶어서 입력해야 합니다.

다음은 CUS310의 소스문입니다.

```
5728PW1 R01M00  880101                SEU SOURCE LISTING

SOURCE FILE . . . . . .  QGPL/QCLSRC
MEMBER  . . . . . .  CUS310

SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...
  100 PGM   PARM(&NBRITEMS &ITEMPRC &PARBAL &PTOTBAL)
  200   DCL VAR(&PARBAL)   TYPE(*DEC) LEN(15 5) /* INPUT AREA INV BALANCE */
  300   DCL VAR(&PTOTBAL)  TYPE(*DEC) LEN(15 5) /* INPUT TOTAL INV BALANCE*/
  400   DCL VAR(&NBRITEMS) TYPE(*DEC) LEN(15 5) /* NUMBER OF ITEMS        */
  500   DCL VAR(&ITEMPRC)  TYPE(*DEC) LEN(15 5) /* PRICE OF THE ITEM      */
  600   DCL VAR(&ARBAL)    TYPE(*DEC) LEN(5 2)  /* AREA INVENTORY BALANCE */
  700   DCL VAR(&TOTBAL)   TYPE(*DEC) LEN(5 2)  /* TOTAL INVENTORY BALANCE*/
  800   DCL VAR(&TOTITEM)  TYPE(*DEC) LEN(5 2)  /* TOTAL PRICE OF ITEMS   */
  900   CHGVAR     VAR(&ARBAL) VALUE(&PARBAL)
 1000   CHGVAR     VAR(&TOTBAL) VALUE(&PTOTBAL)
 1100   IF COND(&NBRITEMS *EQ 0) THEN(DO)
 1200      SNDPGMMSG  MSG('The number of items is zero.  This item +
 1300                   should be ordered.') TOMSGQ(INVLIB/INVQUEUE)
 1400      GOTO       CMDLBL(EXIT)
 1500   ENDDO
 1600   CHGVAR     VAR(&TOTITEM) VALUE(&NBRITEMS * &ITEMPRC)
 1700   IF COND(&NBRITEMS *GT 50) THEN(DO)
 1800      SNDPGMMSG  MSG('Too much inventory for this item.') +
 1900                 TOMSGQ(INVLIB/INVQUEUE)
 2000   ENDDO
 2100   CHGVAR     VAR(&ARBAL)   VALUE(&ARBAL + &TOTITEM)
 2200   IF COND(&ARBAL *GT 1000) THEN(DO)
 2300      SNDPGMMSG  MSG('The area has too much money in +
 2400                   inventory.') TOMSGQ(INVLIB/INVQUEUE)
 2500   ENDDO
 2600   CHGVAR     VAR(&TOTBAL)  VALUE(&TOTBAL + &TOTITEM)
 2700   EXIT:      ENDPGM
```

다음은 첫 번째 중단점에 도달했을 때 표시되는 화면입니다.

```
                          중단점 표시

명령문/명령어 . . . . . . . . . . . . . :   900 /0009
프로그램. . . . . . . . . . . . . . . . :   CUS310
순환 레벨 . . . . . . . . . . . . . . . :   1









계속하려면 Enter 키를 누르십시오.

F3=프로그램 나감   F10=명령 입력
```

다음은 두 번째 중단점에 도달했을 때 표시되는 화면입니다.

```
                          중단점 표시

명령문/명령어 . . . . . . . . . . . . . :   2200 /0022
프로그램. . . . . . . . . . . . . . . . :   CUS310
순환 레벨 . . . . . . . . . . . . . . . :   1
시작 위치 . . . . . . . . . . . . . . . :   1
형식. . . . . . . . . . . . . . . . . . :   *CHAR
길이  . . . . . . . . . . . . . . . . . :   *DCL

변수  . . . . . . . . . . . . . . . . . :   &ARBAL
  유형  . . . . . . . . . . . . . . . . :   PACKED
  길이. . . . . . . . . . . . . . . . . :   5 2
 '610.00'









계속하려면 Enter 키를 누르십시오.

F3=프로그램 나감   F10=명령 입력
```

변수 &ARBAL이 표시됩니다. (&ARBAL의 값은 프로그램에 전달되는 매개변수의 값에 따라 변한다는 점에 주의하십시오.) F10 키를 눌러서 명령 입력 화면을 표시하여 프로그램의 처리를 변경하기 위한 변수 &ARBAL의 값을 변경할 수 있습니다. 또한 CHGPGMVAR(프로그램 변수 변경) 명령을 사용하여 변수 값을 변경할 수도 있습니다.

조건부 중단점

디버그 중인 프로그램에 조건부 중단점을 추가할 수 있습니다. ADDBKP(중단점 추가) 명령을 사용하여 명령문과 조건을 지정하십시오. 조건이 충족되면 시스템은 지정된 명령문에서 프로그램 처리를 중단합니다.

ADDBKP 명령에 건너뜀 값을 지정할 수 있습니다. **건너뜀 값**은 시스템이 프로그램을 중단하기 전에 명령문이 몇 번 처리되어야 하는지를 나타냅니다. 예를 들면, 명령문이 100번 처리된 후 명령문 1200에서 프로그램을 중단하도록 하려는 경우 다음 명령을 입력하십시오.

```
ADDBKP STMT(1200) SKIP(100)
```

SKIP 매개변수가 지정될 때 여러 개의 명령문이 지정되면 각 명령문은 별도의 계수를 가집니다. 다음 명령을 사용하면 명령문이 400®회 처리된 후에만 명령문 150 또는 200에서 프로그램을 중단합니다.

```
ADDBKP STMT(150 200) SKIP(400)
```

명령문 150이 400번 처리되었으나 명령문 200이 300번만 처리되었으므로 프로그램이 명령문 200에서 중단되지 않습니다.

명령문이 SKIP 매개변수에 지정된 수만큼 처리되지 않았으면, DSPBKP(중단점 표시) 명령을 사용하여 명령문의 처리 횟수를 표시할 수 있습니다. 명령문의 SKIP 계수를 0으로 재설정하려면 그 명령문에 대한 중단점을 다시 입력하십시오.

ADDBKP 명령에 보다 일반적인 중단점 조건을 지정할 수 있습니다. 이 표현식은 프로그램 변수, 연산자, 피연산자로서 다른 변수나 상수를 사용합니다. 예를 들어, 변수 &X가 1000보다 클 때 프로그램을 명령문 1500에서 중단하도록 하려는 경우 다음 명령을 입력하십시오.

```
ADDBKP STMT(1500) PGMVAR('&X') BKPCOND(*PGMVAR1 *GT 1000)
```

BKPCOND 매개변수는 세 가지 값을 필요로 합니다.

- 예에서는 첫 번째 값이 PGMVAR 매개변수에서 지정된 첫 번째 변수를 지정합니다. (세 번째 변수를 지정하려면 *PGMVAR3을 사용해야 합니다.)
- 두 번째 값은 반드시 연산자가 되어야 합니다. 유효한 모든 연산자 리스트에 대해서는 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션을 참조하십시오.
- 세 번째 값은 상수나 다른 변수가 될 수도 있습니다. 상수는 숫자, 문자 스트링 또는 비트 스트링이 될 수 있으며, 첫 번째 값에서 지정한 프로그램 변수와 동일한 유형이어야 합니다.

SKIP과 BKPCOND 매개변수는 복합 중단점 조건을 지정하기 위해 함께 사용이 가능합니다. 예를 들어, 지정 명령문이 50번 처리된 후 문자 스트링 &STR이 참일 경우에만 명령문 1000에서 프로그램을 중단하도록 하려는 경우 다음과 같은 명령을 입력하십시오.

```
ADDBKP STMT(1000) PGMVAR('&STR') SKIP(50)
       BKPCOND(*PGMVAR1 *EQ 'TRUE ')
```

프로그램에서 중단점 제거

프로그램에서 중단점을 제거하려면 RMVBKP(중단점 제거) 명령을 사용하십시오. 중단점을 제거하려면 중단점이 정의된 명령문의 명령문 번호를 지정해야 합니다.

추적

추적은 프로그램 내 명령문들의 처리 순서를 기록하는 프로세스입니다. 추적은 추적되는 동안 사용자가 제어를 갖지 못한다는 점에서 중단점과는 다릅니다. 시스템은 처리된 추적문을 기록합니다. 그러나 프로그램이 처리를 완료했을 때에는 추적 정보가 자동으로 표시되지 않습니다. DSPTRCDTA(추적 자료 표시) 명령을 사용하여 추적 정보의 표시를 요구해야 합니다. 이 화면에는 명령문이 처리되었던 순서와 ADDTRC(추적 추가) 명령에 지정된 변수 값(요구되는 경우)이 표시됩니다.

프로그램에 추적 추가

추적의 추가는 추적될 명령문을 지정하고, (원할 경우) 프로그램 변수명을 지정하는 것으로 이루어집니다. 추적되는 명령문이 처리되기 전에 변수 값이 기록됩니다. 또한 추적되는 명령문이 마지막으로 처리된 후에 변경된 변수 값만이 기록되도록 지정할 수도 있습니다. 이 변수는 문자 형식이나 16진 형식으로 표시될 수 있습니다.

추적할 명령문을 지정하려는 경우 다음과 같이 지정할 수 있습니다.

- 추적이 시작될 명령문 ID와 추적이 종료될 명령문 ID
- 프로그램에서 추적될 모든 명령문
- 추적될 명령문의 단일 명령문 ID

STRDBG 또는 CHGDBG 명령에는 한 작업에 대해 기록될 수 있는 명령문 추적의 수와 최대수에 도달했을 때 시스템이 수행해야 하는 조치를 지정할 수 있습니다. 최대값에 도달했을 때 시스템은 다음 조치 중 하나(사용자의 지정에 따라)를 수행합니다.

- 대화식 작업인 경우 다음 중 하나가 이루어질 수 있습니다.
  - 추적 중단(*STOPTRC). 사용자에게 제어가 부여되고(중단점 발생), 일부 추적 정의가 제거되며(RMVTRC 명령), 추적 자료를 지우거나(CLRTRCDTA 명령) 또는 최대값이 변경(CHGDBG 명령의 MAXTRC 매개변수)됩니다.
  - 추적 계속(*WRAP). 이전에 기록된 추적 자료 위에 이 시점 이후에 기록되는 추적 자료가 중복됩니다.
- 일괄처리 작업인 경우 다음 중 하나가 이루어질 수 있습니다.
  - 추적 중단(*STOPTRC). 추적 정의가 제거되고 프로그램이 처리를 계속합니다.
  - 추적 계속(*WRAP). 이전에 기록된 추적 자료 위에 이 시점 이후에 기록되는 추적 자료가 중복됩니다.

디버그 작업중에는 CHGDBG(디버그 변경) 명령을 사용하여 최대 추적 수 및 디폴트 조치를 언제든지 변경할 수 있습니다. 그러나 이러한 변경은 이미 기록된 추적에는 영향을 미치지 않습니다.

하나의 프로그램에 대해서는 총 5개의 명령문 범위만 지정할 수 있으며, 이것은 프로그램의 모든 ADDTRC(추적 추가) 명령에서 가져온 총 수입니다. 또한 각 명령문 범위별로 10개의 변수만 지정할 수도 있습니다.

고급 언어(HLL) 프로그램에서는 서로 다른 명령문과 레이블들이 동일한 내부 명령어에 맵핑될 수 있습니다. 이것은 한 프로그램 안에서 실행 불가 명령문 다음에 또 다른 실행 불가 명령문(예: DO, END)이 오는 경우에 발생합니다. IRP 리스트를 사용하여 동일한 명령어에 맵핑되는 것이 어떤 명령문 또는 레이블인지를 판별할 수 있습니다.

CL 변수를 지정할 때 &와 변수명은 어포스트로피로 묶어야 합니다. 예를 들어, 다음과 같은 경우가 있습니다.

```
ADDTRC PGMVAR('&IN01')
```

명령문 범위를 지정할 때 중단문의 소스문 번호는 시작문의 명령문 번호보다 일반적으로 큽니다. 그러나 추적은 기계 인터페이스(MI) 명령어로 수행되며 일부 컴파일러(특히 RPG/400)에서는 MI 명령어의 순서가 소스문의 순서와 다른 프로그램이 생성됩니다. 따라서 어떤 경우에는 중단문의 MI문 번호가 시작문의 MI문 번호보다 크지 않을 수도 있으며, 이 경우 사용자가 메세지 CPF1982를 수신하게 됩니다.

이 메세지를 수신하게 되면 다음 중 하나를 수행하십시오.

- 프로그램 내 모든 명령문을 추적하십시오.
- 명령문 범위를 하나의 스펙으로 제한하십시오.
- 프로그램의 프로그램 중간 표시(IRP: Intermediate Representation of a Program) 리스트로부터 MI 명령어 번호를 사용하십시오(465 페이지의 『기계 인터페이스 레벨에서의 디버그』 참조).

다음의 ADDTRC(추적 추가) 명령은 프로그램 CUS310에 추적을 추가합니다. CUS310은 디폴트 프로그램이므로 지정할 필요가 없습니다. 변수 &TOTBAL의 값은 추적된 각 명령문이 처리된 시간 사이에서 그 값이 변경된 경우에만 기록됩니다.

```
ADDTRC STMT((900 2700))  PGMVAR('&TOTBAL')  OUTVAR(*CHG)
```

다음 화면들은 추적의 결과로 나타나며, DSPTRCDTA(추적 자료 표시) 명령을 사용하여 표시됩니다. 열 헤더가 모든 화면에서 나오지 않음에 주의하십시오.

```
                        추적 자료 표시

                명령문/
프로그램        명령어                  순환 레벨           순번
CUS310          900                       1             1

시작 위치 . . . . . . . . . . . . . . . :   1
길이  . . . . . . . . . . . . . . . . . :   *DCL
형식. . . . . . . . . . . . . . . . . . :   *CHAR

 변수 . . . . . . . . . . . . . . . . . :   &TOTBAL
  유형  . . . . . . . . . . . . . . . . :   PACKED
  길이. . . . . . . . . . . . . . . . . :   5 2
  '    .00'

                명령문/
프로그램        명령어                  순환 레벨           순번
CUS310          1000                      1             2
CUS310          1100                      1             3  +

계속하려면 Enter 키를 누르십시오.

F3=나감    F12=취소
```

```
                        추적 자료 표시

시작 위치 . . . . . . . . . . . . . . . :   1
길이  . . . . . . . . . . . . . . . . . :   *DCL
형식. . . . . . . . . . . . . . . . . . :   *CHAR

*변수 . . . . . . . . . . . . . . . . . :   &TOTBAL
  유형  . . . . . . . . . . . . . . . . :   PACKED
  길이. . . . . . . . . . . . . . . . . :   5 2
  '   1.00'

                명령문/
프로그램        명령어                  순환 레벨           순번
CUS310          1600                      1             4
CUS310          1700                      1             5
CUS310          2100                      1             6
CUS310          2200                      1             7
CUS310          2600                      1             8  +

계속하려면 Enter 키를 누르십시오.

F3=나감    F12=취소
```

```
                                추적 자료 표시

CUS310          2700                        1             9

시작 위치 . . . . . . . . . . . . . . . . :   1
길이  . . . . . . . . . . . . . . . . . . :   *DCL
형식. . . . . . . . . . . . . . . . . . . :   *CHAR

*변수 . . . . . . . . . . . . . . . . . . :   &TOTBAL
  유형  . . . . . . . . . . . . . . . . . :   PACKED
  길이. . . . . . . . . . . . . . . . . . :   5 2
  '   2.00'









계속하려면 Enter 키를 누르십시오.

F3=나감    F12=취소
```

명령어 단계화

STRDBG 또는 CHGDBG 명령을 사용하여 MAXTRC 매개변수를 1, TRCFULL 매개변수를 *STOPTRC로 설정하면 프로그램의 명령어들을 단계화할 수 있습니다. 추적 범위를 지정하고(ADDTRC 명령) 프로그램이 이 범위 안에서 명령어를 처리하면 오류 메세지가 표시된 중단점 화면이 표시됩니다. 이 때 Enter 키를 누르면 지정된 추적 범위 안에서 처리되는 다음 명령어에 대해서도 동일한 오류 메세지를 가진 또 다른 중단점 화면이 표시됩니다. 추적이 완료되면 추적 자료에는 추적된 명령어 리스트가 들어 있습니다. DSPTRCDTA(추적 자료 표시) 명령을 입력하여 이 자료를 표시할 수 있습니다.

추적 안에서의 중단점 사용

중단점은 추적 범위 안에서 사용될 수 있습니다. 사용자는 추적 안의 중단점 위치에서 추적 자료를 표시하여(DSPTRCDTA 명령) 어떤 조치를 수행해야 하는지를 결정할 수 있습니다. 추적 자료는 중단점이 발생하기 전에 기록됩니다. 추적 정보에는 명령문이 처리되기 전 임의의 변수 값이 들어 있습니다.

시스템에서 추적 정보 제거

추적 정보가 표시된 후 이 정보를 시스템에서 제거할 것인지 또는 시스템에 남겨둘 것인지를 DSPTRCDTA 명령에 지정할 수 있습니다. 추적 정보를 시스템에 남겨두면 다른 추적 정보가 이 정보에 추가됩니다. 그 정보는 디버그 작업이 종료되거나 ENDDBG 명령이 제출될 때까지 (제거되지 않는 한) 시스템에 남아 있습니다. CLRTRCDTA(추적 자료 지우기) 명령을 사용하여 추적 정보를 시스템에서 제거할 수도 있습니다.

프로그램에서 추적 제거

RMVTRC(추적 제거) 명령은 하나 이상의 ADDTRC(추적 추가) 명령에 지정된 범위를 모두 또는 일부 제거합니다. 추적 범위 제거는 RMVTRC 명령에 사용된 명령문 ID를 지정하거나 모든 범위를 제거하도록 지정하는 것으로 이루어집니다.

RMVTRC 명령의 STMT 매개변수를 통해 사용자는 다음을 지정할 수 있습니다.

- ADDTRC 명령이 정의하는 추적 방법에 관계 없이, 지정된 프로그램의 모든 HLL 문이나 기계 명령어 또는 둘 모두 추적되지 않습니다.
- 제거할 HLL문 또는 시스템 명령어 또는 둘 모두의 추적 시작 및 추적 중단 위치입니다.

RMVPGM 및 ENDDBG 명령도 추적을 제거할 수 있으나, 이 명령들은 프로그램을 디버그 모드에서 제거합니다.

표시 기능

디버그 모드에서 사용자는 디버그 작업의 설정 내용을 검토하는 데 필요한 테스트 정보를 표시할 수 있습니다. 또한 디버그 모드에 있는 프로그램들과 이 프로그램에 대해 정의된 중단점 및 추적을 표시할 수 있습니다. 그 밖에 디버그 모드에 있는 프로그램의 상태를 표시할 수 있습니다.

테스트 정보를 표시하려는 경우 다음 명령을 사용할 수 있습니다.

- DSPDBG(디버그 표시). 현재 호출 스택, 디버그 모드 상태에 있는 프로그램명 및 다음을 표시합니다.
  - 중단점(breakpoint)에서 중단된 것
  - 현재 호출된 것
  - 호출된 것의 요구 레벨
  - 디버그 작업에 대해 선택된 디버그 옵션
- DSPBKP(중단점 표시). 프로그램에 현재 정의되어 있는 중단점의 위치를 표시합니다.
- DSPTRC(추적 표시). 프로그램에 현재 정의된 명령문 또는 명령문 범위를 표시합니다.

변수 값 표시

중단점에 있을 때 사용자는 프로그램 변수의 값을 표시할 수 있습니다. ADDBKP 명령에 변수명을 지정하여 중단점 화면에서 자동으로 표시하거나 명령 입력 화면을 표시하는 F10 키를 눌러 중단점에서 DSPPGMVAR(프로그램 변수 표시) 명령을 입력하여 표시할 수도 있습니다. 하나의 DSPPGMVAR 명령에는 10개의 변수만 지정할 수 있

습니다. 문자 변수와 비트 변수의 경우 지정된 길이의 변수 값을 특정 지점에서 시작하여 표시하도록 시스템에 지시할 수 있습니다. 변수는 문자 형식이나 16진 형식으로 표시할 수 있습니다.

주:

1. 배열 변수를 지정하면 다음 중 하나를 수행할 수 있습니다.
   a. 표시하려는 배열 요소의 첨자 값을 지정합니다. 첨자 값은 정수값이거나 프로그램 안의 숫자 변수명일 수 있습니다.
   b. 첨자를 입력하지 않고 배열 전체를 표시합니다.
   c. 하나만 제외하고 모든 첨자에 대해 값을 지정하며, 제외된 하나의 첨자 값에는 별표(*)를 지정하여 배열의 일차원 교차 섹션을 표시합니다.
2. 변수명은 단순명 또는 규정된 이름으로 지정이 가능하며, 어포스트로피(')로 묶어야 합니다. 규정된 이름은 다음 중 한 가지 방법으로 지정이 가능합니다.
   a. 가장 낮은 규정화 레벨에서 가장 높은 규정화 레벨순으로 되어 있으며, 특수 분리자 OF 또는 IN으로 대체되는 변수명. 변수명과 특수 분리자 사이는 공백으로 분리해야 합니다.
   b. 가장 높은 규정화 레벨에서 가장 낮은 규정화 레벨순으로 되어 있으며, 마침표로 분리되는 변수명.

다음의 DSPPGMVAR 명령은 프로그램 CUS310에서 사용된 변수 ARBAL을 표시합니다. CUS310은 디폴트 프로그램이므로 지정할 필요가 없습니다. 전체 값은 문자 형식으로 표시됩니다.

```
DSPPGMVAR  PGMVAR('&ARBAL')
```

결과로 나오는 화면은 다음과 같습니다.

```
                              프로그램 변수 표시

프로그램. . . . . . . . . . . . . . . . :   CUS310
순환 레벨 . . . . . . . . . . . . . . . :   1
시작 위치 . . . . . . . . . . . . . . . :   1
형식. . . . . . . . . . . . . . . . . . :   *CHAR
길이  . . . . . . . . . . . . . . . . . :   *DCL

변수  . . . . . . . . . . . . . . . . . :   &ARBAL
  유형  . . . . . . . . . . . . . . . . :   PACKED
  길이. . . . . . . . . . . . . . . . . :   5 2
 '610.00'









계속하려면 Enter 키를 누르십시오.

F3=나감   F12=취소
```

일부 고급 언어(HLL)에서는 변수를 사용자 정의 포인터 변수(HLL 포인터)에 기준하여 허용합니다. 기본 변수에 포인터를 명확하게 지정하지 않는 경우 HLL 선언(있는 경우)에 지정된 포인터가 사용됩니다. 고급 언어(HLL) 선언에 기본 변수에 대한 기본 포인터가 명확하게 지정되지 않은 경우 반드시 지정해야 합니다. 기본 변수 참조 시 PGMVAR 매개변수를 사용하여 최대 5개까지의 명확한 기본 포인터를 지정할 수 있습니다. 기본 포인터가 여러 개 지정되었을 때는 두 번째 기본 포인터를 찾는 데 첫 번째 기본 포인터를 사용하며, 세 번째 기본 포인터를 찾는 데 두 번째 기본 포인터를 사용하는 것과 같은 방법으로 계속됩니다. 기본 포인터 리스트의 최종 포인터는 1차 변수를 찾는 데 사용됩니다.

변수 값 변경

프로그램 변수의 값을 변경하려면 CHGPGMVAR(프로그램 변수 변경), CHGHLLPTR(HLL 포인터 변경) 또는 CHGPTR(포인터 변경) 명령을 사용하십시오. 프로그램 변수 값을 변경하는 것은 변수명과 변수의 자료 유형과 호환되는 값을 지정하는 것으로 구성됩니다. 예를 들면, 변수가 문자 유형이면 문자 값을 입력해야 합니다.

변수 값을 변경할 때 사용자는 변수가 자동 변수인지, 정적 변수인지를 확인해야 합니다. 이 두 변수의 차이점은 변수에 대한 기억장치에 있습니다. 자동 변수의 경우 기억장치가 프로그램의 호출과 관련됩니다. 따라서 프로그램이 호출될 때마다 변수의 신규 사본이 자동 기억장치에 위치합니다. 자동 변수에 대한 변경은 변경이 이루어진 프로그램 호출에서만 효력이 있습니다.

주: 어떤 언어에서는 프로그램 레벨이 아닌 프로시듀어 레벨에서 호출이 이루어집니다. 이런 경우 자동 변수에 대한 기억장치는 프로시듀어 호출과 관련됩니다. 프로시듀어가 호출될 때마다 변수의 신규 사본이 만들어집니다. 자동 변수로의 변경은 이 프로시듀어가 호출되는 동안에만 효력이 있습니다. 가장 최근의 프로시듀어 호출 내에 있는 자동 변수만이 변경될 수 있습니다. 명령상의 RCRLVL(순환 레벨) 매개변수는 프로시듀어 차원이 아닌 프로그램 차원에만 적용됩니다.

정적 변수의 경우 기억장치는 활성화와 관련됩니다. 정적 변수의 사본은 프로그램이 호출되는 횟수와 관계 없이 하나의 사본만 기억장치에 존재합니다. 정적 변수의 변경은 활성화중에만 유효합니다.

프로그램 변수가 정적 변수인지 아니면 자동 변수인지를 알아보려면 변수가 들어 있는 프로그램이 작성될 때 프로그램의 중간 표시(IRP: Intermediate Representation of a Program) 리스트(GENOPT 매개변수의 *LIST와 *XREF)를 요구하십시오.

배열인 변수를 변경하려면 배열의 한 요소를 지정해야 합니다. 즉, 변경하려는 배열 요소에 대해 첨자 값을 지정해야 합니다.

다른 작업의 디버그를 위해 작업 사용

다음 이유 중 한 가지로 인해 사용자가 별도의 작업을 통해 다른 작업에서 수행 중인 프로그램을 디버그하려는 경우가 있을 것입니다.

- 대화식 작업에 의해 일괄처리 작업이 디버그될 수 있습니다.
- 대화식 작업이 또 다른 대화식 작업으로부터 디버그될 수 있습니다. 이것은 어플리케이션 프로그램 화면을 인터럽트하지 않고서도 디버그 정보를 한 화면에 표시할 수 있도록 합니다.
- 루핑 상태의 대화식 또는 일괄처리 작업을 인터럽트하여 디버그 모드 상태로 만들 수 있습니다.

작업 대기행렬에 제출된 일괄처리 작업의 디버그

작업 대기행렬에 제출된 다른 일괄처리 작업을 디버그하기 위해 별도의 작업을 사용하는 경우 일괄처리 작업을 디버그 모드에 두고 작업 처리가 시작되기 전에 중단점과 추적을 설정할 수 있습니다. 작업 대기행렬에 제출되는 일괄처리 작업을 디버그하려면 다음 단계를 사용하십시오.

1. SBMJOB(작업 제출) 명령을 사용하거나 HOLD(*YES) 상태의 작업을 자동으로 제출하는 프로그램을 사용하여 일괄처리 작업을 제출하십시오.

   ```
   SBMJOB HOLD(*YES)
   ```

2. WRKSBMJOB(제출된 작업에 대한 작업) 명령 또는 WRKJOBQ(작업 대기행렬에 대한 작업) 명령을 사용하여 작업에 할당되는 규정된 작업명(번호/사용자/이름)을 결정하십시오. 또한 SBMJOB 명령은 명령이 처리를 끝내면 완료 메세지에 작업명을 표시합니다.

   WRKJOBQ(작업 대기행렬에 대한 작업) 명령은 특정 작업 대기행렬에서 시작되기를 기다리고 있는 모든 작업을 표시합니다. 작업에 대해 옵션 5를 선택하면 이 화면에서 작업명을 볼 수 있습니다.

3. 사용자가 일괄처리 작업의 디버그를 계획하는 화면에서 다음과 같이 STRSRVJOB(서비스 작업 시작) 명령을 입력하십시오.

   ```
   STRSRVJOB JOB(qualified-job-name)
   ```

4. STRDBG 명령을 입력하고 디버그될 모든 프로그램명을 제공하십시오. 작업이 작업 대기행렬에서 대기 중일 때는 다른 디버그 명령이 입력될 수 없습니다.
5. RLSJOBQ(작업 대기행렬 해제) 명령을 사용하여 작업 대기행렬을 해제하십시오. 작업 시작 준비가 되면 작업 디버그를 시작할 수 있다는 표시가 화면에 나옵니다. F10 키를 눌러 '명령 입력' 화면을 표시하십시오.
6. ADDBKP(중단점 추가) 또는 ADDTRC(추적 추가) 명령과 같은 디버그 명령을 '명령 입력' 화면을 사용하여 입력하십시오.
7. F3 키를 눌러 '명령 입력' 화면에서 나간 후 Enter 키를 눌러 일괄처리 작업을 시작하십시오.

8. 작업이 중단점에서 중단하면 일반적으로 중단점 화면이 나옵니다. 작업이 완료되면 중단점 및 추적을 추가하거나 변수를 표시 또는 변경할 수 없습니다. 그러나 DSPTRCDTA(추적 자료 표시) 명령을 사용하면 추적 자료를 표시할 수 있습니다.
9. 또 다른 일괄처리 작업을 디버그하려면 우선 ENDDBG(디버그 종료) 명령을 사용하여 디버그를 종료한 후 ENDSRVJOB(서비스 작업 종료) 명령을 사용하여 서비스 작업을 종료하십시오.

작업 대기행렬에서 시작되지 않은 일괄처리 작업 디버그

시스템에서 시작되는 일부 작업은 작업 대기행렬로 제출되지 않습니다. 이러한 작업들은 수행을 시작하기 전에는 중단시키는 것이 불가능하지만 일반적으로 디버그는 가능합니다. 작업 대기행렬에서 시작되지 않은 작업을 디버그하려면 다음과 같이 하십시오.

1. 작업이 시작되면 호출되는 프로그램의 이름을 변경하십시오. 예를 들면, 작업이 프로그램 CUST310을 수행하면 이 프로그램의 이름을 CUST310DBG로 변경할 수 있습니다.
2. 원래의 프로그램(이름의 변경되기 전)과 동일한 이름을 가진 작은 CL 프로그램을 작성하십시오. 작은 CL 프로그램에서 DLYJOB(작업 지연) 명령을 사용하여 1분 동안을 지연시킨 후 CALL 명령을 사용하여 이름이 변경된 프로그램을 호출하십시오.
3. CL 프로그램이 1분 동안 지연되도록 일괄처리 작업을 시작하십시오.
4. WRKACTJOB(활동 작업에 대한 작업) 명령을 사용하여 수행 중인 일괄처리 작업을 찾으십시오. 화면이 나오면, 작업 다음에 옵션 5를 입력하여 규정된 작업명(qualified job name)을 구하십시오.
5. STRSRVJOB(서비스 작업 시작) 명령을 다음과 같이 입력하십시오.

   ```
   STRSRVJOB JOB(qualified-job-name)
   ```

6. STRDBG를 입력한 후 ADDBKP(중단점 추가) 또는 ADDTRC(추적 추가) 명령과 같은 다른 디버그 명령을 입력하십시오. 디버그를 계속하십시오.

수행 중인 작업 디버그

작업이 어떤 명령문을 수행할 것인지를 알면 이미 수행 중인 작업을 디버그할 수 있습니다. 예를 들면, 작업이 루핑 상태이거나 디버그될 프로그램을 아직 수행하지 않았다면, 수행 중인 프로그램을 디버그하기를 원할 수 있습니다. 다음과 같이 하면 수행되고 있는 작업을 디버그할 수 있습니다.

1. WRKACTJOB(활동 작업에 대한 작업) 명령을 사용하여 수행 중인 작업을 찾으십시오. 화면이 나오면, 작업 다음에 옵션 5를 입력하여 규정된 작업명을 구하십시오.
2. STRSRVJOB(서비스 작업 시작) 명령을 다음과 같이 입력하십시오.

   ```
   STRSRVJOB JOB(qualified-job-name)
   ```

3. STRDBG(디버그 시작) 명령을 입력하십시오. (이 명령을 입력해도 수행 중인 작업이 중단되지는 않습니다.)

주: DSPDBG(디버그 표시) 명령을 사용하여 호출 스택을 표시하십시오. 그러나 프로그램이 어떤 이유로 인해 중단되지 않는 한, 스택은 잠깐 동안만 정정되고 프로그램은 작업을 계속합니다.

4. 수행될 명령문을 알면 ADDBKP(중단점 추가) 명령을 입력하여 그 명령문에서 작업을 중단하십시오.

   수행될 명령문을 알지 못하면 다음과 같이 하십시오.

   a. ADDTRC(추적 추가) 명령을 입력하십시오.
   b. 잠시 후 RMVTRC(추적 제거) 명령을 입력하여 프로그램 추적을 중단하십시오.
   c. DSPTRCDTA(추적 자료 표시) 명령을 입력하여 처리된 명령문을 표시하십시오. 다음에 처리될 자료 명령문은 추적 자료를 사용하여 판별하십시오(예를 들면, 프로그램 루프 안에 있는 명령문).
   d. ADDBKP(중단점 추가) 명령을 입력하여 그 명령문에서 작업을 중단하십시오.

5. 프로그램이 중단점에서 중단될 때 원하는 디버그 명령을 입력하십시오.

다른 대화식 작업 디버그

작업이 수행 중인지 아니면 메뉴나 명령 입력 화면에서 대기 중인지에 관계 없이 다른 화면에서 작업을 디버그할 수 있습니다. 다른 대화식 작업을 디버그하려면 다음과 같이 하십시오.

1. 디버그될 작업의 규정된 작업명을 판별하십시오. 디버그될 작업의 화면에서 DSPJOB(작업 표시) 명령을 입력하거나 WRKACTJOB(활동 작업에 대한 작업) 명령을 사용하여 이름을 판별하십시오.
2. 규정된 작업명을 사용하여 STRSRVJOB(서비스 작업 시작) 명령을 입력하십시오.
3. STRDBG(디버그 시작) 명령을 입력한 후 원하는 다른 디버그 명령을 입력하십시오. 작업이 이미 수행 중이면 프로그램에서 처리될 명령문을 판별하기 위해 DSPDBG(디버그 표시) 명령을 입력할 수 있습니다.

디버그되는 작업이 중단점에서 중단되면 표시장치는 잠금 상태가 됩니다.

한 작업을 다른 작업에서 디버그할 때의 고려사항

다른 작업에서 대부분의 작업을 디버그할 수 있을지라도 다음을 고려해야 합니다.

- 디버그될 작업은 보류 또는 일시중단시킬 수 없습니다(예를 들면, 다른 그룹 작업이나 2차 작업을 수행할 때).
- STRSRVJOB(서비스 작업 시작) 명령으로 다른 작업을 서비스할 때 사용자는 서비스를 수행하고 있는 작업을 디버그할 수 없습니다. 모든 디버그 명령들은 서비스될 작업에만 적용됩니다. 서비스를 수행하는 작업을 디버그하려면 다른 작업의 서비스를 종료하거나 다른 작업이 서비스를 하게 한 후 그 작업을 디버그해야 합니다.
- 디버그 명령은 작업이 중단점에서 중단되지 않아도 다른 작업에서 조작이 가능합니다. 예를 들면, 수행 중인 작업을 디버그하고 있고 DSPPGMVAR(프로그램 변수 표

시) 명령을 입력하면 지정된 변수가 표시됩니다. 작업이 계속 수행되므로 변수 값은 명령이 입력된 후 곧바로 변경될 수 있습니다.

- 디버그되는 작업은 디버그 명령에 응답할 충분한 우선순위가 있어야 합니다. 사용자가 낮은 우선순위로 일괄처리 작업을 디버그하고 있고, 그 작업이 처리 시간을 얻지 못하면 사용자가 발행한 디버그 명령은 작업으로부터 응답을 기다립니다. 작업이 응답하지 않으면 명령이 종료되고 오류 메세지가 표시됩니다.
- 작업 자체가 디버그되고 있으면 작업을 서비스하거나 디버그할 수 없습니다. 그러나 작업이 다른 작업에 서비스를 제공하거나 디버그 중이면 작업을 서비스 및 디버그할 수 있습니다.

기계 인터페이스 레벨에서의 디버그

명령의 PGMVAR 매개변수에 MI 오브젝트 정의 벡터(ODV) 번호를 지정하고 명령의 STMT 매개변수에 MI 명령어 번호를 지정하면 기계 인터페이스(MI) 레벨에서 프로그램을 디버그할 수 있습니다. 시스템은 고급 언어(HLL: High Level Language) 명령문 번호에서와 마찬가지로 MI 명령어 번호에서도, 중단점이 있는 위치에서 중단합니다. ODV나 MI 명령어 번호는 반드시 그 앞에 슬래시(/)를 사용해야 하며, 시스템에 MI 레벨에서 디버그 중임을 알리기 위해서 이 번호를 어포스트로피(')로 묶어야 합니다(예, `/1A`).

ODV와 MI 명령어 번호는 대부분의 HLL 컴파일러에 의해 작성되는 IRP 리스팅에서 얻을 수 있습니다. 프로그램 작성 시 IRP 리스팅을 작성하는 GENOPT 매개변수의 *LIST 값을 사용하십시오.

주: 기계 인터페이스 레벨에서 디버그할 때는 기계 인터페이스 레벨에서 정의된 특성만 사용할 수 있으며, 보통 테스트 환경으로 전달되는 HLL 특성은 사용할 수 없습니다. 이 HLL 특성에는 변수 유형, 분수, 길이 및 배열 정보 등이 포함될 수 있습니다. 예를 들면, HLL 프로그램의 숫자 변수는 정확한 10진수 정렬 없이 표시되거나 문자 스트링으로도 표시될 수 있습니다.

보안 고려사항

프로그램을 디버그하려면 그 프로그램에 대해 *CHANGE 권한이 있어야 합니다. 사용자에게 프로그램을 디버그할 권한이 부여되어 있는지를 판별할 때 다른 사용자의 프로파일을 채택함으로써 사용이 가능하게 된 *CHANGE 권한은 고려되지 않습니다. 이것은 사용자가 다른 사용자의 프로파일을 채택하여 디버그 모드로 프로그램 자료에 액세스하지 못하도록 합니다.

또한 일시적으로 허용된 사용자 권한을 가지고 디버그하려는 프로그램의 사용자 정의 중단점에서는 일시적으로 허용된 프로파일 권한이 아닌 자신의 사용자 프로파일 권한만이 부여될 뿐입니다. ADDBKP(중단점 추가) 명령에 의해 추가되었든지 아니면 모니터되

지 않은 이탈 메세지에 의해 추가되었든지에 관계 없이, 이전의 프로그램 호출에 의해 일시적으로 허용된 모든 중단점에 대한 권한은 사용자에게 부여되지 않습니다.

디버깅 중 COPY, SAVE, RESTORE, CRTDUPOBJ, CHKOBJITG 사용

라이브러리나 프로그램을 지정하기 위해 CL 명령을 사용하는 경우에는 디버그 기능이 실행되는 동안 프로그램으로부터 중단점이나 명령문 추적을 **일시적으로 제거**시킬 수 있습니다. 중단점이나 명령문 추적은 CL 명령 실행을 완료하면 다시 *복원*됩니다. 중단점이나 추적이 제거될 때 작업 기록부에는 CPD190A 메세지가 놓이게 되고, 중단점이나 명령문 추적이 다시 복원되면 또 다른 CPD190A 메세지가 작업 기록부에 놓입니다.

중단점이나 명령문 추적은 라이브러리를 지정하기 위해 다음 CL 명령을 사용하는 경우 프로그램으로부터 **일시적으로 제거**될 수 있습니다.

CHKOBJITG	CPY	CPROBJ	RSTLIB	SAVLIB
	CPYLIB	CRTDUPOBJ	RSTOBJ	SAVOBJ
				SAVSYS
				SAVCHGOBJ

주: CL 명령이 프로그램에서 실행될 때는 프로그램으로 중단점이나 추적을 추가시키지 못할 수 있습니다. 프로그램에서 어떤 명령이든지 실행되고 있는 동안 ADDBKP(중단점 추가) 명령이나 ADDTRC(추적 추가) 명령을 사용하면 CPF7102 오류 메세지를 수신합니다.

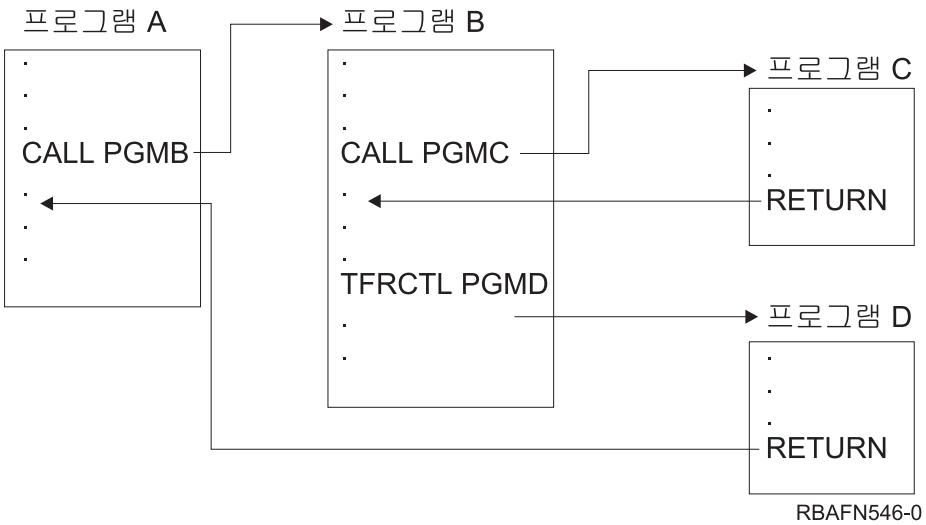
부록 A. TFRCTL 명령

TFRCTL(제어 전송) 명령은 명령에 지정된 프로그램을 호출하고, 그 프로그램으로 제어를 전달하며, 전송하는 프로그램을 호출 스택에서 제거합니다. 호출 스택상의 프로그램 수를 줄이면 성능이 향상될 수 있습니다. CALL 명령을 사용하면 호출된 프로그램은 CALL 명령이 들어 있는 프로그램으로 제어를 리턴시킵니다. TFRCTL 명령을 사용하면 호출 스택의 첫 번째 프로그램으로 제어가 리턴됩니다. 그러면, 첫 번째 프로그램이 CALL 명령 다음의 순차 명령어를 시작합니다.

주: ILE CL 프로시듀어에서는 TRFCTL 명령이 유효하지 않습니다.

TFRCTL 명령 사용

다음 그림에서 프로그램 A가 USRPRF(*OWNER)로 지정되면 소유자의 권한은 표시된 모든 프로그램에 영향을 미칩니다. 프로그램 B가 USRPRF(*OWNER)로 지정되면 소유자의 권한은 프로그램 B와 C가 활동 중일 때만 영향을 미칩니다. 프로그램 B가 프로그램 D로 제어를 전송하면 프로그램 B는 더 이상 호출 스택에 존재하지 않으며, 프로그램 B의 소유자는 프로그램 D가 수행 중인 동안은 권한이 있는 것으로 고려되지 않습니다. 프로그램이 처리를 완료하면(제어를 리턴하거나 전송함으로써) 소유자의 권한은 더 이상 유효하지 않습니다. 프로그램 B가 발행한 대체는 프로그램 D가 수행되는 동안에만 유효하며, 프로그램 D가 리턴되면 효력을 상실합니다.



TFRCTL 명령의 형식은 다음과 같습니다.

```
TFRCTL PGM(library-name/program-name) PARM(CL-variable)
```

프로그램(및 라이브러리 규정자)은 변수가 될 수 있습니다.

이 명령의 매개변수 인수로는 변수만 사용할 수 있으며, 이 변수는 전송 프로그램을 호출한 프로그램으로부터 인수 리스트 안의 매개변수로서 수신되어야 합니다. 즉, TFRCTL 명령은 TFRCTL 명령을 수행하는 프로그램으로 전달되지 않은 변수는 전달할 수 없습니다.

다음 예에서 첫 번째 TFRCTL은 유효합니다. 두 번째 TFRCTL 명령은 &B가 아래 프로그램으로 전달되지 않았으므로 유효하지 않습니다. 세 번째 TFRCTL 명령은 상수가 인수로 지정될 수 없으므로 유효하지 않습니다.

```
PGM PARM(&A)
DCL &A *DEC 3
DCL &B *CHAR 10
IF (&A *GT 100) THEN (TFRCTL PGM(PGMA) PARM(&A)) /*   valid   */
IF (&A *GT 50) THEN (TFRCTL PGM(PGMB) PARM(&B))  /* not valid */
ELSE (TFRCTL PGM(PGMC) PARM('1'))                /* not valid */
ENDPGM
```

PARM 매개변수는 84 페이지의 『프로그램과 프로시듀어 간의 매개변수 전달』에 설명되어 있습니다.

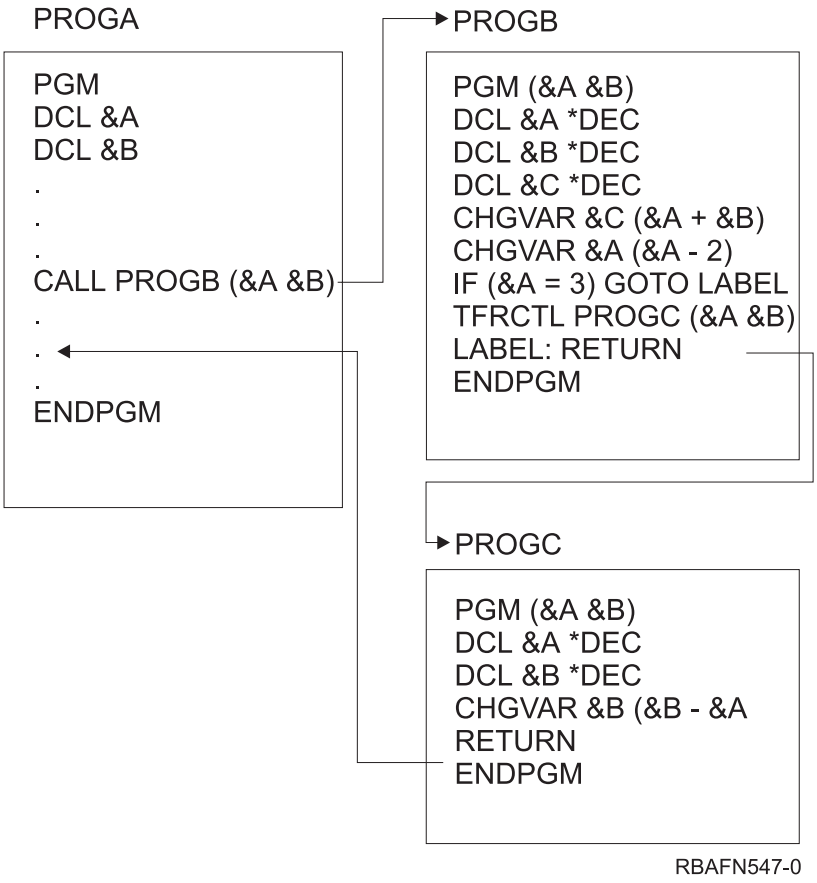
매개변수 전달

TFRCTL 명령은 CALL 명령이 매개변수를 전달하는 것과 같은 방식으로 호출된 프로그램으로 매개변수를 전달하는 데 사용될 수 있으나, 다음과 같은 제한사항이 있습니다.

- 전달된 매개변수가 CL 변수여야 합니다.
- 전송 프로그램이 전달한 CL 변수는 그 프로그램에 의해 매개변수로 수신되어야 합니다.
- 이 명령은 OPM CL 프로그램에서만 유효합니다.

다음 예에서 PROGA는 PROGB를 호출하여 두 개의 변수 &A와 &B를 PROGB에 전달합니다. PROGB는 이 두 개의 변수와 내부적으로 선언된 다른 변수 &C를 사용합니다. 제어가 PROGC로 전송되면 &A와 &B만 PROGC로 전달될 수 있습니다.

PROGC가 처리를 종료하면 제어는 처음에 이 변수를 시작한 PROGA로 리턴합니다.



부록 B. 작업 기록부 출력 파일

작업 기록부 지시

QMHCTLJL(작업 기록부 제어) API 또는 DSPJOBLOG(작업 기록부 표시) 명령을 사용하면 하나 또는 두 개의 데이터베이스 파일로 한 작업의 작업 기록부를 지시할 수 있습니다. 첫 번째 파일은 1차 작업 기록부 파일입니다. 이 파일에는 메세지 ID, 메세지 유형 및 메세지 심각도와 같은 메세지에 필수적인 정보가 들어 있습니다. 처리하려고 선택한 각 메세지에 대해 1차 작업 기록부 파일에서 하나의 레코드가 작성됩니다. 두 번째 파일은 2차 작업 기록부 파일입니다. QMHCTLJL API를 사용하여 이 파일을 작성할 수 있으나 이것은 선택사항이기도 합니다.

2차 작업 기록부 파일에는 메세지에 대한 1차 및 2차 레벨 텍스트가 들어 있습니다. 텍스트는 인쇄 형식으로 되어 있습니다. 메세지 자료는 메세지 설명으로 통합되고 결과는 하나 이상의 인쇄 형식으로 형식화됩니다. 처리를 위해 선택된 각 메세지의 경우 2차 작업 기록부 파일에 둘 이상의 레코드가 있을 수 있는데, 첫 번째 및 두 번째 레벨 인쇄 행에 레코드가 각각 한 개씩 있습니다.

1차 파일 안의 레코드는 메세지 참조 키를 사용하여 2차 파일 안의 레코드에 연관시킬 수 있습니다. 1차 파일 안의 각 레코드에는 관련 메세지의 메세지 참조 키(MRK)인 필드가 들어 있습니다. 이와 마찬가지로, 각 2차 파일 레코드 안에는 관련 메세지의 MRK가 들어 있습니다. 메세지의 MRK는 작업의 문맥 안에서 고유합니다. 1차 파일 레코드의 MRK가 일단 알려지고 나면, 관련 2차 레코드에만 동일한 MRK 값이 들어 있을 것이므로 이것을 쉽게 식별할 수 있습니다.

1차 작업 기록부 모델

1차 작업 기록부 파일을 위한 IBM 제공 모델은 QSYS 라이브러리 안의 QAMHJLPR입니다. 1차 레코드 형식은 QMHPFT입니다. 이 형식의 자세한 설명은 다음과 같습니다.

필드 순서	필드명	자료 유형	길이(바이트)	필드 설명
1	QMHJDT	DATE	10	작업 기록부 작성 날짜
2	QMHJTM	TIME	8	작업 기록부 작성 시간
3	QMHMRK	CHAR	4	메세지 참조 키
4	QMHTYP	CHAR	10	메세지 유형
5	QMHSEV	BIN	4	메세지 심각도
6	QMHMID	CHAR	7	메세지 ID
7	QMHDAT	DATE	10	메세지 송신 날짜
8	QMHTIM	TIME	8	메세지 송신 시간
9	QMHMF	CHAR	20	메세지 파일명
10	QMHRPY	CHAR	4	응답 참조 키

필드 순서	필드명	자료 유형	길이(바이트)	필드 설명
11	QMHRQS	CHAR	1	요구 메세지 상태
12	QMHSTY	CHAR	1	송신 프로그램 유형
13	QMHRTY	CHAR	1	수신 프로그램 유형
14	QMHSSN	BIN	4	송신 프로그램 명령문 수
15	QMHRSN	BIN	4	수신 프로그램 명령문 수
16	QMHCID	BIN	4	메세지 자료의 CCSID 또는 즉시 메세지
17	QMHPRL	CHAR	1	메세지 여과 인디케이터
18	QMHSPR	VAR CHAR	256 MAX	송신 프로시듀어명
19	QMHSMD	CHAR	10	송신 모듈명
20	QMHSPG	CHAR	12	송신 프로그램명
21	QMHSLB	CHAR	10	송신 라이브러리명
22	QMHSTM	CHAR	30	송신 프로그램의 명령문 번호
23	QMHRPR	VAR CHAR	256 MAX	수신 프로시듀어명
24	QMHRMD	CHAR	10	수신 모듈명
25	QMHRPG	CHAR	10	수신 프로그램명
26	QMHRLB	CHAR	10	수신 프로그램 라이브러리명
27	QMHRTM	CHAR	30	수신 프로그램의 명령문 번호
28	QMHSYS	CHAR	8	시스템명
29	QMHJOB	CHAR	26	규정된 작업명
30	QMHMDT	VAR CHAR	3000 MAX	메세지 자료 또는 즉시 메세지
31	QMHCSP	VAR CHAR	4096 MAX	완전한 송신 프로시듀어명
32	QMHCRP	VAR CHAR	4096 MAX	완전한 수신 프로시듀어명
33	QMHLSP	VAR CHAR	6144 MAX	긴 송신 프로그램명
34	QMHTID	CHAR	8	스레드
35	QMHMSC	ZONED	6,0	마이크로초
36	QMHFUS	CHAR	10	시작 사용자

이 레코드의 필드 정의는 다음과 같습니다.

QMHJDT

작업 기록부 작성 날짜; DATE(10)

작업 기록부 작성이 시작된 날짜. 필드는 데이터베이스 레코드의 날짜 필드입니다. 날짜의 형식은 *ISO입니다. 이 날짜 필드값의 형식은 yyyy-mm-dd입니다. 동일한 작업 기록부용으로 작성된 각 레코드는 이 필드에서 동일한 값을 가집니다.

QMHJTM

작업 기록부 작성 시간; TIME(8)

작업 기록부 작성이 시작된 시간. 이 필드는 데이터베이스 레코드에서 시간 필드로 정의됩니다. 시간 형식은 *ISO로 정의됩니다. 이 시간 필드값의 형식은 hh.mm.ss입니다. 동일한 작업 기록부용으로 작성된 각 레코드는 이 필드에서 동일한 값을 가집니다.

QMHMRK

메세지 참조 키; CHAR(4)

관련 메세지가 작업 메세지 대기행렬에서 가지고 있던 메세지 참조 키. 레코드는 메세지 참조 키별 오름차순으로 1차 데이터베이스 파일에 놓입니다. 단일 작업 기록부용으로 작성된 레코드 세트 안에서 이 필드는 각 레코드별로 고유하므로 레코드

에 대한 고유 키로서 사용할 수 있습니다. 둘 이상의 작업 기록부에 대한 레코드가 동일한 멤버에 놓이면 그 키가 더 이상 고유하지 않습니다.

QMHTYP

메세지 유형; CHAR(10)

관련 메세지의 메세지 유형. 다음 특수 값 중 하나가 이 필드에 나옵니다.

***CMD**
CL 프로그램 실행 시 기록된 명령

***COMP**
완료 메세지 유형

***COPY**
송신자의 사본 메세지 유형

***DIAG**
진단 메세지 유형

***ESCAPE**
이탈 메세지 유형

***INFO**
정보 메세지 유형

***INQ** 조회 메세지 유형

***NOTIFY**
통지 메세지 유형

***RQS** 요구 메세지 유형

***RPY** 응답 메세지 유형

QMHSEV

메세지 심각도; BIN(4)

메세지가 가진 심각도. 이 값은 0-99입니다.

QMHMID

메세지 ID; CHAR(7)

메세지에 대한 메세지 ID. 메세지 ID가 없는 즉시 메세지의 경우 이 필드에 특수 값 *IMMED가 포함됩니다.

QMHDAT

메세지 송신 날짜; DATE(10)

메세지가 송신된 날짜. 이 필드는 데이터베이스 레코드에 날짜 필드로 정의됩니다. 날짜의 형식은 *ISO입니다. 이 필드값의 형식은 yyyy-mm-dd입니다.

QMHTIM

메세지 송신 시간; TIME(8)

메세지가 송신된 시간. 이 필드는 데이터베이스 레코드에서 시간 필드로 정의됩니다. 시간 형식은 *ISO로 정의됩니다. 이 필드값의 형식은 hh.mm.ss입니다.

QMHMF

메세지 파일; CHAR(20)

메세지에 대한 메세지 설명을 확보하는 데 사용되는 메세지 파일명. 필드의 첫 번째 10자에는 메세지 파일명이 들어 있습니다. 두 번째 10자에는 라이브러리명이 들어 있습니다. 필드 QMHMID에 즉시 메세지를 나타내는 *IMMED가 들어 있으면, 이 필드는 모두 공백입니다.

QMHRPY

응답 참조 키; CHAR(4)

- 메세지 유형이 조회, 통지 또는 송신자의 사본인 경우 이것은 관련 응답 메세지의 메세지 참조 키입니다.
- 사용이 가능한 응답 메세지가 없으면, 이 필드에는 널값('00000000'X)이 들어 있습니다.
- 메세지 유형이 조회, 통지 또는 송신자의 사본이 아닌 경우에도 이 필드에는 널값이 들어 있습니다.

엄격하게 메세지 참조 키별로 오름차순을 유지하려면 응답 메세지에 대한 레코드가 조회, 통지 또는 송신자의 사본 메세지에 대한 레코드 바로 뒤에 있어서는 안됩니다.

QMHRQS

요구 메세지 상태; CHAR(1)

- 메세지 유형이 *RQS인 경우 이것은 요구 메세지가 수행되었는지 여부를 보여주는 인디케이터입니다.
- 인디케이터가 0('F0'X)으로 설정되어 있으면 요구가 실행되지 않은 것입니다.
- 인디케이터가 1('F1'X)로 설정되어 있으면 요구가 실행된 것입니다.

메세지 유형이 *RQS가 아닌 경우 이 인디케이터는 항상 0입니다.

QMHSTY

송신 프로그램 유형; CHAR(1)

인디케이터는 송신 프로그램이 OPM 프로그램 또는 ILE 프로그램인지의 여부를 나타내는 다음과 같은 값을 가집니다.

- 인디케이터가 1('F0'X)로 설정된 경우 송신 프로그램은 12자 이하의 프로시듀어명을 갖는 OPM 프로그램입니다. 프로그램명은 QMHSPG 필드 및 QMHLSP 필드에 놓입니다.

- 인디케이터가 0('F1'X)으로 설정된 경우 송신 프로그램은 256자 이하의 이름을 갖는 ILE 프로그램 또는 SLIC입니다. 프로시듀어명은 QMHSPR 및 QMHCSP 필드에 놓입니다.
- 인디케이터가 2('F2'X)로 설정된 경우 송신 프로그램은 257자 이상 4096자 이하의 프로시듀어명을 갖는 ILE 프로그램입니다. 완전한 송신 프로시듀어명은 QMHCSP 필드이며 QMHSPR 필드는 공백입니다.
- 인디케이터가 3('F3'X)으로 설정된 경우 송신 프로그램은 13자 이상 256자 이하의 이름을 갖는 SLIC 프로그램입니다. 완전한 송신 프로시듀어명은 QMHLSP 필드이며, QMHSPG 필드는 공백입니다.

QMHRTY

수신 프로그램 유형; CHAR(1)

수신 프로그램의 유형을 보여주는 다음과 같은 값을 사용하는 인디케이터:

- 이 인디케이터가 0('F0'X)으로 설정된 경우 수신 프로그램은 OPM 프로그램입니다. 프로그램명은 QMHRPG 필드에 놓입니다.
- 인디케이터가 1('F1'X)로 설정된 경우 수신 프로그램은 256자 이하의 프로시듀어명을 갖는 ILE 프로그램입니다. 프로시듀어명은 QMHRPR 및 QMHCRP 필드에 놓입니다.
- 인디케이터가 2('F2'X)로 설정된 경우 수신 프로그램은 257자 이상 4096자 이하의 프로시듀어명을 갖는 ILE 프로그램입니다. 전체 수신 프로시듀어명은 QMHCRP 필드에 놓이며, QMHRPR 필드는 공백입니다.

QMHSSN

송신 프로그램의 명령문 수; BIN(4)

송신 프로그램의 명령문 수.

- 송신 프로그램 유형 필드인 QMHSTY에 0('F0'X) 또는 3('F3'x)이 있으면, 이 필드에는 0 또는 1의 값이 포함됩니다.
- 송신 프로그램 유형 필드에 1('F1'X) 또는 2('F2'X)가 있으면, 이 필드에는 0, 1, 2, 3의 값이 포함됩니다.

이 필드에 제공된 값이 QMHSTM 필드에 있는 명령문의 수를 정의합니다.

QMHRSN

수신 프로그램의 명령문 수; BIN(4)

수신 프로그램의 명령문 번호의 수.

- 수신 프로그램 유형 필드인 QMHRTY에 0('F0'X)이 있으면, 이 필드에는 0 또는 1의 값이 포함됩니다.
- 수신 프로그램 유형 필드에 1('F1'X) 또는 2('F2'X)가 있으면, 이 필드에는 0, 1, 2, 3의 값이 포함됩니다. 이 필드에 제공되는 값이 QMHRTM 필드에 있는 명령문 수를 정의합니다.

QMHCID

메세지 자료의 CCSID; BIN(4)

메세지 자료의 CCSID나 QMHMDT 필드에 포함된 즉시 메세지.

QMHPRL

메세지 여과 인디케이터; CHAR(1)

메세지가 수신 프로그램으로 여과되었는지 여부를 보여주는 인디케이터.

- 메세지가 여과되지 않은 경우에는 이 필드에 0('F0'X)이 포함됩니다.
- 메세지가 송신된 경우에는 이 필드에 1('F1'X)이 포함됩니다.

메세지 여과는 ILE 프로그램에서만 발생할 수 있습니다. 따라서 수신 프로그램 유형 필드인 QMHRTY에 1('F1'X)이나 2('F2'X)가 있으면, 이 필드에는 1만 포함됩니다.

QMHSPR

송신 프로시듀어명; VAR CHAR(*)

- 송신 프로그램 유형 필드인 QMHSTY에 0('F0'X)이나 3('F3'X)이 있으면, 이 필드에는 값 *N이 포함됩니다.
- 송신 프로그램 유형 필드인 QMHSTY에 1('F1'X)이 포함되면 이 필드에는 송신 ILE 프로시듀어명이 포함됩니다. 이름의 최대 길이는 256자입니다.
- 송신 프로그램 유형 필드인 QMHSTY에 2('F2'X)가 있으면 이 필드에는 공백이 포함되며, 반면에 전체 이름은 QMHCSP 필드에 포함됩니다.

이 필드에는 송신 프로그램 유형 1('F1'X) 또는 2('F2'X)에 대한 내포 프로시듀어명이 있으며, 각 프로시듀어명은 콜론으로 분리됩니다. 맨 바깥쪽의 프로시듀어명이 처음으로 식별되고, 그 다음으로 그 안에 포함된 프로시듀어가 식별됩니다. 맨 안쪽의 프로시듀어는 스트링에서 마지막으로 식별됩니다.

QMHSMD

송신 모듈명; CHAR(10)

- 송신 프로그램 유형 필드인 QMHSTY에 0('F0'X)이나 3('F3'X)이 있으면, 이 필드에는 값 *N이 포함됩니다.
- 송신 프로그램 유형 필드인 QMHSTY에 1('F1'X)이나 2('F2'X)가 있으면, 이 필드에는 송신 ILE 모듈명이 포함됩니다.

QMHSPG

송신 프로그램명; CHAR(12)

- 송신 프로그램 유형 필드인 QMHSTY에 0('F0'X), 1('F1'X)이나 2('F2'X)가 있으면, 그 필드에는 송신 메세지로부터의 프로그램명이 포함됩니다.
- 송신 프로그램 유형이 3('F3'X)이면 이 필드에는 공백이 포함되고, QMHLSP 필드에는 송신 프로그램명이 포함됩니다.

QMHSLB

송신 라이브러리명; CHAR(10)

송신 프로그램이 포함되었던 라이브러리명.

QMHSTM

송신 프로그램의 명령문 번호; CHAR(30)

송신 프로그램이 메세지를 송신했던 명령문 번호. 각 명령문 번호의 길이는 10자입니다.

- 송신 프로그램 유형 필드인 QMHSTY에 0('F0'X)이나 3('F3'X)이 있으면, 처음 10자리에는 적어도 하나의 명령문 번호가 들어갑니다. 그 명령문 번호가 MI 명령어 번호를 나타냅니다. 그 번호는 16진수입니다.
- 송신 프로그램의 유형 필드에 1('F1'X) 또는 2('F2'X)가 들어 있으면, 이 필드는 명령문 번호 0, 1, 2 또는 3을 포함할 수 있습니다. 필드 QMHSSN은 명령문 번호의 수를 지정합니다. 이 경우 명령문 번호는 고급 언어 명령문 번호이며 MI 명령어 번호가 아닙니다. 각 번호는 10진수입니다.

QMHRPR

수신 프로시듀어명; VAR CHAR(*)

- 수신 프로그램 유형 필드에 0('F0)이 있으면, 이 필드에는 값 *N이 포함됩니다.
- 수신 프로그램 유형 필드인 QMHRTY에 1('F1'X)이 있으면, 이 필드에는 수신 ILE 프로시듀어명이 포함됩니다. 이름의 최대 길이는 256자입니다.
- 수신 프로그램 유형 필드인 QMHRTY에 2('F2'X)가 있으면 이 필드는 공백을 포함하는 반면에 QMHCRP 필드에는 전체 이름이 포함됩니다.

이 필드에는 송신 프로그램 유형 1('F1'X) 또는 2('F2'X)에 대한 내포 프로시듀어명이 있으며, 각 프로시듀어명은 콜론으로 분리됩니다. 맨 바깥쪽의 프로시듀어명이 처음으로 식별되고, 그 다음으로 그 안에 포함된 프로시듀어가 식별됩니다. 맨 안쪽의 프로시듀어는 스트링에서 마지막으로 식별됩니다.

QMHRMD

수신 모듈명; CHAR(10)

- 수신 프로그램 유형 필드에 0('0F'X)이 있으면, 이 필드에는 값 *N이 포함됩니다.
- 수신 프로그램 유형 필드인 QMHRTY에 1('F1'X)이나 2('F2'X)가 있으면, 이 필드에는 수신 ILE 모듈명이 포함됩니다.

QMHRPG

수신 프로그램명; CHAR(10)

메세지가 송신되었던 OPM 또는 ILE 프로그램의 프로그램명.

QMHRLB

수신 라이브러리명; CHAR(10)

수신 프로그램이 들어 있었던 라이브러리명.

QMHRTM

수신 프로그램의 명령문 번호; CHAR(30)

메세지가 송신되었을 때 수신 프로그램이 중단되었던 명령문 번호. 각 명령문 번호의 길이는 10자입니다.

- 수신 프로그램 유형 필드인 QMHRTY에 0('F0'X)이 있으면, 적어도 처음 10자에는 하나의 명령문 번호가 들어갑니다. 그 명령문 번호가 MI 명령어 번호를 나타냅니다. 그 번호는 16진수입니다.

기타 다른 값을 가진 수신 프로그램 유형의 경우에는 이 필드에 0, 1, 2, 3 명령문 번호가 들어갑니다. 필드 QMHRSN은 명령문이 몇 개 있는지를 지정합니다. 이 경우 명령문 번호는 고급 언어 명령문 번호이며 MI 명령어 번호가 아닙니다. 각 번호는 10진수입니다.

QMHSYS

시스템명; CHAR(8)

작업 기록부가 작성되었던 시스템명.

QMHJOB

규정된 작업명; CHAR(26)

메세지가 기록되고 있는 완전히 규정된 작업 이름. 처음 10자리에는 작업명이, 그 다음 10자리에는 사용자명, 그리고 마지막 6자리에는 작업 번호가 들어 있습니다.

QMHMDT

메세지 자료; VAR CHAR(*)

필드 QMHMID에 특수 값 *IMMED가 들어 있으면, 이 필드는 즉시 메세지를 포함합니다. 그렇지 않으면 이 필드에는 메세지가 송신되었을 때 사용되었던 메세지 자료가 들어 있습니다. 이 필드는 최대 3000자까지 포함할 수 있습니다. 즉시 메세지나 메세지 자료가 더 긴 경우에는 3000자에서 잘립니다.

메세지 자료에 포인터가 포함되면 메세지 자료가 데이터베이스 파일에 기록되기 전에는 포인터가 유효해지지 않습니다.

QMHCSP

완전한 송신 프로시듀어명; CHAR(VAR)

- 송신 프로그램 유형 필드에 0('F0'X)이나 3('F3'X)이 있으면, 이 필드에는 공백이 포함됩니다.
- 송신 프로그램 유형 필드에 1('F1'X)이나 2('F2'X)가 있으면, 이 필드에는 전체 ILE 프로시듀어명이 포함됩니다. 이름의 최대 길이는 4096자입니다.

이 필드에는 내포 프로시듀어명이 있으며, 각 프로시듀어명은 콜론으로 분리됩니다. 맨 바깥쪽의 프로시듀어명이 처음으로 식별되고, 그 다음으로 그 안에 포함된 프로시듀어가 식별됩니다. 맨 안쪽의 프로시듀어는 스트링에서 마지막으로 식별됩니다.

QMHCRP

완전한 수신 프로시듀어명; CHAR(VAR)

- 송신 프로그램 유형이 0('F0'X)이면, 이 필드에는 공백이 포함됩니다.
- 수신 프로그램 유형이 1('F1'X)이나 2('F2'X)이면, 이 필드에는 전체 ILE 프로시듀어명이 포함됩니다. 이름의 최대 길이는 4096자입니다.

  이 필드에는 내포 프로시듀어명이 있으며, 각 프로시듀어명은 콜론으로 분리됩니다. 맨 바깥쪽의 프로시듀어명이 처음으로 식별되고, 그 다음으로 그 안에 포함된 프로시듀어가 식별됩니다. 맨 안쪽의 프로시듀어는 스트링에서 마지막으로 식별됩니다.

QMHLSP

긴 송신 프로그램명; CHAR(VAR)

이 필드에는 모든 송신 프로그램 유형에 대해 송신된 메세지로부터의 전체 송신 프로그램명이 포함됩니다. 이름의 최대 길이는 6144자입니다.

QMHTID

스레드; CHAR(8)

이 필드는 메세지를 송신한 작업 내의 스레드를 식별합니다.

QMHMSC

마이크로초; ZONED(6,0)

메세지가 전달된 시간의 마이크로초 부분입니다. 메세지가 전송된 시간을 더 자세히 알고자 할 때 사용할 수 있습니다.

QMHFUS

시작 사용자; CHAR(10)

메세지 송신 시 스레드 실행에 사용된 사용자 프로파일명입니다.

2차 작업 기록부 파일을 위한 IBM 제공 모델은 QSYS 라이브러리 안의 QAMHJLSC입니다. 2차 레코드 형식명은 QMHSFT입니다. 2차 레코드 형식의 자세한 설명은 다음과 같습니다.

필드 순서	필드명	자료 유형	길이(바이트)	필드 설명
1	QMHJDS	DATE	10	작업 기록부 작성 날짜
2	QMHJTS	TIME	8	작업 기록부 작성 시간
3	QMHMKS	CHAR	4	메세지 참조 키
7	QMHSYN	CHAR	8	시스템명
8	QMHJBN	CHAR	26	규정된 작업명
4	QMHLNN	BIN	4	메세지 행 번호

필드 순서	필드명	자료 유형	길이(바이트)	필드 설명
5	QMHSID	BIN	4	텍스트 행의 CCSID
6	QMHTTY	CHAR	1	메세지 텍스트 인디케이터
9	QMHLIN	CHAR	78	메세지 텍스트 행

필드 길이는 필드의 총 바이트 수를 나타냅니다.

이 레코드의 필드 정의는 다음과 같습니다.

QMHJDS

작업 기록부 작성 날짜; DATE(8)

작업 기록부 작성이 시작된 날짜. 필드는 데이터베이스 레코드의 날짜 필드입니다. 날짜의 형식은 *ISO입니다. 이 필드값의 형식은 yyyy-mm-dd입니다. 동일한 작업 기록부용으로 작성된 각 레코드는 이 필드에서 동일한 값을 가집니다.

QMHJTS

작업 기록부 작성 시간; TIME(8)

작업 기록부 작성이 시작된 시간. 이 필드는 데이터베이스 레코드에서 시간 필드로 정의됩니다. 시간 형식은 *ISO로 정의됩니다. 이 필드값의 형식은 hh.mm.ss입니다. 동일한 작업 기록부용으로 작성된 각 레코드는 이 필드에서 동일한 값을 가집니다.

QMHMKS

메세지 참조 키; CHAR(4)

관련 메세지가 작업 메세지 대기행렬에서 가지고 있던 메세지 참조 키. 레코드는 메세지 참조 키별로 오름차순으로 2차 데이터베이스 파일에 놓입니다. 특정 메세지 참조 키에 대해 둘 이상의 2차 레코드가 있을 수 있습니다. 이 필드는 관련된 1차 레코드에도 존재합니다. 따라서 1차 레코드에서 일단 메세지 참조 키가 확보되면 2차 파일에서 관련된 레코드를 읽는 데 사용할 수 있습니다.

QMHSYN

시스템명; CHAR(8)

작업 기록부가 작성되었던 시스템명.

QMHJBN

규정된 작업명; CHAR(26)

메세지가 기록되고 있는 완전히 규정된 작업 이름. 처음 10자리에는 작업명이, 그 다음 10자리에는 사용자명, 그리고 마지막 6자리에는 작업 번호가 들어 있습니다.

QMHLNN

메세지 행 번호; BIN(4)

텍스트 유형 안의 행 번호. 첫 번째 및 두 번째 레벨 텍스트의 경우 둘 다 행 번호가 텍스트의 첫 번째 행의 1에서 시작하여 그 레벨 안에서 각 행이 추가될 때마다 1씩 증가됩니다.

QMHSID

메세지 텍스트 행의 CCSID; BIN(4)

필드 QMHLIN에 포함된 메세지 텍스트 행의 CCSID.

QMHTTY

메세지 텍스트 유형; CHAR(1)

필드 QMHLIN에 첫 번째 또는 두 번째 레벨 텍스트의 행이 포함되는지를 지정하는 인디케이터. 이 필드는 다음 값 중 하나를 포함합니다.

1 필드 QMHLIN에는 1차 레벨 텍스트가 포함됩니다.

2 필드 QMHLIN에는 2차 레벨 텍스트가 포함됩니다.

QMHLIN

메세지 텍스트 행: CHAR(78)

이 필드에는 1차 또는 2차 레벨 텍스트의 한 행이 포함됩니다.

부록 C. 사용권 프로그램(LP) 안의 IBM 제공 라이브러리

iSeries 서버에는 많은 라이브러리의 정의가 들어 있습니다. 이 정의들이 시스템에 저장되는 대다수 오브젝트들의 구성 방법을 제공합니다.

다음 표에는 IBM 제공 라이브러리들이 자신이 제공하는 사용권 프로그램(LP) 밑에 영문자순으로 나옵니다.

- 『OS/400 사용권 프로그램용 IBM 제공 라이브러리』에서는 기본 오퍼레이팅 시스템 OS/400 (OS/400) 사용권 프로그램에서 사용하기 위해 제공되는 라이브러리를 보여줍니다.
- 485 페이지의 『다른 iSeries 사용권 프로그램용 IBM 제공 라이브러리』에서는 다른 모든 iSeries 사용권 프로그램용으로 제공되는 라이브러리를 보여줍니다.
  - 사용권 프로그램들은 각 프로그램의 완전한 서술명을 사용하여 영문자순으로 나옵니다. 시스템의 사용권 프로그램 설치 메뉴에 이 프로그램들이 나옵니다.
  - 또한 사용권 프로그램에 피처가 있으면 피처명과 번호가 그 피처를 포함하고 있는 라이브러리 앞에 나옵니다.

OS/400 사용권 프로그램용 IBM 제공 라이브러리

표 12. OS/400 프로그램용 IBM 제공 라이브러리

프로그램명	라이브러리명	라이브러리 목적
OS/400(5722-SS1)		
	QCCA	CCA 암호 서비스 제공자
	QCLUSTER	HA 교환 가능 자원
	QDB2MS	DB2 멀티시스템
	QDNS	정의역명 시스템
	QDOC	시스템 보조 기억장치 풀(ASP) 즉, ASP 1에 상주하는 문서 라이브러리 오브젝트가 들어 있습니다. 이 곳에 문서 라이브러리 서비스(DLS) 시스템 오브젝트가 상주합니다. 시스템과 함께 제공되지 않으므로 IPL 시 작성되지 않습니다.
	QDOC*nnnn*	사용자 보조 기억장치 풀(ASP)에 상주하는 문서 라이브러리 오브젝트가 들어 있습니다. *nnnn*은 0002 - 0016의 ASP 번호입니다. 시스템과 함께 제공되지 않으므로 IPL 시 작성되지 않습니다.
	QDSNX	시스템과 함께 제공되지 않으므로 설치 시 작성되지 않습니다.
	QFNTCPL	AFP* 호환용 폰트
	QFNTWT	추가 폰트
	QFPNTWE	NetWare 확장 통합
	QGDDM	그래픽 자료 표시 관리자(GDDM*) 및 표시 그래픽 루틴(PGR)용 지원

표 12. OS/400 프로그램용 IBM 제공 라이브러리 (계속)

프로그램명	라이브러리명	라이브러리 목적
	QGPL	사용자의 범용 라이브러리
	QGPLTEMP	시스템과 함께 제공되지 않으므로 설치 시 작성되지 않습니다.
	QHLPSYS	일부 시스템 기능의 온라인 문서
	QICSS	DCM(Digital Certificate Manager)
	QICU	Unicode용 국제적 구성요소
	QIWS	호스트 서버
	QJRNL	HA 저널 성능
	QMSE	매체 및 기억장치 확장 기능(MSE) 라이브러리
	QPASE	이식 가능 어플리케이션 솔루션 환경
	QPFRDATA	시스템 수집 성능 자료 라이브러리
	QQALIB	질의 응답 유틸리티 라이브러리
	QRCL	RCLSTG 명령을 통한 오브젝트 재생용 라이브러리
	QRECOVERY	시스템 회복 라이브러리
	QRPLOBJ	대체 오브젝트용 라이브러리
	QSCxxxxxxx	APAR 자료 수집용 자료 라이브러리로서 xxxxxxx는 문제점 식별자의 마지막 7자릿수
	QSHELL	Qshell
	QSMP	DB2 대칭형 멀티프로세싱
	QSOC	OptiConnect
	QSPL	스풀링 라이브러리
	QSR	오브젝트 연결
	QSRV	시스템 서비스 라이브러리
	QSYS	시스템 라이브러리
	QSYSCGI	확장 기본 디렉토리 지원
	QSYSDIR	확장 기본 디렉토리 지원
	QSYSINC	시스템 개방성 포함 제품
	QSYSLOCALE	*LOCALE 오브젝트 작성 시 사용할 로케일 소스 멤버가 포함되어 있습니다.
	QSYSNLS	확장 NLS 지원
	QSYSVxRxMx	iSeries CL 컴파일러 라이브러리, 이전 릴리스 지원으로서 x는 이전 릴리스의 버전, 릴리스, 수정 레벨입니다.
	QSYS2	확장 기본 지원으로서, 이름이 Q자로 시작하지 않는 오브젝트(예: CPI용 오브젝트)에 대한 보조 시스템 라이브러리입니다.
	QSYSxxxx	온라인 정보
	QTEMP	사용자의 임시 라이브러리
	QUSRSYS	추가 IBM 제공 오브젝트
	QUSRTEMP	시스템과 함께 제공되지 않으므로 설치 시 작성되지 않습니다.
	QUSRTOOL	예제 툴
System/36 환경용 라이브러리		
	QSSP	System/36 환경 라이브러리
	QS36F	System/36 환경 파일 라이브러리로서 시스템에서 작성합니다.
	#CGULIB	System/36 문자 생성 유틸리티(CGU)
	#DFULIB	System/36 자료 파일 유틸리티(DFU)
	#DSULIB	System/36 개발 지원 유틸리티(DSU)
	#LIBRARY	System/36 환경, 사용자의 범용 라이브러리
	#SDALIB	System/36 화면 설계 유틸리티(SDA)
	#SEULIB	System/36 소스 입력 유틸리티(SEU)

표 12. OS/400 프로그램용 IBM 제공 라이브러리 (계속)

프로그램명	라이브러리명	라이브러리 목적
System/38 환경용 라이브러리		
	QSYS38	System/38 환경 라이브러리

다른 iSeries 사용권 프로그램용 IBM 제공 라이브러리

표 13. iSeries 서버의 기타 LP용 라이브러리

프로그램명	라이브러리명	라이브러리 목적
iSeries용 IBM Advanced DBCS Printer Support (5722-AP1)		
	QAPS	확장 2바이트 문자 세트(DBCS) 프린터 지원
	QAPS2	AS/400용 Advanced DBCS Printer Support-IPDS
IBM AS/400용 확장 기능 인쇄 DBCS 폰트 (5769-FN1)		
	QFNT60	AFP DBCS Fonts/400-Base Support
	라이브러리 목적: 일본어 폰트	
	QFNT61	AFP DBCS Fonts/400-Japanese
	라이브러리 목적: 한국어 폰트	
	QFNT62	AFP DBCS Fonts/400-Korean
	라이브러리 목적: 대만어 폰트	
	QFNT63	AFP DBCS Fonts/400-Traditional Chinese
	라이브러리 목적: 간체 한자 폰트	
	QFNT64	AFP DBCS Fonts/400-Simplified Chinese
	라이브러리 목적: 태국어 폰트	
	QFNT65	AFP DBCS Fonts/400-Thai
IBM AS/400용 확장 기능 인쇄 폰트 (5769-FNT)		
	QFNT00	OS/400 Base support for AFP™ 폰트
	QFNT01	OS/400 Sonoran Serif** 폰트 지원 (Sonoran Serif는 기능 상 Monotype Times New Roman**과 같습니다.)
	QFNT02	OS/400 Sonoran Serif Headliner 폰트
	QFNT03	OS/400 Sonoran Sans Serif** 폰트 지원 (Sonoran Sans Serif는 기능 상 Monotype Arial**과 같습니다.)
	QFNT04	OS/400 Sonoran Sans Serif Headliner 폰트
	QFNT05	OS/400 Sonoran Sans Serif 압축 폰트
	QFNT06	OS/400 Sonoran San Serif 확장 폰트
	QFNT07	OS/400 Monotype Garamond** 폰트
	QFNT08	OS/400 Century Schoolbook** 폰트
	QFNT09	OS/400 Pi 및 특수 문자 폰트
	QFNT10	OS/400 ITC Souvenir** 폰트
	QFNT11	OS/400 ITC Avant Garde Gothic** 폰트
	QFNT12	OS/400 수학 및 과학 폰트
	QFNT13	OS/400 DATA1 폰트
	QFNT14	OS/400 APL2® 폰트
	QFNT15	OS/400 OCR A 및 OCR B 폰트
iSeries용 Advanced Job Scheduler(5722-JS1)		
	QIJS	작업 스케줄러
iSeries용 IBM Advanced Function Printing Utilities (5722-AF1)		

표 13. iSeries 서버의 기타 LP용 라이브러리 (계속)

프로그램명	라이브러리명	라이브러리 목적
	QAFP	iSeries용 IBM Advanced Function Printing Utilities
AS/400용 IBM Application Program Driver (5722-PD1)		
	QAPD	Application Program Driver/400
IBM iSeries용 백업 회복 및 매체 서비스 (5722-BR1)		
	QBRM	BRM Services/400 지원
	QUSRBRM	BRM Services/400 사용자 자료
	Q1ABRMSFnn	ASP(nn)당 저장 파일 라이브러리
IBM S/400용 업무용 그래픽 유틸리티 (5722-DS1)		
	QBGU	AS/400 Business Graphics Utility(BGU)
IBM iSeries용 CICS Transaction Server (5722-DFH)		
	QCICS	CICS/400
	QCICSSAMP	샘플 CICS® 어플리케이션 프로그램
iSeries용 IBM Communications Utilities (5722-CM1)		
	QRJE	리모트 작업 항목(RJE) 및 VM/MVS 브릿지
iSeries용 Content Manager(5722-VI1)		
	QVI	iSeries용 Content Manager
iSeries용 Content Manager OnDemand(5722-RD1)		
	QRDARS	OnDemand
	QUSRRDARS	OnDemand 사용자 자료
	QUSROND	OnDemand 사용자 자료
Cryptographic Access Provider 128비트 (5722-AC3)		
	QCAP3	Cryptographic Access Provider 128비트
IBM AS/400용 Cryptographic Support (5722-CR1)		
	QCRP	IBM AS/400용 Cryptographic Support
iSeries용 DataPropagator Relational V8(5722-DP4)		
	QDPR	DB2 DataPropagator
IBM iSeries용 DB2 조회 관리자 및 SQL 개발 킷 (5722-ST1)		
	QSQL	Structured Query Language/400(SQL/400®)
iSeries용 DB2 Universal Database Extenders V7.2(5722-DE1)		
	QDBEX	DB2 UDB Extenders
	QDBXM(옵션 2)	XML Extender
	QDB2TX(옵션 1)	Text Extender
	QIMO(옵션 3)	Text Search Engine
AS/400용 DCE Base Services(5769-DC1)		
	QDCE2	AS/400용 DCE Base Services
AS/400용 DCE DES Library Routines(5769-DC3)		
	QDCEE	DCE DES Library Routines
	QDCE2	AS/400용 DCE Base Services
Developer Kit for Java (5722-JV1)		
	QJAVA	Developer Kit for Java
iSeries용 HTTP Server(5722-DG1)		
	QHTTPSVR	HTTP Server
	QTCM	Triggered Cache Manager
ILE C (5722-WDS)		
	QCLE	ILE C
ILE C++ (5722-WDS)		
	QCPPLE	ILE C++
	QCXXN	ILE C++, 이전 릴리스 지원
ILE COBOL (5722-WDS)		
	QCBLLE	ILE COBOL 라이브러리

표 13. iSeries 서버의 기타 LP용 라이브러리 (계속)

프로그램명	라이브러리명	라이브러리 목적
	QCBLLEP	ILE COBOL, 이전 릴리스 지원
	QLBL	OPM COBOL
	#COBLIB	System/36 호환용 COBOL
	QCBL	System/38 호환용 COBOL
ILE RPG (5722-WDS)		
	QRPG	RPG/400
	QRPGLE	ILE RPG/400
	QRPGLEP	ILE RPG/400, 이전 릴리스 지원
	#RPGLIB	System/36 호환용 RPG II
	QRPG38	System/38 호환용 RPG III
iSeries용 Infoprint Server(5722-IP1)		
	QIPS	Infoprint Server
IBM e(logo)server iSeries Access 제품군 (5722-XW1)		
	QCA400W	iSeries Access 제품군 Base
Windows®용 iSeries Access(5722-XE1)		
	QCAEXP	Windows용 iSeries Access
웹용 iSeries Access(5722-XH1)		
	QIWA	웹용 iSeries Access
무선 iSeries Access(5722-XP1)		
	QIWR	무선 iSeries Access
iSeries Client Encryption(128비트) (5722-CE3)		
	QCE3	Client Encryption 128비트
Windows Server용 iSeries Integration(5722-WSV)		
	QNTAP	Windows Server용 Integration
IBM iSeries용 관리 시스템 서비스 (5722-MG1)		
	QSVMSS	SystemView Managed System Services/400
	QSVDSTRPS	사용자 오브젝트용 SystemView 분배 저장소로서 LP 설치 시 작성됩니다.
IBM iSeries용 성능 분석 툴 (5722-PT1)		
	QPFR	IBM iSeries용 성능 분석 툴
iSeries용 IBM 조회 (5722-QU1)		
	QQRYLIB	iSeries용 IBM 조회
IBM AS/400용 System/38 유틸리티 (5722-DB1)		
	QIDU	System/38 Query, 텍스트 관리 및 자료 파일 유틸리티(DFU)
IBM iSeries용 시스템 관리자 (5722-SM1)		
	QSMU	SystemView System Manager/400
IBM iSeries용 TCP/IP 연결 유틸리티 (5722-TC1)		
	QTCP	TCP/IP Connectivity Utilities/400
IBM Toolbox for Java(5722-JC1)		
	QJT400	IBM Toolbox for Java
AS/400용 VisualAge Generator Server(5769-VG1)		
	OVGEN	VisualAge Generator Server
	QGPL	사용자의 범용 라이브러리
IBM iSeries용 WebSphere Studio Development Suite (5722-WDS)		

표 13. iSeries 서버의 기타 LP용 라이브러리 (계속)

프로그램명	라이브러리명	라이브러리 목적
	QPDA	다음 툴과 유틸리티가 포함되어 있습니다. • APF(확장 프린터 기능) • CGU(문자 생성 유틸리티), DBCS 시스템 전용 • DFU(자료 파일 유틸리티) • ISDB(대화식 소스 디버거) • PDM(프로그래밍 개발 관리자) • RLU(보고서 배치 유틸리티) • SDA(화면 설계 유틸리티) • SEU(소스 입력 유틸리티)

부록 D. CL 명령어 및 키워드의 약어

이 섹션에는 IBM OS/400 및 기타 IBM iSeries 사용권 프로그램의 일부인 CL 명령에 사용되는 약어의 영문 리스트가 있습니다.

이 정보는 명령어 정의를 사용할 때 사용자들이 일관된 방식으로 명령어와 키워드의 이름을 지정할 수 있도록 지원하기 위한 것입니다. (339 페이지의 제 9 장 『명령 정의』 및 iSeries Information Center의 **프로그래밍** 범주에서 *CL* 섹션에 나오는 명령 정의 명령문에 대한 주제를 참조하십시오.)

CL 명령어 동사 약어

대부분의 CL 명령어는 일관된 이름 지정 스타일을 따르고 있습니다. 명령어의 처음 세 자는 수행하는 조치를 나타냅니다. 명령어의 나머지 글자들은 조치의 대상이 되는 오브젝트를 설명합니다.

세 자로 된 명령어 접두부의 또 다른 이름이 명령어 '동사'입니다. 모든 CL 명령어의 대다수가 다음과 같은 공통 명령어 동사 중 하나를 사용합니다.

동사 약어	의미
ADD	추가
CHG	변경
CRT	작성
DLT	삭제
DSP	표시
END	종료
RMV	제거
STR	시작
WRK	작업

다음은 명령어 동사에 사용되는 모든 약어의 리스트입니다.

동사 약어	의미
ADD	추가
ALC	할당
ALM	경보
ANS	응답
ANZ	분석
APY	적용
ASK	요청
BLD	빌드
CFG	구성
CHG	변경
CHK	검사
CLO	닫기

동사 약어	의미
CLR	지우기
CMP	비교
CNL	취소
CPH	암호 해독
CPR	압축
CPY	복사
CRT	작성
CVT	변환
DCP	압축 해제
DLC	할당 해제
DLT	삭제
DLY	지연
DMP	덤프
DSC	단절
DSP	표시
DUP	사본
EDT	편집
EJT	방출
EML	에뮬레이트
ENC	암호화
END	종료
EXP	내보내기
EXT	추출
FIL	파일
FMT	형식
FND	찾기
GEN	생성
GRT	부여
HLD	보류
IMP	가져오기
INS	설치
INZ	초기화
LNK	링크
LOD	로드
MGR	마이그레이트
MON	모니터
MOV	이동
MRG	병합
OPN	열기
ORD	순서
OVR	대체
PAG	페이지 지정
PKG	패키지
POS	위치
PRM	승격
PRT	인쇄
PWR	전원
QRY	조회
RCL	재생
RCV	수신
RGZ	재구성
RLS	릴리스
RMV	제거
RNM	이름 변경

동사 약어	의미
RPL	대체
RQS	요구
RRT	재라우트
RSM	재개
RST	복원
RTV	검색
RUN	실행
RVK	취소
SAV	저장
SBM	제출
SET	설정
SLT	선택
SND	송신
STR	시작
TFR	전송
TRC	추적
UPD	갱신
VFY	확인
VRY	변환
WRK	작업

CL 명령어 약어

다음은 명령어 동사 약어를 포함하여 CL 명령어에 사용되는 모든 약어의 리스트입니다.

명령어 약어	의미
A(접두어)	속성
ABN	비정상
ACC	액세스 코드
ACCGRP	액세스 그룹
ACG	계정
ACN	조치
ACNE	조치 항목
ACT	활동 중, 활동, 활성화
ADD	추가
ADM	어플리케이션 개발 관리자, 관리
ADP	허용, 허용하는
ADPI	어댑터 정보
ADPP	어댑터 프로파일
ADPT	어댑터
ADR	주소
ADSM	ADSTAR 분배 기억장치 관리자
AFP	확장 기능 인쇄
AGR	계약
AGT	에이전트
AJE	자동시작 작업 항목
ALC	할당
ALR	경고
ALRD	경고 설명
ALRTBL	경고표

명령어 약어	의미
ALS	별명
ANS	응답
ANZ	분석
AP	액세스 경로
APAR	APAR
APF	확장 프린터 기능
APP	어플리케이션
APPC	APPC(advanced program-to-program communications)
APPN	APPN(advanced peer-to-peer networking)
APY	적용
ARA	영역
ARC	아카이브
ASC	비동기
ASK	요청
ASN	연관
ASP	보조 기억장치 풀
AST	지원
ATM	비동기 전송 모드
ATN	어텐션
ATR	속성
AUD	감사
AUT	권한
AUTE	인증 항목
AUTL	권한 부여 리스트
BACK	뒤로
BAL	균형
BAS	BASIC 언어
BCD	바코드
BCH	일괄처리
BCK	백업
BCKUP	백업
BGU	업무용 그래픽 유틸리티
BKP	중단점
BKU	백업
BND	바인딩, 바인드
BP	부트 프로토콜
BRM	BRMS(백업 회복 및 매체 서비스)
BSC	2진 동기
BSCF	bsc 파일
BUF	버퍼
C	C 언어
CAL	캘린더
CALL	호출
CAP	캡처
CBL	COBOL 언어
CCS	제어 서버 변경
CCT	IPX 회로
CCTRTE	회로 라우트
CCTSRV	회로 서비스
CDE	코드, 코드화
CDS	코드화 자료 저장
CFG	구성
CFGL	구성 리스트

명령어 약어	의미
CFGLE	구성 리스트 항목
CGY	범주
CHG	변경
CHK	검사
CHT	도표
CICS	고객 정보 제어 시스템
CL	제어 언어
CLD	C 로케일 설명
CLG	카탈로그
CLNUP	클린업
CLO	닫기
CLR	지우기
CLS	클래스
CLT	클라이언트
CLU	클러스터
CMD	명령
CMN	통신
CMNE	통신 항목
CMNF	통신 파일
CMP	비교
CMT	확약
CNL	취소
CNN	연결
CNNL	연결 리스트
CNNLE	연결 리스트 항목
CNR	컨테이너
CNT	접속
CNV	대화
CODE	공동 개발 환경
COL	콜렉션
COM	커뮤니티
COSD	서비스 클래스 설명
CP	검사 지연중
CPH	암호 해독
CPIC	공통 프로그래밍 인터페이스 통신
CPP	C++ 언어
CPR	압축
CPT	구성요소
CPY	복사
CPYSCN	화면 복사
CRG	클러스터 자원 그룹
CRL	크롤러(crawler)
CRP	암호
CRQ	변경 요구
CRSDMN	정의역 간
CRT	작성
CSI	통신측 정보
CSL	콘솔
CST	제한사항, 사용자 정의
CTG	카트리지
CTL	제어
CTLD	제어기 설명
CUR	현재
CVG	범위

명령어 약어	의미
CVN	변환
CVT	변환
D(접두어)	설명
DAT	날짜
DB	데이터베이스
DBF	데이터베이스 파일
DBG	디버그
DCL	선언
DCP	압축 해제
DCT	사전
DDI	분산 자료 인터페이스
DDM	분산 자료 관리
DDMF	분산 자료 관리 파일
DEP	종속
DEV	장치
DEVD	장치 설명
DFN	정의
DFT	디폴트
DFU	자료 파일 유틸리티
DHCP	동적 호스트 구성 프로토콜
DIR	디렉토리
DIRE	디렉토리 항목
DIRSHD	디렉토리 새도우
DKT	디스켓
DKTF	디스켓 파일
DL	DataLink
DLC	할당 해제
DLF	DataLink 파일
DLFM	DataLink 파일 관리자
DLO	문서 라이브러리 오브젝트
DLT	삭제
DLY	지연
DMN	정의역
DMP	덤프
DNS	정의역명 서비스
DO	수행
DOC	문서
DOM	Domino
DPCQ	DSNX/PC 대기행렬
DPR	DataPropagator Relational
DSC	단절
DSK	디스크
DSP	표시
DSPF	화면 파일
DST	분배
DSTL	분배 리스트
DSTLE	분배 리스트 항목
DSTQ	분배 대기행렬
DSTSRV	분배 서비스
DTA	자료
DTAARA	자료 영역
DTAQ	자료 대기행렬
DUP	사본
DWN	다운

명령어 약어	의미
E(접두어)	입력
EDT	편집
EDTD	편집 설명
EDU	교육
EJT	방출
EML	에뮬레이트, 에뮬레이션
ENC	암호화
END	종료
ENR	등록
ENV	환경
ENVVAR	환경 변수
EPM	확장 프로그램 모델
ERR	오류
ETH	이더넷
EWC	확장 무선 제어기
EWL	확장 무선 회선
EXIT	나감
EXP	만기, 내보내기
EXT	추출
F(접두어)	파일
FAX	팩시밀리
FCN	기능
FCT	양식 제어표
FCTE	양식 제어표 항목
FD	파일 설명
FFD	파일 필드 설명
FIL	파일
FILL	채움
FLR	폴더
FMT	형식
FNC	재무관리
FND	찾기
FNT	폰트
FNTRSC	폰트 자원
FNTTBL	폰트표
FORM	양식
FORMD	양식 설명
FORMDF	양식 정의
FR	프레임 릴레이
FRM	from
FRW	방화벽
FTN	FORTRAN 언어
FTP	파일 전송 프로토콜
FTR	필터
GDF	그래픽 자료 형식
GEN	생성
GO	가기
GPH	그래픽
GPHPKG	그래프 패키지
GRP	그룹
GRT	부여
GSS	그래픽 기호 세트
HDB	호스트 데이터베이스
HDW	하드웨어

명령어 약어	의미
HDWRSC	하드웨어 자원
HLD	보류
HLL	고급 레벨 언어
HLP	도움말
HLR	보유자
HOST	호스트
HST	이력
HTE	호스트표 항목
HTTP	하이퍼텍스트 전송 프로토콜
I(접두어)	정보, 항목, ILE
ICF	시스템 내 통신 기능
ICFF	icf 파일
IDD	대화식 자료 정의 유틸리티
IDLC	ISDN 자료 링크 제어
IDX	색인
IDXE	색인 항목
IFC	인터페이스
IMG	이미지
IMP	가져오기
INF	정보
INP	입력
INS	설치
INT	내부 기계
INTR	시스템 내
INZ	초기화
IPI	IPX를 통한 인터넷 프로토콜
IPL	초기 프로그램 로드
IPS	SNA를 통한 인터넷 프로토콜
IPX	네트워크 간 패킷 교환
IPXD	IPX 설명
ISDB	대화식 소스 디버거
ISDN	종합 정보 통신망
ITF	대화식 단말 기능
ITG	무결성
ITM	항목
IWS	지능형 워크스테이션
JOB	작업
JOBD	작업 설명
JOBE	작업 항목
JOBQ	작업 대기행렬
JOBQE	작업 대기행렬 항목
JRN	저널
JRNRCV	저널 리시버
JS	작업 스케줄러
JVA	Java
JVM	JVM(Java Virtual Machine)
KBD	키보드
KEY	키
L(접두어)	리스트
LAN	근거리 통신망(LAN)
LANG	언어
LBL	레이블
LCK	잠금
LCL	로컬

명령어 약어	의미
LCLA	로컬 속성
LF	논리 파일
LFM	논리 파일 멤버
LIB	라이브러리
LIBL	라이브러리 리스트
LIBM	라이브러리 멤버
LIC	사용권
LIN	회선
LIND	회선 설명
LNK	링크
LOC	위치
LOCALE	로케일
LOD	로드
LOF	논리 광 파일
LOG	기록부
LOGE	기록부 항목
LPD	라인 프린터 디먼
LPDA®	링크 문제점 판별 지원
LPR	라인 프린터 리퀘스터
LWS	로컬 워크스테이션
M(접두어)	멤버
MAC	메세지 인증 코드
MAIL	메일
MAP	맵
MAX	최대
MBR	멤버
MDL	모델
MED	매체
MEDDFN	매체 정의
MEDI	매체 정보
MFS	마운트 파일 시스템
MGD	관리
MGR	마이그레이트, 마이그레이션, 관리자
MGTCOL	관리 콜렉션
MLB	매체 라이브러리
MLM	매체 라이브러리 매체
MNT	분, 유지보수
MNU	메뉴
MOD	모드, 모듈
MODD	모드 설명
MON	모니터
MOV	이동
MRG	병합
MSF	메일 서버 구조
MSG	메세지
MSGD	메세지 설명
MSGF	메세지 파일
MSGQ	메세지 대기행렬
MST	마스터
M36	AS/400 Advanced 36® 기계
M36CFG	AS/400 Advanced 36 기계 구성
NAM	이름
NCK	별명
NDSCTX	NetWare 디렉토리 서비스 문맥

명령어 약어	의미
NET	네트워크
NETF	네트워크 파일
NFS	네트워크 파일 시스템
NODGRP	노드 그룹
NODL	노드 리스트
NTB	netbios
NTF	NetFinity
NTS	주
NTW	NetWare
NWI	네트워크 인터페이스
NWS	네트워크 서버
NWSAPP	네트워크 서버 어플리케이션
NWSD	네트워크 서버 설명
OBJ	오브젝트
OCL	연산 제어 언어
OF	광 파일
OFC	오피스
OFF	끄기
OMC	오브젝트 관리 주기
OPC	opticonnect
OPN	열기
OPT	광
ORD	순서
OUT	송신, 출력
OUTQ	출력 대기행렬
OUTQD	출력 대기행렬 설명
OVL	오버레이
OVLU	오버레이 유틸리티
OVR	대체
OWN	소유자
PAG	페이지, 페이지 지정
PAGDFN	페이지 정의
PAGS	페이지 세그먼트
PAGSEG	페이지 세그먼트
PARM	매개변수
PART	부분
PASTHR	pass through
PC	퍼스널 컴퓨터
PCD	pc 문서
PCL	프로토콜
PCO	PC Organizer
PCY	정책
PDF	PDF(Portable Document Format)
PDG	인쇄 설명자 그룹
PDM	프로그래밍 개발 관리자
PEX	성능 탐색기
PF	실제 파일
PFD	인쇄 출력 형식 정의
PFM	실제 파일 멤버
PFR	성능
PFRG	성능 그래픽
PFRT	성능 분석 툴
PFU	인쇄 형식 유틸리티
PFX	접두부

명령어 약어	의미
PGM	프로그램
PGP	1차 그룹
PGR	호출기
PHS	단계
PIN	개인용 식별 번호
PJ	사전시작 작업
PJE	사전시작 작업 항목
PKG	패키지
PLI	PL/I
PMN	허용
PMT	프롬트
PNLGRP	패널 그룹
POF	실제 광 파일
POL	풀
POP	포스트 오피스 프로토콜
PORT	포트
POS	위치
PPP	지점 간 프로토콜
PRB	문제점
PRC	프로시듀어
PRD	제품
PRF	프로파일
PRFL	프로파일 리스트
PRJ	프로젝트
PRM	승격
PRP	준비
PRS	퍼스널
PRT	인쇄
PRTF	프린터 파일
PRTQ	인쇄 대기행렬
PSFCFG	인쇄 서비스 기능 구성
PTC	휴대용 트랜잭션 컴퓨터
PTF	프로그램 임시 수정
PTP	지점 간
PTR	포인터
PVD	제공자
PWD	암호
PWR	전원
PYM	지불
QM	조회 관리
QRY	조회
QRYF	조회 파일
QSH	Qshell 인터프리터
QST	질문
RBD	리빌드
RCD	레코드
RCL	재생
RCV	수신
RCY	회복
RDAR	보고서/자료 아카이브 및 검색
RDB	관계형 데이터베이스
RDR	판독기
REF	참조
REG	등록

명령어 약어	의미
REX	REXX(재구성 확장 실행 언어)
RGP	순위 그룹
RGPE	순위 그룹 항목
RGZ	재구성
RJE	rje
RLS	릴리스
RLU	보고서 배치 유틸리티
RMC	보고서 관리 주기
RMT	리모트
RMV	제거
RNM	이름 변경
ROLL	화면이동
RPC	리모트 프로시듀어 호출
RPDS	VM/MVS 브릿지(이전의 리모트 스풀링 통신 서브시스템(RSCS)/PROFS 분배 서비스)
RPG	보고서 프로그램 생성기
RPL	대체
RPT	보고서
RPY	응답
RPYL	응답 리스트
RQS	요구
RRT	재라우트
RSC	자원
RSI	리모트 시스템 정보
RSM	재개
RST	복원
RTD	RouteD(TCP/IP)
RTE	라우트 항목
RTGE	라우팅 항목
RTL	유통
RTLF	유통 파일
RTN	리턴
RTV	검색
RUN	실행
RVK	취소
RWS	리모트 워크스테이션
RXC	REXEC(리모트 실행)
SAV	저장
SAVF	저장 파일
SAVRST	저장 및 복원
SBM	제출
SBS	서브시스템
SBSD	서브시스템 설명
SCD	스케줄
SCDE	스케줄 항목
SCHIDX	탐색 색인
SCHIDXE	탐색 색인 항목
SCN	화면
SDA	화면 설계 유틸리티
SDLC	동기 자료 링크 제어
SEC	보안
SET	설정
SEU	소스 입력 유틸리티
SFW	소프트웨어

명령어 약어	의미
SHD	새도우
SHRPOOL	공유 풀(pool)
SIGN	부호
SIT	상황
SLT	선택
SLTE	선택 항목
SMG	시스템 관리자
SMW	시스템 관리자 워크스테이션
SMTP	단순 우편 전송 프로토콜
SNA	시스템 네트워크 구조
SND	송신
SNI	IPX를 통한 SNA
SNMP	단순 네트워크 관리 프로토콜
SNPT	SNA pass through
SNUF	SNA 업라인 기능
SOC	제어부
SPADCT	철자법 지원 사전
SPL	스풀링
SPLF	스풀 파일
SPT	지원
SPTN	지원 네트워크
SQL	구조화 조회 언어
SRC	소스
SRCF	소스 파일
SRV	서비스
SRVPGM	서비스 프로그램
SSN	세션
SSND	세션 설명
SST	시스템 서비스 툴
STC	통계
STG	기억장치
STGL	기억장치 링크
STM	스트림, 명령문
STR	시작
STS	명령문 상태
SVR	서버
SWA	활동 중 저장
SWL	중단 단어 리스트
SYS	시스템
SYSCTL	시스템 제어
SYSDIR	시스템 디렉토리
SYSVAL	시스템 값
S34	System/34
S36	System/36
S38	System/38
TAP	테이프
TAPF	테이프 파일
TBL	표
TBLE	표 항목
TCP	TCP/IP(전송 제어 프로토콜/인터넷 프로토콜)
TDLC	쌍축 자료 링크 제어
TELN	텔넷
TFR	전송
TFTP	단순 파일 전송 프로토콜

명령어 약어	의미
THD	스레드
THLD	임계값
TIE	기술 정보 교환
TIEF	타이 파일
TIMZON	시간대
TNS	트랜잭션
TO	to
TOS	서비스 유형
TPL	템플리트
TRC	추적
TRG	트리거
TRN	토큰링 네트워크
TRP	트랩
TXT	텍스트
TYPE	유형
T1	전송 클래스 1
UBC	Ultimedia 비즈니스 회의
UDFS	사용자 정의 파일 시스템
UPD	갱신
UPG	업그레이드
USF	Ultimedia 시스템 기능
USG	사용
USR	사용자
USRIDX	사용자 색인
USRPRF	사용자 프로파일
USRPRTI	사용자 인쇄 정보
USRQ	사용자 대기행렬
USRSPC	사용자 공간
VAL	값
VAR	변수
VFY	확인
VLDL	유효성 확인 리스트
VOL	볼륨
VRY	변환
VT	VT100 또는 VT220
VWS	가상 워크스테이션
WAIT	대기
WLS	무선
WNT	Windows NT
WP	워드 프로세싱
WRK	작업
WSE	워크스테이션 항목
WSG	워크스테이션 게이트웨이
WSO	워크스테이션 오브젝트
WTR	출력기
X25	X.25

CL 명령어 키워드 및 약어

각 명령어 매개변수에는 그것에 연관된 키워드명이 있습니다. 키워드명에는 10자까지 사용할 수 있습니다. 키워드명은 명령어 동사로 시작할 필요가 없는 경우 명령어와 같은 스타일을 따르지 않습니다. 가능하면 하나의 단어나 약어를 사용하십시오. 예를 들어 CL 명령어의 공통 키워드명에는 OBJ(오브젝트), LIB(라이브러리), TYPE, OPTION, TEXT, AUT(권한) 등이 있습니다.

매개변수를 설명하기 위해 하나 이상의 단어나 약어를 사용할 경우 키워드명을 구성하십시오. 키워드명은 표준 명령어 약어와 축약되지 않은 짧은 단어의 조합을 사용하여 구성하십시오. 예를 들어 OBJTYPE은 짧은 단어 'TYPE'과 약어 'OBJ'를 결합시킨 공통 키워드명입니다.

키워드명의 두 가지 기본적인 목표는 같은 기능을 제공하는 명령들 사이에서 인식될 수 있도록 하는 것과 일관성이 유지되도록 하는 것입니다. 간단한 단어와 표준 약어를 사용하여 키워드명이 인식되도록 하는 데 도움을 줄 수 있습니다.

다음은 CL 명령어 매개변수 키워드명에 사용되는 약어의 리스트입니다.

키워드 약어	의미
A(접두어)	속성, 활동, 주소
ABS	추상, 절대
ABN	비정상
AC	자동 호출
ACC	액세스, 액세스 코드
ACCMTH	액세스 방식
ACG	계정
ACK	수신확인
ACN	조치
ACP	허용
ACQ	확보
ACSE	관련 제어 서비스 요소
ACT	활동, 활동 중, 활성화, 조치
ACTSNBU	교환 네트워크 백업 활성화
ADDR(또는 ADR)	주소
ADJ	인접, 조정
ADL	추가
ADM	관리, 어플리케이션 개발 관리자
ADMD	관리 정의역
ADP	허용, 허용하는
ADPT	어댑터
ADR(또는 ADDR)	주소
ADV	진행
AFN	유사성
AIX	AIX 오퍼레이팅 시스템
AJE	자동시작 작업 항목
AFP	확장 기능 인쇄
ALC	할당
ALM	경보
ALR	경고

키워드 약어	의미
ALRD	경고 설명
ALS	별명
ALW	허용
ANET	인접 네트워크 엔티티 제목
ANS	응답
ANZ	분석
AP	액세스 경로
APAR	APAR
APF	확장 프린터 기능
APPP	어플리케이션 프로세스
APW	확장 인쇄 출력기
APP	어플리케이션
APPC	APPC(advanced program-to-program communications)
APPN	APPN(advanced peer-to-peer networking)
APY	적용
ARA	영역
ARP	주소 해결 프로토콜
ASC	비동기 통신
ASCII	미국 표준 정보 교환 코드
ASMT	할당
ASN	할당, 연관
ASP	보조 기억장치 풀
AST	지원
ASYNC	비동기
ATD	유인
ATN(또는 ATTN)	어텐션
ATR(또는 ATTR)	속성
ATTACH	접속
ATTN(또는 ATN)	어텐션 키
ATTR(또는 ATR)	속성
AUD	감사
AUT	권한, 권한이 있는 권한 부여
AUTL	권한 부여 리스트
AUTO	자동
AUX	보조
AVG	평균
AVL	사용 가능
BAL	균형
BAS	BASIC 언어, 기본
BCD	바코드, 브로드캐스트 자료
BCH	일괄처리
BCKLT	백라이트
BCKUP(또는 BKU)	백업
BDY	경계
BEX	분기 확장자
BGU	업무용 그래픽 유틸리티
BIN	2진
BIO	블록 입/출력
BITS	자료 비트
BKP	중단점
BKT	대괄호, 중단
BKU(또는 BCKUP)	백업
BLDG	빌드

키워드 약어	의미
BLK	블록
BLN	깜박임 커서
BND	바인딩, 바인드
BNR	배너
BOT	맨 아래
BRK	일시 중단
BSC	2진 동기 통신
BSCEL	2진 동기 통신 동등 링크
BSP	백스페이스
BUF	버퍼
C	C 언어
CAB	캐비넷
CAL	캘린더
CAP	용량, 캡처
CB	취소
CBL	COBOL 언어
CCD	호출 제어 자료
CCSID	코드화 문자 세트 ID
CCT	회로
CDE	코드
CDR	호출 상세 레코드
CFG	구성
CFGL	구성 리스트
CFGLE	구성 리스트 항목
CFM	확인, 확정
CGU	문자 생성 유틸리티
CHAR(또는 CHR)	문자
CHG	변경
CHK	검사
CHKSUM	체크섬
CHKVOL	검사 볼륨 ID
CHL	채널
CHR(또는 CHAR)	문자
CHRSTR	문자 스트링
CHT	도표
CGY	범주
CKR	체커
CKS	체크섬
CL	제어 언어
CLG	카탈로그
CLN	지우기, 클린업
CLNS	비연결 모드 네트워크 서비스
CLNUP(또는 CLN)	클린업
CLO	닫기
CLR	지우기
CLS	클래스
CLSF	클래스 파일
CLT	클라이언트
CLU	클러스터
CMD	명령
CMN	통신
CMNE	통신 항목
CMP	비교
CMT	확약, 주석

키워드 약어	의미
CNG	혼잡
CNL	취소
CNLMT(또는 CNNLMT)	연결 한계
CNN	연결
CNNL	연결 리스트
CNNLMT	연결 한계
CNR	컨테이너
CNS	상수
CNT	접속
CNTL(또는 CTL)	제어
CNTRY	국가
CNV	대화
COD	코드
CODPAG	코드 페이지
CODE	코드, 공동 개발 환경
COL	열, 수집, 콜렉션
COM	공통, 커뮤니티
COMPACT	컴팩트, 컴팩션
CON	기밀
CONCAT	연결
COND	조건
CONS	연결 모드 네트워크 서비스
CONTIG	인접
CONT	계속
COS	서비스 클래스
COSD	서비스 클래스 설명
COSTBYTE	바이트당 비용
COSTCNN	연결당 비용
COVER	겉표지
CP	제어점
CPB	변경 프로파일 분기, 호환성
CPI	인치당 문자 수
CPL	완료
CPP	C++ 언어
CPR	압축
CPS	호출 진행 신호
CPT	구성요소
CPU	중앙 처리 장치
CPY	복사
CPYRGT	저작권
CRC	순환 중복 검사
CRDN	기밀
CRG	비용, 클러스터 자원 그룹
CRL	상관
CRQ	변경 요구
CRQD	변경 요구 설명
CRSDMNK	정의역 간 키
CRT	작성
CSI	통신측 정보
CSL	콘솔
CSR	커서
CST	제한사항, 비용
CTG	카트리지
CTL	제어기, 제어

키워드 약어	의미
CTLD	제어기 설명
CTN	경합
CTS	송신을 위해 지우기
CTX	문맥
CUR	현재
CVN	변환
CVR	커버
CVT	변환
CXT(또는 CTX)	문맥
CYC	주기
D(접두어)	설명
DAP	디렉토리 액세스 프로토콜
DAT	날짜
DB	데이터베이스
DBCS	2바이트 문자 세트
DBF	데이터베이스 파일
DBG	디버그
DBR	데이터베이스 관계
DCE	자료 통신 장비, 분산 컴퓨팅 환경
DCL	선언
DCP	압축 해제
DCT	사전
DDI	분산 자료 인터페이스
DDM	분산 자료 관리
DDMF	분산 자료 관리 파일
DDS	자료 서술 스펙
DEC	10진
DEGREE	병렬 처리 정도
DEL	전달
DEP	종속
DEPT	부서
DES	자료 암호화 표준
DEST	목적지
DEV	장치
DEVD	장치 설명
DFN	정의
DFR	연기
DFT	디폴트
DFU	자료 파일 유틸리티
DIAG	대화
DIF	차이점
DIFF	차별화
DIR	디렉토리
DKT	디스켓
DLC	할당 해제
DLO	문서 라이브러리 오브젝트
DLT	삭제
DLVRY	전달
DLY	지연
DMN	정의역
DMP	덤프
DN	식별명
DOC	문서
DPR	DataPropagator Relational

키워드 약어	의미
DRAWER	드로어
DRT	직접
DRV	드라이브
DSA	디렉토리 시스템 에이전트
DSAP	목적지 서비스 액세스점
DSB	작동 불가능
DSC	단절
DSK	디스크
DSP	표시
DST	일광 절약 시간, 전용 서비스 툴, 분배
DTA	자료
DTE	자료 단말 장치
DTL	상세
DUP	사본
DVL	개발
DWN	다운
E(접두어)	입력
EBCDIC	확장 2진 코드화 십진 교환 코드
ECN	명시적 혼잡 통지
EDT	편집
EDU	교육
EIM	기업망 ID 맵핑
EJT	방출
ELAN	이더넷 LAN
ELEM	요소
ELY	조기
EML	에뮬레이트, 에뮬레이션
ENB	작동 가능
ENC	코드화
ENR	등록
ENT	입력
ENV	환경
EOF	파일 끝
EOR	레코드 끝
EOV	볼륨 끝
ERR	오류
EST	설정
ETH	이더넷
EVT	이벤트
EXC	제외
EXCH	교환
EXD	확장
EXEC	실행
EXIST	존재
EXN	확장
EXP	만료, 만기
EXPR	표현식
EXT	추출, 확장
F(접두어)	파일
FAIL	실패
Fnn	기능 키 ‘nn’
FA	파일 속성
FAX	팩시밀리
FCL	기능

키워드 약어	의미
FCN	기능, 기능적
FCT	양식 제어표
FCTE	양식 제어표 항목
FEA	프런트 엔드 어플리케이션
FEA	프런트 엔드 어플리케이션
FEAT	피처
FID	파일 ID
FIL	파일
FLD	필드
FLG	플래그
FLIB	파일 라이브러리
FLR	폴더
FLW	흐름
FLV	실패 시 전환
FMA	폰트 관리 지원
FMT	형식
FNC	재무관리
FNT	폰트
FORMDF	양식 정의
FP	중재점
FRAC	분수
FRC	강제
FRI	금요일
FRM	from, 프레임
FRQ	빈도
FSC	회계
FSN	파일 순번
FST	맨 처음
FTAM	파일 전송, 액세스, 관리
FTP	파일 전송 프로토콜(TCP/IP)
FTR	필터
GC	가비지 콜렉션
GCH	가비지 콜렉션 힙
GDF	그래픽 자료 파일
GDL	지침
GEN	생성
GID	그룹 ID 번호
GIV	제공
GLB	글로벌
GNL	일반
GPH	그래프
GRP	그룹
GRT	부여
GSS	그래픽 기호 세트
GVUP	포기
HCP	호스트 명령 프로세서
HDL(또는 HNDL)	핸들
HDR	헤더
HDW	하드웨어
HEX	16진
HFS	계층 파일 시스템
HLD	보류
HLL	고급 언어
HLP	도움말

키워드 약어	의미
HLR	보유자
HNDL(또는 HDL)	핸들
HPCP	호스트 대 프린터 코드 페이지
HPFCS	호스트 대 프린터 폰트 문자 세트
HPR	고성능 라우팅
HRZ	수평
HST	이력
HTML	하이퍼텍스트 마크업 언어
HTTP	하이퍼텍스트 전송 프로토콜(TCP/IP)
I(접두어)	정보, ILE
ICF	시스템 내 통신 기능
ICV	초기 체인값
ID	식별자
IDD	대화식 자료 정의
IDL	유휴
IDLC	통합 자료 링크 제어
IDP	교환 문서 프로파일
IDX	색인
IE	정보 요소
IFC	인터페이스
IGC	ideographic(2바이트 문자 세트)
IGN	무시
IFC	인터페이스
ILE	통합 언어 환경
IMG	이미지
IN	입력
INAC(또는 INACT)	비활동
INACT(또는 INAC)	비활동
INC	포함
IND	간접
INF	정보
INFSKR	Infoseeker
INH	방해
INIT	개시
INL	초기
INM	중간
INP	입력
INPACING	인바운드 페이싱
INQ	조회
INS	설치
INST	인스턴스
INT	대화식, 정수, 내부
INTNET	인터넷
INTNETA	인터넷 주소
INTR	시스템 내
INV	회의 참석자, 명세, 호출
INZ	초기화
IP	인터넷 프로토콜
IPDS	지능형 프린터 자료 스트림
IPI	IPX를 통한 IP
IPL	초기 프로그램 로드
IPX	인터넷 패킷 교환
ISDN	종합 정보 통신망
ISP	인터넷 서비스 제공자

키워드 약어	의미
IT	중간 텍스트, 내부 텍스트
ITF	대화식 단말 기능
ITM	항목
ITV	간격
IW2	IPX WAN 버전 2 프로토콜
J(접두어)	작업
JDFT	결합 디폴트
JE(접두어)	작업 항목
JFLD	결합 필드
JORDER	결합 파일 순서
JRN	저널
JRNRCV	저널 리시버
JVA	Java
KBD	키보드
KNW	지식
KPF	간지(kanji) 프린터 기능
KWD	키워드
L(접두어)	리스트
LADN	라이브러리 할당 문서명
LAN	근거리 통신망(LAN)
LANG(또는 LNG)	언어
LBL	레이블
LCL	로컬
LCLE	로컬 위치 항목
LCK	잠금
LDTIME	조달 시간
LE(접두어)	리스트 항목
LEC	LAN 에뮬레이션 클라이언트
LECS	LAN 에뮬레이션 구성 서버
LEN	길이
LES	LAN 에뮬레이션 서버
LF	논리 파일
LFM	논리 파일 멤버
LFT	왼쪽
LGL	논리
LIB	라이브러리
LIBL	라이브러리 리스트
LIC	사용권, 사용권이 부여됨, 사용권 내부 코드
LIFTM	활동 시간
LIN	회선, 회선 설명
LMI	로컬 관리 인터페이스
LMT	한계
LNG(또는 LANG)	언어
LNK	링크
LNR	리스너
LOC	위치
LOD	로드
LPI	인치당 행 수
LRC	수평 중복 검사
LRG	대형
LRSP	로컬 응답
LST	리스트, 최종
LTR	문자
LVL	레벨

키워드 약어	의미
LWS	로컬 워크스테이션
LZYWRT	지연 기록
M(접두어)	멤버, 메세지
MAC	매크로, 매체 액세스 제어
MAINT	유지보수
MAJ	주
MAP	맵, 생산 자동화 프로토콜
MAX	최대
MBR	멤버
MBRS	멤버
MCA	메세지 채널 에이전트
MCH	기계
MDL	모델
MDM	모뎀
MDTA	메세지 자료
MED	매체
MEDI	매체 정보
METAFILE	메타표 파일
MFR	제조업체
MFS	마운트 파일 시스템
MGR	관리자
MGRR	관리자 등록
MGT	관리
MID	보통
MIN	최소, 최소화
MLB	매체 라이브러리 장치
MLT	승수, 복수
MM	멀티미디어
MNG	관리
MNT	유지보수, 마운트
MNU	메뉴
MOD	모드, 모듈
MODD	모드 설명
MON	모니터, 월요일
MOV	이동
MQM	메세지 대기행렬 관리자
MRG	병합
MRK	마크
MRT	복수 리퀘스터 단말기
MSF	메일 서버 구조
MSG	메세지
MSGS	메세지
MSGQ	메세지 대기행렬
MSR	측정
MSS	관리 시스템 서비스
MST	마스터
MTD	마운트
MTG	회의
MTU	최대 전송 단위
MTH	방법
MULT(또는 MLT)	복수
M36	AS/400 Advanced 36 기계
M36CFG	AS/400 Advanced 36 기계 구성
N(접두어)	이름, 네트워크

키워드 약어	의미
NAM	이름
NBR	번호
NCK	별명
NDE	노드
NDM	정상 단절 모드
NDS	Netware 디렉토리 서비스
NEG	부정, 조정
NEP	무한 수행 프로그램
NET	네트워크
NFY	통지
NL	네트워크층
NLSP	NetWare 링크 서비스 프로토콜
NML	이름 리스트
NNAM(또는 NCK)	별명
NOD	노드
NODL	노드 리스트
NORM	정상
NOTVLD	유효하지 않은
NPRD	비생산적
NRM	정상, 정상 응답 모드
NRZI	0으로 다시 변환하지 않음
NT	네트워크 종료
NTB	NetBIOS
NTBD	NetBIOS 설명
NTC	주의
NTF	NetFinity
NTP	네트워크 시간 프로토콜
NTS	주
NTW	NetWare
NTW3	NetWare 3.12
NUM	숫자, 번호
NWI	네트워크 인터페이스
NWID	네트워크 인터페이스 설명
NWS	네트워크 서버
NWSD	네트워크 서버 설명
NXT	다음
OBJ	오브젝트
OBS	관찰할 수 있는 정보
OFC	오피스
OFSET	오프셋
OMT	생략
OPN	열기
OPR	연산자, 연산
OPT	옵션, 광, 최적
ORD	순서
ORG	조직
ORGUNIT	조직 단위
OS	오퍼레이팅 시스템
OSDB	오브젝트 저장 데이터베이스
OUT	출력
OVF	넘침
OVL	오버레이
OVR	대체
OVRFLW	넘침

키워드 약어	의미
OWN	소유자, 소유
PAD	패킷 어셈블리/디어셈블리
PAG	페이지, 페이지 지정
PAL	제품 활동 기록부
PARM	매개변수
PASTHR	passthru
PBL	예상할 수 있는
PBX	사설 구내 교환
PC	퍼스널 컴퓨터
PCD	PC 문서
PCL	프로토콜
PCO	PC Organizer
PCS	퍼스널 컴퓨터 지원
PCT	퍼센트
PCTA	퍼스널 컴퓨터 텍스트 지원
PCY	정책
PDF	PDF(Portable Document Format)
PDG	인쇄 설명자 그룹
PDM	프로그래밍 개발 관리자
PDU	프로토콜 자료 단위
PENWTH	펜 너비
PERS	퍼스널
PF	실제 파일
PFnn	프로그램 기능 키 'nn'
PFD	인쇄 출력 형식 정의
PFM	실제 파일 멤버
PFVLM	실제 파일 변수 길이 멤버
PFR	성능
PFX	접두부
PGM	프로그램
PGP	1차 그룹
PGR	호출기
PHCP	프린터 대 호스트 코드 페이지
PHFCS	프린터 대 호스트 폰트 문자 세트
PHS	단계
PHY	실제
PIN	개인용 식별 번호
PJE	사전시작 작업 항목
PKA	공용 키 알고리즘
PKG	패키지
PKT	패킷
PL	표시층
PLC	위치
PLL	폴, 폴링
PLT	플로터
PMN	허용
PMP	단일 지점 대 복수 지점
PMT	프롬트
PND	지연중
PNL	패널
PNT	점
POL	풀
POLL	폴, 폴링
POP	포스트 오피스 프로토콜(TCP/IP)

키워드 약어	의미
PORT	포트 번호
POS	긍정, 위치
PPP	지점 간 프로토콜
PP	프리프로세서
PPR	용지
PPW	페이지 프린터 출력기
PRB	문제점
PRC	프로시듀어, 프로세스
PRD	제품, 생산적
PRJ	프로젝트
PREBLT	사전빌드
PRED	선행 프로그램
PREEST	사전설정
PREF	기본설정, 우선
PREOPR	사전조작
PREREQ	전제조건
PRF	프로파일
PRI	1차
PRJ	프로젝트
PRM	승격, 매개변수
PRMD	개별 관리 정의역
PRN	모
PRO	제안
PROC(또는 PRC)	프로시듀어, 처리
PROD	실행
PROP	특성
PRP	준비, 전파
PRS	퍼스널
PRT	인쇄, 프린터
PRTQ	인쇄 대기행렬
PRV	이전
PRX	프록시
PSAP	표시층 서비스 액세스점
PSF	인쇄 서비스 기능
PSN	표시
PSTOPR	사후조작
PTC	보호, 휴대용 트랜잭션 컴퓨터
PTF	프로그램 임시 수정
PTH	경로
PTL	부분
PTN	파티션, 분할
PTP	지점 간
PTR	포인터
PTY	우선순위
PUB	공용
PUNS	천공
PVC	영구 가상 회로
PVD	제공자
PVT	사설
PWD	암호
PWR	전원
Q(접두어)	대기행렬
QE(접두어)	대기행렬 항목
QLTY	품질

키워드 약어	의미
QOS	서비스 품질
QRY	조회
QST	질문
QSTDB	질의응답 데이터베이스
QSTLOD	질의응답 로드
QUAL	규정자
RAR	라우트 추가 저항
RBD	리빌드
RCD	레코드
RCDS	레코드
RCL	재생
RCMS	리모트 변경 관리 서버
RCP	수신자
RCR	순환
RCV	수신
RCY	회복
RDB	관계형 데이터베이스
RDN	상대 식별명
RDR	판독기
RDRE	판독기 항목
REACT	재활성화
REASSM	리어셈블리
REC	레코드
RECNN	재연결
REF	참조
REINZ	재초기화
REL	관계, 해제
REP	대표, 영업대표
REQ(또는 RQS)	필수, 요구, 리퀘스터
RES	상주, 해상도
RESYNC	재동기화
RET	보유
REX	REXX 언어
RFS	거절
RGS	등록
RGT	오른쪽
RGZ	재구성
RINZ	재초기화
RIP	라우팅 정보 프로토콜
RJE	리모트 작업 항목
RJT	거부
RLS	릴리스
RMD	메모, 통지
RMT	리모트
RMV	제거
RNG	범위
RNM	이름 변경
RPG	RPG 언어
RPL	대체
RPT	보고서
RPY	응답
RQS(또는 REQ)	요구, 리퀘스터
RQT	필요조건
RRSP	리모트 응답

키워드 약어	의미
RRT	재라우트
RSB	리어셈블리
RSC	자원
RSL	결과 해상도
RSM	재개
RSP	응답
RSRC	자원
RST	복원
RSTD	제한
RTE	라우트
RTG	라우팅
RTL	유통
RTM	재전송
RTN	리턴, 재전송
RTR	라우터
RTT	회전
RTV	검색
RTY	재시도
RU	요구 단위
RVK	취소
RVS	반전
RWS	리모트 워크스테이션
SAA®	시스템 어플리케이션 구조
SADL	새들(saddle)
SAP	서비스 액세스점
SAT	토요일
SAV	저장
SAVF	저장 파일
SBM	제출
SBS	서브시스템
SCD	스케줄
SCH	탐색
SCN	화면
SCT	섹터
SDLC	동기 자료 링크 제어
SDU	서비스 자료 단위
SEC	두 번째, 보안, 보안
SEG	세그먼트
SEGMENT	세그먼트화
SEL(또는 SLT)	선택
SENSITIV	민감성
SEP	분리자
SEQ	순서, 순차적
SEV	심각도
SFW	소프트웨어
SGN	사인 온
SHD	새도우
SHF	시프트
SHM	단기 보류 모드
SHR	공유
SHUTD	시스템 종료
SI	시프트 인
SIG	서명, 서명된
SIGN	사인 온

키워드 약어	의미
SIZ	크기
SL	세션층
SLR	선택기
SLT(또는 SEL)	선택
SMAE	시스템 관리 어플리케이션 엔티티
SMG	시스템 관리자
SMTP	단순 우편 전송 프로토콜
SMY	요약
SNA	시스템 네트워크 구조
SNBU	교환 네트워크 백업
SND	송신
SNG	단일
SNI	IPX를 통한 SNA
SNP	스냅
SNPT	SNA pass-through
SNUF	SNA 업라인 기능
SO	시프트 아웃
SOC	제어부
SPA	철자법 지원
SPC	공간, 특수
SPD	제공
SPF	특정
SPID	서비스 제공자 ID
SPL	스풀, 스풀링
SPR	대치
SPT	지원
SPTN	지원 네트워크
SPX	순차 패킷 교환
SQL	구조화 조회 언어
SRC	소스
SRCH(또는 SCH)	탐색
SRM	시스템 자원 관리
SRQ	시스템 요구
SRT	정렬
SRV	서비스
SSAP	소스 서비스 액세스점, 세션층 서비스 액세스점
SSCP	시스템 서비스 제어점
SSL	보안 소켓층
SSN	세션
SSND	세션 설명
SSP	일시중단
SST	시스템 서비스 툴
STAT	통계 자료 레코드
STATION	편의 스테이션
STC	통계
STD	표준
STG	기억장치
STK	스택
STM	스트림
STMF	스트림 파일
STMT	상태
STN	스테이션
STP	단계
STPL	스테이플

키워드 약어	의미
STR	시작
STS	명령문 상태
STT	상태
STX	텍스트 시작 문자
SUB	대체, 주제
SUBADR	부속 주소
SUBALC	부속 할당
SUBDIR	서브디렉토리
SUBFLR	서브폴더
SUBNET	서브네트워크
SUBPGM	서브프로그램
SUBST	대체
SUCC	상속자
SUN	일요일
SURNAM	성
SVC	교환 가상 회로
SVR	서버
SWL	중단 단어 리스트
SWS	스위치
SWT	스위치, 교환
SWTSET	스위치 설정
SYM	기호
SYN	구문
SYNC	동기
SYS	시스템
SYSLIBL	시스템 라이브러리 리스트
S36	System/36
TAP	테이프
TAPDEV	테이프 장치
TBL	표
TCID	전송 연결 ID
TCP	TCP/IP(전송 제어 프로토콜/인터넷 프로토콜)
TCS	전화 연결 서비스
TDC	전화 자료 콜렉터
TDLC	쌍축 자료 링크 제어
TEID	단말기 종료점 ID
TEL	전화
TELN	TELNET(TCP/IP)
TERM	단말기
TFR	전송
TGT	목표
THD	스레드
THLD	임계값
THR	통해, 처리량
THRPUT	처리량
THS	사전
THU	목요일
TIE	기술 정보 교환
TIM	시간
TIMMRK	타임마크
TIMO	시간종료
TIMOUT(또는 TIMO)	시간종료
TIMZON	시간대
TKN	토큰

키워드 약어	의미
TL	전송층
TM	시간
TMN	전송
TMP	임시
TMPL(또는 TPL)	템플리트
TMR	타이머
TMS	전송
TMT	전송
TNS	트랜잭션
TOKN(또는 TKN)	토큰
TOT	총계
TPDU	전송층 프로토콜 자료 단위
TPL	템플리트, 토폴로지
TPT	전송
TRANS	임시, 트랜잭션
TRC	추적
TRG	트리거
TRM	기간
TRN	토큰링 네트워크, 변환
TRNSPY	투명성
TRP	트랩
TRS	임시
TRUNC	절단
TSE	시간 분할 끝
TSP	시간소인
TSAP	전송층 서비스 액세스점
TSK	타스크
TST	테스트
TUE	화요일
TWR	타워
TXP	전송
TXT	텍스트
TYP	유형
T1	전송 클래스 1
T2	전송 클래스 2
T4	전송 클래스 4
UDFS	사용자 정의 파일 시스템
UDP	사용자 데이터그램 프로토콜
UI	사용자 인터페이스, 번호를 지정하지 않은 정보
UID	사용자 ID 번호
UNC	분류하지 않은
UNL	링크되지 않음
UNPRT	인쇄할 수 없는
UNQ	고유
UOM	측정 단위
UPCE	범용 제품 코드 유형 E 바코드
UPD	갱신
UPG	업그레이드
URL	단일 자원 로케이터
USG	사용
USR	사용자
VAL	값
VAR	변수
VCT	가상 회로

키워드 약어	의미
VDSK	가상 디스크
VER(또는 VSN)	버전
VFY	확인
VLD	유효, 유효성, 유효성 확인
VND	밴더
VOL	볼륨
VRF	검증
VRT	가상
VRY	변환
VSN(또는 VER)	버전
VWS	가상 워크스테이션
WAN	광역 네트워크
WCH	감시
WDW	창
WED	수요일
WIN	승자
WK	약
WNT	Windows NT
WP	워드 프로세싱
WRD	워드
WRK	작업
WRT	기록
WS	워크스테이션
WSC	워크스테이션 제어기
WSCST	워크스테이션 사용자 정의 오브젝트
WSE	워크스테이션 항목
WSG	워크스테이션 게이트웨이(TCP/IP)
WSO	워크스테이션 오브젝트
WTR	출력기
WTRE	출력기 항목
WTRS	출력기
X25	X.25
X31	X.31
X400	X.400
3270	3270 표시 화면

부록 E. 주의사항

이 정보는 미국에서 제공되는 제품과 서비스용으로 작성된 것입니다. IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 IBM 이외의 제품, 프로그램 또는 서비스를 사용할 때 그 작동을 평가 및 검증하는 것을 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2 바이트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적인 보증을 포함하여 (단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증 없이 이 책을 『현 상태대로』 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 이 변경사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통고없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

(1) 독립적으로 작성된 프로그램과 기타 프로그램(본 프로그램 포함) 간의 정보 교환 및 (2) 교환된 정보의 상호 이용을 목적으로 정보를 원하는 프로그램 라이센스 사용자는 다음 주소로 문의하십시오.

| 135-270
| 서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
| 한국 아이.비.엠 주식회사
| 고객만족센터

이러한 정보는 해당 조항 및 조건에 따라(예를 들면, 사용료 지불 포함) 사용할 수 있습니다.

이 정보에 기술된 라이센스가 있는 프로그램 및 이 프로그램에 대해 사용 가능한 모든 라이센스가 있는 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위해 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 추가 비용없이 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이들 샘플 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 암시하지 않습니다. 귀하는 IBM의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 추가 비용없이 이러한 샘플 응용프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 그 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.

이 정보를 소프트카피로 보는 경우에는 사진과 컬러 삽화가 제대로 나타나지 않을 수도 있습니다.

프로그래밍 인터페이스 정보

CL 프로그래밍 문서는 고객이 CL 프로그램을 작성하는 데 사용할 수 있는 프로그래밍 인터페이스에 대해 설명합니다.

상표

다음 용어는 미국 또는 기타 국가에서 사용되는 IBM Corporation의 상표입니다.

Advanced 36
AFP
Application System/400
CICS
CICS/400
Client Access
COBOL/400
C/400
DataPropagator
DB2
DB2 Universal Database
e(로고)
IBM
Integrated Language Environment
IPDS
iSeries
Operating System/400
OS/400
RPG/400
SAA
System/36
System/38

Microsoft, Windows, Windows NT, 및 Windows 로고는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스 표입니다.

참고 문헌

OS/400 프로그래밍에 대한 자세한 정보는 다음 책과 iSeries Information Center 주제를 참조하십시오.

- Application Display Programming

  이 책은 화면 작성 및 유지보수를 위한 DDS 사용법, 화면 파일의 작성 및 화면 파일에 대한 작업, 온라인 도움말 정보의 작성, 화면 정의를 위한 UIM 사용법, 패널 그룹과 레코드 및 문서 사용법을 설명합니다.

- 백업 및 회복

  이 매뉴얼은 iSeries 서버의 회복 및 가용성 옵션에 대한 일반 정보를 제공합니다. 이 매뉴얼에서는 서버에서 사용할 수 있는 옵션을 설명하고 그러한 옵션을 비교 및 대조하며 더 자세한 정보가 있는 위치를 알려줍니다.

- iSeries Information Center의 정보 중 **시스템 관리** 범주에 있는 **백업 및 회복** 주제에는 백업 및 회복 전략을 계획하는 방법, 서버의 백업 방법, 테이프 라이브러리의 관리 방법 및 데이터의 디스크 보호를 설정하는 방법에 대한 정보가 들어 있습니다. 또한 iSeries(TM) Navigator에 대한 백업, 회복 및 미디어 서비스 플러그 인, 서버 회복에 대한 정보, 백업 및 회복에 대해 자주 묻는 질문과 그에 대한 답변도 제공합니다.

- iSeries Information Center의 **프로그래밍** 범주에 있는 **CL** 주제에서는 OS/400 명령과 기타 IBM에서 제공하는 명령을 사용하여 프로그램을 작성하는 시스템 프로그래머 및 시스템 관리자를 위한 정보를 제공합니다.

- ILE 개념

  이 책은 OS/400 사용권 프로그램의 통합 언어 환경(IEL) 구조에 관한 개념과 용어에 대해 설명합니다. 또한 모듈 작성, 바인딩, 프로그램 수행, 프로그램 디버깅 및 예외 처리 등의 주제에 대해서도 다룹니다.

- iSeries Information Center의 **데이터베이스** 범주에서는 어플리케이션 또는 사용자 인터페이스를 통해 서버 자료에 액세스 및 관리하는 것은 물론 참조 무결성 및 병렬 데이터베이스 처리와 같은 고급 기능을 제공할 수 있도록 하는 iSeries용 DB2 UDB(Universal Database™)에 대한 정보를 제공합니다.

- iSeries Information Center의 **파일 및 파일 시스템** 범주에서는 IBM iSeries 서버의 데이터베이스 파일 관리 기능 및 통합 파일 시스템 기능에 대한 정보를 제공합니다.

- iSeries Information Center의 **프로그래밍** 범주에 있는 **국제화** 주제에서는 자료 처리 관리자, 시스템 오퍼레이터 및 관리자, 어플리케이션 프로그래머, 일반 사용자, IBM 마케팅 영업대표 및 시스템 엔지니어에게 iSeries 서버의 국제화 기능을 이해하고 사용하는 데 필요한 정보를 제공합니다. 이 책을 통해 iSeries 서버의 국제화 및 복수 언어 지원 계획, 설치, 구성 및 사용을 위한 준비를 할 수 있습니다. 또한 복수 언어 자료의 데이터베이스 관리를 위한 설명과 복수 언어 시스템에 있어서의 어플리케이션 고려사항을 알 수 있습니다.

- Printer Device Programming

  이 책은 iSeries 서버의 인쇄 요소 및 개념, 프린터 파일 및 인쇄 스풀링 지원을 포함하여 프린터 연결을 위한 고유 정보를 제공합니다.

- iSeries Information Center의 **인쇄** 범주에서는 인쇄 기능을 계획하고 구성하는 방법과 기본적인 인쇄 정보를 제공합니다.

- 보안 - 참조서

이 책은 일반적인 보안 개념과 시스템 상의 보안 계획에 대해 설명합니다. 또한 자원 보안과 관련하여 모든 사용자들에게 유용한 정보를 제공합니다.

- iSeries Information Center의 **보안** 범주에서는 iSeries 보안을 설정 및 계획하는 방법, 네트워크 및 통신 어플리케이션의 보안 방법, iSeries 서버에 높은 보안 수준의 암호화 처리 기능을 추가하는 방법 등에 대한 정보를 제공합니다. 또한 오브젝트 서명 및 서명 유효성, ID 맵핑, 인터넷 보안 위험에 대비한 솔루션 등에 대한 정보도 제공합니다.
- iSeries Information Center의 **시스템 관리** 범주에서는 작업 관리 환경의 작성 및 변경, 시스템 값에 대한 작업, 시스템 성능을 향상시키기 위한 성능 자료의 수집 및 사용 등에 대한 정보를 제공합니다.
- iSeries Information Center의 **프로그래밍** 범주에 있는 **API** 주제에서는 OS/400 어플리케이션 프로그래밍 인터페이스(API)를 사용하려는 숙련된 어플리케이션 프로그래머 및 시스템 프로그래머를 위한 정보와 예를 제공합니다.

이 책에 언급된 OS/400 프로그래밍 유틸리티에 대한 자세한 정보는 다음 책을 참조하십시오.

- ADTS/400: Character Generator Utility
- ADTS/400: Programming Development Manager

- ADTS for AS/400: Screen Design Aid 
- ADTS for AS/400: Source Entry Utility 

색인

[가]

감사 저널 항목 표시(DSPAUDJRNE) 명령 132
값
  매개변수 376
갱신
  사용 정보 148
갱신 권한
  갱신 130
건너 뜀 값
  정의 454
검사
  오브젝트 20, 172
  프로그램 유효성 341
검색
  구성 상태 20, 65
  구성 소스 20, 65
  네트워크 속성 66
  라이브러리 설명 137
  메세지 20, 277
  멤버 설명 20, 69
  사용자 프로파일 20, 68
  시스템 값 20, 62
  오브젝트 설명 68, 144
  자료 영역 20, 112
  작업 속성 20, 67
  프로그램 속성 212
  프로그램 작성 명령 20
  프로파일 속성 68
  CL 프로시듀어 메세지 277
경고 ID
  지정 230
계산 20
  CHGVAR(변수 변경) 명령 16
  참조 : 표현식
고급 언어(HLL) 프로그램 195
  혼합 리스트 366
  QCMDEXC 프로그램 191
공유된 갱신 불가(*SHRNUP) 상태 164
공유된 갱신(*SHRUPD) 잠금 상태 164
관계
  명령 정의의 부분 402
  PARM문과 DCL 명령 402
관계식 49
구문
  명령 3
구문 검사 195
구성 상태
  검색 20, 65
구성 상태 검색(RTVCFGSTS) 명령 20, 65
구성 소스
  검색 20, 65
구성 소스 검색(RTVCFGSRC) 명령 20, 65
권한
  갱신 130
  라이브러리 125, 130
  명령 정의 343
  삭제 130
  새로 작성된 오브젝트에 대한 디폴트 132
  실행 130
  오브젝트 130
  오브젝트 관리 130
  오브젝트 조작 130
  오브젝트 존재 130
  읽기 130
  자료 130
  조합 130
  추가 130
  *ALL 130
  *CHANGE 130
  *EXCLUDE 130
  *USE 130
규정된 이름
  명령 정의의 예 408
  오브젝트 액세스 117
  용 구문 117
  정의 372
  지정 28
  프롬트를 사용하여 지정 117
  CL 또는 HLL 사용 373
  CL 프로그램 안의 처리 373
  CPP로 전달 373, 375
  REXX의 사용 375
규정자(QUAL)문
  사용 372
  예 372, 408
  정의 344
기능
  테스트
    설명 11
  CL 명령 20
기록
  요구 처리 프로그램 275
  CL 프로시듀어에 주석 34
  REXX 명령 처리 프로시듀어 403
기록부
  시스템 표시 331
  이력 331
  일괄처리 작업에 대한 고려사항 331
  작업 319
  화면 333
  QHST(이력) 331

[나]

날짜
  변환 20
  형식 변환 63
날짜 변환(CVTDAT) 명령 20, 63
날짜의 형식
  변환 63
내부 단계화 디버그 명령 437
내장 기능 16
내장 IF(If) 명령 41
내포
  설명 447
내포된 DO 그룹
  예 43
네트워크 속성
  검색 66
네트워크 속성 검색(RTVNETA) 명령 66
논리 제어 명령 16
논리식 49

[다]

단순 리스트
  매개변수 값
    설명 360
    정의 360
    CPP로 전달 360

단순 리스트 *(계속)*
CL 또는 HLL 사용 361
REXX의 사용 363
대기행렬
메세지 10, 236
메세지 대기행렬 전달 유형 변경 241
메세지 제거로부터 279
메세지로부터 수신 271
외부 메세지(*EXT) 242
작업 메세지 대기행렬 242
QSYSMSG 290
대기(WAIT) 명령 20, 184
대체
데이터베이스 파일 16
메세지 파일 232
대체 변수 223
대화식
입력 15
작업
디버그 모드 464
작업 기록부
고려사항 330
데이터베이스 파일 187
대체 16
자료 대기행렬로 사용 100
자료 영역 수신 187
출력 파일 참조 188
데이터베이스 파일로 대체(OVRDBF) 명령 16
도움말 정보
참조 : 온라인 도움말 정보
도움말 패널 그룹
온라인 도움말 정보 414
독립 ASP의 메세지 파일 218
디버거
ILE 소스 420
디버그
명령 420
변경 446
세션
프로그램 오브젝트 제거 425
프로그램 오브젝트 준비 421
프로그램에 추가 423
시작 455
화면 459
디버그 명령
BREAK 432
CLEAR 432
디버그 변경(CHGDBG) 명령 446
디버그 시작(STRDBG) 명령
예 445
파일 갱신 방지 446
프로그램 추가 446
디버그 표시(DSPDBG) 명령 459
디버깅 모드 444
기계 인터페이스 레벨 465
다른 작업에서 462
다른 작업으로부터 어떤 한 작업에 대한 고려사항 464
대화식 작업 464
수행 중인 작업 463
시작 446
어플리케이션 테스트 445
작업 대기행렬에 제출된 일괄처리 작업 462
작업 대기행렬에서 시작되지 않은 일괄처리 작업 463
ILE 소스 디버거 명령 423
ILE 소스 디버거 시작 423
ILE 프로그램 419
디폴트 값
매개변수에 대한 정의 350
메세지 228
명령 변경 397
응답 228
디폴트 처리 284
모니터되지 않은, 디폴트 처리 448
테스트 시 모니터되지 않은 메세지 448
디폴트 표 350
디폴트 프로그램
테스트 시 사용된 446

[라]

라이브러리
권한 130
그룹화 8
보안 129
비움 135
삭제 135
설명 117
실행 129
오브젝트 그룹화 128
오브젝트 설명 검색 68, 144
오브젝트에 놓기 134
이전 릴리스 78
라이브러리 *(계속)*
자원 할당 164
작성 129
재명명 고려사항 158
정의 7
테스트 129
화면
라이브러리 리스트 127
오브젝트 136
오브젝트 설명 139
이름과 내용 136
라이브러리 리스트
규정된 이름과 비교 121
변경 28, 124
부분
사용자 부분 117
시스템 부분 설명 117
제품 라이브러리 117
현재 라이브러리 117
사용자 부분 123
설정 126
시스템 부분 123
오브젝트 액세스 119
입력
제거 124
추가 124
작업 123
저장 125
제품 라이브러리 123
초기화
QSYSLIBL 시스템 값 124
QUSRLIBL 시스템 값 124
탐색 순서 119
현재 라이브러리 117, 123
화면 127
*CURLIB 값 117
라이브러리 리스트 변경(CHGLIBL) 명령 28, 124
라이브러리 리스트 항목 제거(RMVLIBLE) 명령 124
라이브러리 리스트 항목 추가(ADDLIBLE) 명령 124
라이브러리 비움(CLRLIB) 명령 135
라이브러리 삭제(DLTLIB) 명령 135
라이브러리 설명
검색 137
화면 137
라이브러리 설명 검색(RTVLIBD) 명령 137

라이브러리 설명 표시(DSPLIBD) 명령 137
라이브러리 작성(CRTLIB) 명령 129
라이브러리 표시(DSPLIB) 명령 136
라이브러리명
  지정 117
라이브러리에 오브젝트 놓기 134
레이블
  CL 프로시듀어 36
로컬 자료 영역 108
루트 소스 보기
  사용 421
리모트 자료 대기행렬
  리모트 자료 대기행렬 98
리모트 자료 영역
  리모트 자료 영역 110
리스트
  단순 리스트에 대해 CL 또는 HLL 361
  명령 정의 391
  정의 359
  지정할 변수 28
  CL 또는 HLL에 대한 혼합 366
  CL 또는 HLL에 대해 369
  REXX
    단순 363
    리스트 안의 371
    혼합 367
리스트 안의 리스트로 368
  CL 또는 HLL 사용 369
  REXX의 사용 371
리스팅 보기
  사용 422
리턴 코드
  매개변수 49
  요약 49, 76
  BASIC 프로그램 49
  CL 프로시듀어 49
  Pascal 프로그램 49
  PL/I 프로그램 49
  RPG IV 프로그램 49
리턴 코드(RTNCDE) 매개변수 49
리턴(RETURN) 명령 20, 84

[마]

매개변수
  가능한 선택사항 및 값 376
  값
    길이 350
매개변수 *(계속)*
  값 *(계속)*
    유효 346
  매개변수에 대한 제한값 345
  속성 정보 전달 346
  수신 89
  순서 84
  유형
    경로명(*PNAME) 347
    널(*NULL) 347
    논리(*LGL) 347
    명령문 레이블 347
    문자(*CHAR) 347
    변수명(*VARNAME) 347
    유효 매개변수 조합 359
    이름(*NAME) 347
    정수(*INTn) 347
    총칭명(*GENERIC) 347
    10진(*DEC) 347
  유효 매개변수 346
  전달 89, 468
  정의 346
    값 길이 350
    고려사항 345
    규정된 이름 사용 372
    단순 리스트 359
    디폴트 값 350
    리스트 안의 리스트로 368
    리턴값 346
    매개변수 유형별로 유효한 359
    상수값 346
    설명 345
    속성 정보 전달 346
    예 351
    옵션 346
    유형 347
    유효 값 346
    유효 값 결정 346
    유효 조합 359
    제한값 345
    키워드, 명명 347
    필수 346
    혼합 리스트로 365
  지정
    값 길이 346
    값과 함께 리턴된 길이 346
    프롬프트 텍스트 346
  키 381
매개변수 *(계속)*
  키 식별 382
  프로그램 간의 전달 84
  후미 공백 32
  CMD 명령 19
  EXITPGM(나감 프로그램) 204
  RQSDTA(자료 요구) 19
  RTNCDE(리턴 코드) 49
  TEXT(텍스트) 139
  참조 : 명령 정의
매개변수 값
  대체 29
  리스트
    단순 360
    정의 359
    혼합 365
매개변수 값 표의 길이 350
매개변수 값의 리스트
  단순 359
  요소
    요소(ELEM)문의 사용 365
  정의 359
매개변수 선택 376
매개변수 조합표 351
매개변수(PARM) 명령 정의문
  사용 345
  설명 344
  예 405
  참조 : 매개변수
매개변수(PARM)문
  사용 345
  예 351
메뉴
  소개 5
  프로그램 203
메세지 236, 331
  검색
    CL 프로시듀어 20, 277
  대기행렬 10
  디폴트 값 228
  매개변수 60
  메세지 파일 대체 232
  메세지 파일 크기 218
  메세지 ID의 할당 220
  명령문 상태
    사용 251
    설명 286
    정의 217

메세지 *(계속)*
모니터링
사용 60
설명 280
숫자 부속 유형 코드 221
예 20
사전정의
메세지 대기행렬 215
설명 9
IBM 제공 파일 215
사전정의된 설명 219
서브파일
사용 197
설명
정의 10
송신 216, 247
수신
CL 프로그램 271
CL 프로시듀어 20, 271
시스템 사용자에게 송신 247
시스템 응답 리스트 사용 315
심각도 코드 할당 222
에 대한 작업 216, 247
예
변경 252
송신 252
온라인 도움말 정보 221
완료 217
요구 217, 273
유형 215
유효성 검사 225
응답 217
이력 기록부에 기록 319
이탈
목적 251
설명 280
정의 217
일시 중단(break) 전달 238
일시 중단(break) 처리 프로그램 241
작업 기록부에 기록 319
작업 메세지 대기행렬 242
전달 238
전달 모드 변경 241
정보 216, 250
정의 9
대체 변수 223
도움말 221
설명 221
메세지 *(계속)*
제거
메세지 대기행렬로부터 279
CL 프로시듀어 20
조회 216, 250
즉시 9, 215
진단 217
처리 215
테스트 시 디폴트 처리 449
텍스트 221
통지 217, 286
파일
IBM 제공 215
파일에 추가 219
필터
설명 321
화면
명령 옵션 216
일시 중단(break) 전달 238
2바이트 자료
정의 231
CL 프로그램으로부터 송신 248
IBM 제공 메세지 파일 215
QHST(이력 기록부) 파일 336
QSYSMSG 메세지 대기행렬로 송신된
CPD4070 291
CPF0907 291
CPF1269 292
CPF1393 292
CPF1397 292
CPF510E 293
CPF5167 293
CPF5244 293
CPF5248 293
CPF5250 293
CPF5251 294
CPF5257 294
CPF5260 294
CPF5274 294
CPF5341 295
CPF5342 295
CPF5344 295
CPF5346 295
CPF5355 296
CPF8AC4 296
CPF9E7C 296
CPI091F 296
CPI0948 297
메세지 *(계속)*
QSYSMSG 메세지 대기행렬로 송신된 *(계속)*
CPI0949 297
CPI0950 297
CPI0953 297
CPI0954 297
CPI0955 297
CPI095A 298
CPI0964 298
CPI0965 298
CPI0966 298
CPI096B 298
CPI096C 298
CPI096D 298
CPI096E 299
CPI0970 299
CPI0988 299
CPI0989 300
CPI0998 300
CPI0999 300
CPI099C 300
CPI099D 301
CPI099E 301
CPI099F 302
CPI1117 304
CPI1136 304
CPI1138 304
CPI1139 304
CPI1153 304
CPI1154 304
CPI1159 304
CPI1160 304
CPI1161 304
CPI1162 305
CPI1165 305
CPI1166 305
CPI1167 305
CPI1168 305
CPI1169 305
CPI116A 302
CPI116B 302
CPI116C 303
CPI1171 305
CPI1468 306
CPI2209 306
CPI2239 306
CPI2283 306

메세지 *(계속)*
QSYSMSG 메세지 대기행렬로 송신된 *(계속)*
CPI2284 306
CPI22AA 306
CPI8A13 306
CPI8A14 307
CPI9014 307
CPI9490 307
CPI94A0 307
CPI94CE 307
CPI94CF 307
CPI94FC 307
CPI96C0 307
CPI96C1 308
CPI96C2 308
CPI96C3 308
CPI96C4 308
CPI96C5 308
CPI96C6 308
CPI96C7 308
CPP0DD9 308
CPP0DDA 308
CPP0DDB 309
CPP0DDC 309
CPP0DDD 309
CPP0DDE 309
CPP0DDF 309
CPP1604 309
CPP29B0 309
CPP29B8 309
CPP29B9 310
CPP29BA 310
CPP951B 310
CPP9522 310
CPP955E 310
CPP9575 310
CPP9576 310
CPP9589 310
CPP9616 310
CPP9617 311
CPP9618 311
CPP961F 311
CPP9620 311
CPP9621 311
CPP9622 311
CPP9623 311
CPP962B 312
메세지 *(계속)*
QSYSMSG 메세지 대기행렬로 송신된 *(계속)*
CPPEA02 312
CPPEA04 312
CPPEA05 312
CPPEA12 312
CPPEA13 312
CPPEA26 312
CPPEA32 312
CPPEA38 313
CPPEA39 313
QSYSMSG로부터 수신하는 샘플 프로그램 313
QSYSOPR 메세지 대기행렬로 송신된
CPP0DDD 309
CPP1604 309
CPPEA02 312
CPPEA04 312
CPPEA05 312
CPPEA12 312
CPPEA13 312
CPPEA26 312
CPPEA32 312
CPPEA38 313
CPPEA39 313
메세지 검색(RTVMSG) 명령 20, 277
메세지 기록 레벨
상세 321
상위 레벨 321
메세지 대기행렬
기억장치량 238
메세지 송신 247
변경 238, 241
에 대한 작업 216
워크스테이션 239
작성 216, 238
프로그램에서 메세지 송신 248
호출 스택 입력 243
QSYSMSG 290
QSYSOPR 239
메세지 대기행렬 변경(CHGMSGQ) 명령 238, 241
메세지 대기행렬 유형 표 249
메세지 대기행렬 작성(CRTMSGQ) 명령 238
메세지 도움말 221
메세지 디폴트 전달 238
메세지 서브파일 197
메세지 설명
변경 216, 220, 236
에 대한 작업 216
정의 9
제거 216, 220
추가 216
값 215
대체 변수 223
예 230
파일에 219
화면 220, 230
메세지 설명 변경(CHGMSGD) 명령 220, 236
메세지 설명 제거(RMVMSGD) 명령 220
메세지 설명 추가(ADDMSGD) 명령
예 230
정보 지정 215
파일명 219
FMT(형식) 매개변수 223
메세지 송신(SNDMSG) 명령 247
메세지 송신(SNDMSG) 화면 194
메세지 유형 표 249
메세지 전달 보류 238
메세지 참조 키 271
메세지 통지 전달 238
메세지 파일
대체 232
변경 216
병합 218, 220
입력 *크기* 지정 219
작성 216, 217, 219
최대 *크기* 지정 218
메세지 파일 병합(MRGMSGF) 명령 218, 220
메세지 파일 작성(CRTMSGF) 명령 217, 219
메세지 파일로 대체(OVRMSGF) 명령 232
메세지 ID
지정 221
메세지에 대한 응답 225
메세지에 대한 작업 20
메세지의 일시 중단(break) 전달 238
메세지, 즉시 9
멤버
소스
삭제 409
멤버 설명
검색 20, 69

멤버 설명 검색(RTVMBRD) 명령 20, 69
명령
  디버그 420
  설명 1
  이름을 같게 함 442
  STEP 디버그 436
  참조 : 명령 정의
명령 도움말, CL
  작성
    단계 414
  정의
    설명 414
명령 디폴트
  변경 397
명령 문서, CL
  작성
    단계 417
  정의
    설명 417
명령 분석기 나감점 205
명령 사용
  인쇄 20
명령 사용 인쇄(PRTCMDUSG) 명령 20
명령 입력 화면 321
명령 정의
  규정된 이름 사용 372
  단순 리스트 359
  매개변수 유형별로 유효한 매개변수 359
  매개변수 조합표 359
  매개변수에 대한 리턴값 346
  매개변수의 프롬트 텍스트 346
  변경의 효과 395
  사용 5
  상태
    설명 340
    처리 시 오류 393
    DEP 375
    ELEM 365
    QUAL 372
  소개 5
  소스 리스트 391
  예 373
    디폴트 대체 명령 작성 406
    매개변수에 대한 정의 350
    약어 명령 작성 409
    어플리케이션 프로그램을 호출하는 명령 작성 405
    출력 대기행렬 표시 명령 작성 407
명령 정의 *(계속)*
  오브젝트 341
  자료 유형 및 매개변수에 대한 제한사항 351
  정의
    단순 리스트 359
  처리
    CL 프로그램 안의 규정된 이름 373
  필수 매개변수 346
  혼합 리스트로 365
  화면 393
  참조 : 매개변수
  참조 : 명령 처리 프로그램(CPP)
  참조 : 오브젝트
명령 처리 프로그램(CPP)
  기록 401
  설명 342
  예 406
  정의 5
  참조 : 명령 정의
명령 처리 프로시듀어
  REXX 작성 403
명령 프롬터
  소개 6
명령문 보기
  사용 422
명령문 조합표 354
명령어, 단계화 458
명령의 조건부 처리 35
명령(CMD) 매개변수 19
명령(CMD)문
  예 405
  정의 344
명령, CL 16, 20, 36, 37, 39, 41, 42, 48, 84, 184, 365, 390
  구성 상태 검색(RTVCFGSRC) 20, 65
  구성 소스 검색(RTVCFGSRC) 20, 65
  기능 20
  날짜 변환(CVTDAT) 20, 63
  네트워크 속성 검색(RTVNETA) 66
  데이터베이스 파일로 대체(OVRDBF) 16
  디버그 변경(CHGDBG) 446
  디버그 시작(STRDBG) 446, 455
  디버그 표시(DSPDBG) 459
  라이브러리 리스트 변경(CHGLIBL) 28, 124
  라이브러리 리스트 항목 제거(RMVLIBLE) 124
명령, CL *(계속)*
  라이브러리 리스트 항목 추가(ADDLIBLE) 124
  라이브러리 비움(CLRLIB) 135
  라이브러리 삭제(DLTLIB) 135
  라이브러리 설명 검색(RTVLIBD) 137
  라이브러리 설명 표시(DSPLIBD) 137
  라이브러리 작성(CRTLIB) 129
  라이브러리 표시(DSPLIB) 136
  메세지 대기행렬 변경(CHGMSGQ) 238, 241
  메세지 대기행렬 작성(CRTMSGQ) 238
  메세지 설명 표시(DSPMSGD) 220, 230
  메세지 송신(SNDMSG) 247
  메세지 파일 작성(CRTMSGF) 217, 219
  메세지 파일로 대체(OVRMSGF) 232
  멤버 설명 검색(RTVMBRD) 20, 69
  명령 사용 인쇄(PRTCMDUSG) 20
  명령 처리 프로그램(CPP) 342
  바인드 제어 언어 작성(CRTBNDCL) 20
  변경 395
  사용자 메세지 송신(SNDUSRMSG) 20, 250
  사용자 프로파일 검색(RTVUSRPRF) 20, 68
  서비스 프로그램 작성 20
  선택 종료(ENDSELECT) 20
  선택 프롬팅 199
  속성 389
  수신 종료(ENDRCV) 184, 186
  시스템 값 검색(RTVSYSVAL) 20, 62
  시스템 라이브러리 리스트 변경(CHGSYSLIBL) 124
  오브젝트 검사(CHKOBJ) 20, 172
  오브젝트 사본 작성(CRTDUPOBJ) 156
  오브젝트 설명 검색(RTVOBJD) 68, 144
  오브젝트 이동(MOVOBJ) 153
  오브젝트 잠금에 대한 작업(WRKOBJLCK) 167
  오브젝트 재명명(RNMOBJ) 158
  오브젝트 할당 해제(DLCOBJ) 166
  온라인 도움말 정보, 제공 414
  요구 종료(ENDRQS) 449
  일시 중단 메세지 송신(SNDBRKMSG) 248
  자료 영역 검색(RTVDTAARA) 20, 112
  자료 영역 변경(CHGDTAARA) 20, 112
  자료 영역 삭제(DLTDTAARA) 20

명령, CL *(계속)*
자료 영역 작성(CRTDTAARA) 20, 111
자료 영역 표시(DSPDTAARA) 20, 112
자원 재생(RCLRSC) 447
작성
단계 340
정의 341
처리 389
작성 예 405
작업 속성 검색(RTVJOBA) 20, 67
작업 표시(DSPJOB) 167
정의
규정된 이름 372
리스트 안의 리스트로 368
명령어 344
발생한 오류 393
설명 340
소스 리스트 391
예 405
유효성 검사 404
종속 관계 375
혼합 리스트 365
정의 변경의 효과 395
정의된, 필요한 권한 343
제어 언어 모듈 작성(CRTCLMOD) 20, 70
제어 이전(TFRCTL) 467, 468
중단점 재개(RSMBKP) 451
중단점 제거(RMVBKP) 455
중단점 추가(ADDBKP) 450
중단점 표시(DSPBKP) 459
처리 프로그램(CPP)
기록 401
정의 5
추적 자료 지우기(CLRTRCDTA) 455, 456
추적 자료 표시(DSPTRCDTA) 456, 458
추적 추가(ADDTRC) 456
추적 표시(DSPTRC) 459
파일 삭제(DLTF) 16
파일 선언(DCLF)
변수 25
사용 177
설명 16
파일 송신(SNDF) 175, 186
파일 송/수신(SNDRCVF) 175, 179
파일 수신(RCVF) 175, 186
명령, CL *(계속)*
프로그래머 메뉴 시작 (STRPGMMNU) 203
프로그램 메세지 송신 (SNDPGMMSG) 16, 248
프로그램 변수 변경(CHGPGMVAR) 461
프로그램 변수 표시(DSPPGMVAR) 459
프로그램 작성 20
프로그램 제거(RMVPGM) 446
프로그램 제어 변경 명령 20
프로그램 종료(ENDPGM) 16, 20
프로그램 추가(ADDPGM) 446
프로그램 호출(CALL) 81, 88
프로시듀어 호출(CALLPRC) 82
프롬터 사용 199
프롬트 대체 프로그램 지정
변경시 386
작성 시 386
현재 라이브러리 변경(CHGCURLIB) 124
호출
설명 95
화면 393
ADDBKP(중단점 추가) 450
ADDLIBLE(라이브러리 리스트 항목 추가) 124
ADDMSGD(메세지 설명 추가) 219
대체 변수 정의 223
예 230
ADDPGM(프로그램 추가) 446
ADDTRC(추적 추가) 456
CALLPRC(프로시듀어 호출) 82, 95
CALL(프로그램 호출) 81, 88
CHGCMD(명령 변경) 395
CHGCURLIB(현재 라이브러리 변경) 124
CHGDBG(디버그 변경) 446
CHGDTAARA(자료 영역 변경) 20, 112
CHGJOB(작업 변경) 319
CHGLIBL(라이브러리 리스트 변경) 28, 124
CHGMSGD(메세지 설명 변경) 220, 236
CHGMSGQ(메세지 대기행렬 변경) 238, 241
CHGPGMVAR(프로그램 변수 변경) 461
CHGSYSLIBL(시스템 라이브러리 리스트 변경) 124
CHGVAR(변수 변경) 16, 30
CHKOBJ(오브젝트 검사) 20, 172
CL 변수 선언(DCL) 16, 20
명령, CL *(계속)*
CL 프로그램에서 자주 사용되는 19
CL 프로시듀어 기록 70
CL 프로시듀어 한계 설정 명령 20
CL 프로시듀어에서 사용되는 19
CLRLIB(라이브러리 비움) 135
CLRTRCDTA(추적 자료 지우기) 455, 456
CMD(명령)문 345
CREATE BOUND CONTROL LANGUAGE(바인드된 CL 작성) 20
CRTCLMOD(제어 언어 모듈 작성) 20, 70
CRTCMD(명령 작성) 341, 389
CRTDTAARA(자료 영역 작성) 20, 111
CRTDUPOBJ(오브젝트 사본 작성) 156
CRTLIB(라이브러리 작성) 129
CRTMSGF(메세지 파일 작성) 217, 219
CRTMSGQ(메세지 대기행렬 작성) 238
CRTPGM(프로그램 작성) 20
CRTSRVPGM(서비스 프로그램 작성) 20
CVTDAT(날짜 변환) 20, 63
DCL(CL 변수 선언) 16, 20
DCLF(파일 선언)
변수 25
사용 177
설명 16
DLCOBJ(오브젝트 할당 해제) 166
DLTCMD(명령 삭제) 390
DLTDTAARA(자료 영역 삭제) 20
DLTF(파일 삭제) 16
DLTLIB(라이브러리 삭제) 135
DLTMCD(명령 삭제) 390
DLTPGM(프로그램 삭제) 20
Do For(DOFOR) 20, 45
Do Until(DOUNTIL) 20, 44
Do While(DOWHILE) 20, 44
DO(Do) 42
DOFOR(Do For) 20, 45
DOUNTIL(Do Until) 20, 44
DOWHILE(Do While) 20, 44
DSPBKP(중단점 표시) 459
DSPCMD(명령 표시) 393
DSPDBG(디버그 표시) 459
DSPDTAARA(자료 영역 표시) 20, 112
DSPJOBLOG(작업 기록부 표시) 328
DSPJOB(작업 표시) 167
DSPLIBD(라이브러리 설명 표시) 137

명령, CL *(계속)*
DSPLIB(라이브러리 표시) 136
DSPLOG(기록부 표시) 333
DSPMSGD(메세지 설명 표시) 220, 230
DSPMSG(메세지 표시) 238
DSPOBJD(오브젝트 설명 표시)
공통 속성 115
기록부 버전 선택 332
사용 139
DSPPGMVAR(프로그램 변수 표시) 459
DSPSPLF(스풀 파일 표시) 327
DSPTRCDTA(추적 자료 표시) 456, 458
DSPTRC(추적 표시) 459
End Do(ENDDO) 20, 42
ENDDO(End Do) 20, 42
ENDPGM(프로그램 종료) 16, 20
ENDRCV(수신 종료) 184, 186
ENDRQS(요구 종료) 449
ENDSELECT(선택 종료) 20, 48
GOTO(Go To) 20, 36
ITERATE(Iterate) 20, 46
Iterate(ITERATE) 20
LEAVE(Leave) 20, 47
Leave(LEAVE) 20
LODRUN(매체 프로그램 로드 및 수행) 212
MONMSG(메세지 모니터) 60, 280
MOVOBJ(오브젝트 이동) 153
MRGMSGF(메세지 파일 병합) 218, 220
OTHERWISE(Otherwise) 20, 48
Otherwise(OTHERWISE) 20
OVRDBF(데이터베이스 파일로 대체) 16
OVRMSGF(메세지 파일로 대체) 232
PRTCMDUSG(명령 사용 인쇄) 20
RCLRSC(자원 재생) 447
RCVF(파일 수신) 175, 186
RCVMSG(메세지 수신) 271, 272
RMVBKP(중단점 제거) 455
RMVLIBLE(라이브러리 리스트 항목 제거) 124
RMVMSGD(메세지 설명 제거) 220
RMVMSG(메세지 제거) 20, 279
RMVPGM(프로그램 제거) 446
RNMOBJ(오브젝트 재명명) 158
RSMBKP(중단점 재개) 451
RTVCFGSRC(구성 소스 검색) 20, 65
RTVCFGSTS(구성 상태 검색) 20, 65
RTVDTAARA(자료 영역 검색) 20, 112
명령, CL *(계속)*
RTVJOBA(작업 속성 검색) 20, 67
RTVLIBD(라이브러리 설명 검색) 137
RTVMBRD(멤버 설명 검색) 20, 69
RTVMSG(메세지 검색) 20, 277
RTVNETA(네트워크 속성 검색) 66
RTVOBJD(오브젝트 설명 검색) 68, 144
RTVSYSVAL(시스템 값 검색) 20, 62
RTVUSRPRF(사용자 프로파일 검색) 20, 68
SELECT(Select) 20, 48
Select(SELECT) 20
SNDBRKMSG(일시 중단 메세지 송신) 248
SNDF(파일 송신) 175, 186
SNDMSG(메세지 송신) 247
SNDPGMMSG(프로그램 메세지 송신) 16, 248
호출 스택 입력 255
SNDRCVF(파일 송/수신) 175, 179
SNDRPY(응답 송신) 20, 279
SNDUSRMSG(사용자 메세지 송신) 20, 250
STRDBG(디버그 시작) 446, 455
STRPGMMNU(프로그래머 메뉴 시작) 203
TFRCTL(제어 이전) 467, 468
WHEN(When) 20, 48
When(WHEN) 20
WRKOBJLCK(오브젝트 잠금에 대한 작업) 167
명령, 감사 저널 항목 표시
DSPAUDJRNE 132
모니터되지 않은 메세지
중단점 표시 448
처리 448
모니터링
메세지
사용 60
특정 명령 레벨 281
프로그램 레벨 281
CL 프로시듀어 280
모듈
설명 2
모듈 속성
화면 76
모듈 오브젝트
보기 변경 427, 428
무조건 분기 35
무조건부 중단점
설정 430
제거 430
문서화 보조
CL 명령 리스트 71
문자
소문자
변수 29
문자 길이 오류 94

[바]

방지
상태 메세지 표시 287
작업 기록부 328
작업 기록부 생성 328
테스트 시 파일 갱신 446
배타적 읽기 허용(*EXCLRD) 잠금 상태 164
배타적(*EXCL) 잠금 상태 164
범용 라이브러리(QGPL) 136
변경
디버그 446
라이브러리 리스트 28, 124
메세지 대기행렬 238, 241
메세지 설명 220, 236
명령 395
모듈 오브젝트 427, 428
변수
예 30, 254
CL 프로시듀어 16, 20
변수 값
프로그램 안의 461
수행 시 CL 프로그램 198
시스템 라이브러리 리스트 124
자료 영역 20, 112
작업 319
프로그램 변수 461
프로그램상의 명령 정의 효과 395
현재 라이브러리 124
변경 권한 130
변경 허용
변경
변경 허용 사용 239
변수
규정된 이름 지정 28
대체 223
리스트 지정 28

변수 *(계속)*
매개변수 값 대체 29
변경
값 30, 440
예 30, 254
프로그램 안의 값 461
CL 프로시듀어 16, 20
변수 안의 소문자 29
변수로 선언된 인디케이터 176
사용된 값 62
선언
설명 27
파일에 대해 177
필드에 대해 177
시스템 값 검색 62
에 대한 작업 25
오브젝트 작성 25
이름을 같게 함 442
정의 25
프로그램 안의 값 표시 459
화면 438
변수 변경(CHGVAR) 명령
예 30, 254
정의 20
변환
날짜 20, 63
날짜 형식 63
병합
메세지 파일 218, 220
보기
프로그램 소스 427
보안
오브젝트 130, 131
보호
의도하지 않은 수정을 통해 파일이, 테스트 446
분기
무조건부 35
비움
라이브러리 135
추적 자료 455
비정상 작업 종료 209

[사]

사용
루트 소스 보기 421
리스팅 보기 422
사용 *(계속)*
명령문 보기 422
QCMDCHK 프로그램 195
사용 정보
갱신 148
갱신 안 됨 152
표 148
사용자 메세지
송신
기능 216
정보 250
조회 250
CL 프로시듀어 20
사용자 메세지 송신(SNDUSRMSG) 명령 20, 250
사용자 인터페이스 관리 프로그램(UIM) 414
사용자 프로파일 검색(RTVUSRPRF) 명령 20, 68
사전정의 메세지 9, 215
삭제
라이브러리 135
명령 390
소스 멤버 410
오브젝트 163
자료 영역 20
파일 16
파일 멤버 410
프로그램 20
프로그램 오브젝트 410
HLL 프로그램 410
QHST 파일 338
삭제 권한 130
상세 메세지
설명 321
상수값
매개변수에 대한 정의 346
상태
명령 정의 340
상태 메세지
모니터링 286
송신 251
수신 280
정의 217
표시 방지 287
서브스트링 기능
규정된 이름 처리 373
설명 55
서브스트링 내장 기능(%SUBSTRING)
규정된 이름 처리 373
설명 55
서브파일
메세지 197
서비스 프로그램 2
서비스 프로그램 작성(CRTSRVPGM) 명령 20
선언
CL 변수 16
선택 종료(ENDSELECT) 명령 20
선택 프롬팅
문자 설명 표 202
문자 표 201
설명 199
선택적 매개변수
정의 346
설정
중단점 429
설치
CL 컴파일러 지원 79
성능
고려사항 100
메세지 대기행렬 100
이점
TFRCTL 명령 사용 467
자료 대기행렬 장점 100
세기 숫자
CPP(명령 처리 프로그램)에 대한 매개변수 값
날짜 347
세션
디버그
프로그램 오브젝트 제거 425
프로그램에 추가 423
소스 디버거
ILE
시작 423
소스 리스트
명령 정의 391
소스 멤버
삭제 409
소스 보기
에 대한 작업 443
속성
검색 212
기본 139
메세지 대기행렬 238

속성 *(계속)*
명령 389
모듈 표시 76
새로 작성된 오브젝트에 대한 디폴트 134
서비스 139
오브젝트 설명 139
전체 139
프로그램 표시 76
송신
메세지 247, 252
사용자 메세지 20, 250
시스템 사용자에게 메세지를 247
응답 20, 279
일시 중단 메세지 248
파일
예 186
자료 179
표시장치로 자료 175
프로그램 메세지 16, 248
화면 파일 20, 175
수
동시에 디버그될 수 있는 프로그램 446
리스트 안의 값 346
명령문 범위의 수 456
추적할 명령문 범위 456
수신
데이터베이스 파일 20, 175
메세지
기능 20
정보 배치 272
CL 프로그램 안의 271
CL 프로시듀어 271
사용자 응답 20
자료 표시 175
종료 184, 186
파일
예 179, 186
수신 종료(ENDRCV) 명령
복수 장치 화면 파일 184, 186
수행
계산
관계 49
문자 49
산술 49
수행 시
수행 시 CL 명령의 사용자 변경 허용 198
숫자 매개변수 값
대체 29
변수 대체 19
스위치 함수 58
스택, 호출
설명 447
오류가 있는 요구의 제거 449
테스트 정보 표시 459
호출 제거 82, 83
CALL 명령에 대한 관계 82
CALLPRC 명령에 대한 관계 83
스풀 파일
화면 327
시간종료 210, 212
시스템 값
검색 20, 62
시스템 값 검색(RTVSYSVAL) 명령 20, 62
시스템 기록부
명명 버전 332
참조 : 시스템 값
시스템 라이브러리 리스트
변경 124
시스템 라이브러리 리스트 변경 (CHGSYSLIBL) 명령 124
시스템 라이브러리(QSYS) 123, 136
시스템 사용자
메세지 송신 247
시스템 오퍼레이터(QSYSOPR) 메세지 대기행렬 237, 241
시스템 응답 리스트 315
시스템에서 사용하지 않는 오브젝트 검출 147
시작
디버그 446, 455
프로그래머 메뉴 203
ILE 소스 디버거 423
실행 권한 130
실행 라이브러리 129, 208
심각도 코드(severity code) 222

[아]

압축
오브젝트 160
오브젝트 도표 160
압축 해제
오브젝트 160
어플리케이션 프로그래밍 인터페이스(API)
사용일 계수 152
언어
서로 다른 것을 사용 137
피처(feature) 코드 137
에 대한 작업
메세지 247
오브젝트 잠금 167
여과 252
연산자
관계 49
논리 49
문자 49
산술 49
예
검색
네트워크 속성 66
사용자 프로파일 68
시스템 값 62
오브젝트 설명 146
자료 영역 112
작업 속성 67
내포된 DO 그룹 43
논리 변수 표시 439
논리식 49
단순명 사용 261
디폴트 메세지 프로그램의 샘플 229
라이브러리 리스트 대체 125
라이브러리 리스트 저장 125
메뉴 제어 181
메세지 230
메세지 처리 프로그램 241
메세지 파일 대체 234
명령 처리 프로그램 406
모니터링
특정 명령에 대한 메세지 281
프로시듀어 안의 메세지 283
문자 변수 표시 439
변경
메세지 252
변수 값 30
잠금 상태 167
변수 변경
논리 441
문자 441
10진 442
변수 속성 442
서브스트링 기능 55
설명
메세지 230

예 *(계속)*
송신
메세지 252
프로그램 메세지 251
수행 시 스택 호출 258, 269
스위치 함수 59
시스템 값 변환 63
오브젝트
규정된 이름 8
오브젝트 이동 155
오브젝트의 규정된 이름 8
완전명 사용 262
일시 중단(break) 처리 프로그램 289
자료 대기행렬 103
작성
디폴트 대체 명령 406
명령 343, 405, 406
약어 명령 작성 409
어플리케이션 프로그램을 호출하는 명령 405
출력 대기행렬을 표시하기 위한 명령 407
CL 프로시듀어 18
작업 기록부에 메세지 기록 321
전달
매개변수 91
프로그램으로 제어 81
프로시듀어 제어 82
정의
매개변수 351, 405
명령어에 대한 프롬프트 텍스트 405
제어 이전(TFRTCL) 명령 468
조건부 중단점 434
처리
CL 프로그램 안의 규정된 이름 373
초기 프로그램 125
추가
프로그램에 대한 추적 455
프로그램에 중단점 452
컴파일러 리스팅 72
프롬트 대체 프로그램 386
화면 파일 178
화면 파일 선언 178
10진 변수 표시 439
16진 형식으로 변수 표시 440
2진 기능 53
ADDMSGD(메세지 설명 추가) 명령 230
BIN 기능 53
예 *(계속)*
CALL 명령 81
CALLPRC 명령 82
CL 프로그램
규정된 이름 처리 373
CL 프로그램 샘플 207
CL 프로그램 안의 DBCS 자료 206
CL 프로시듀어
단순 18
일반적 15, 21
처리 제어 24
CRTMSGF(메세지 파일 작성) 명령 219
DDS
화면 파일 178
DO 명령 42
DOFOR 명령 45
DOUNTIL 명령 44
DOWHILE 명령 44
ENDDO 명령 42
ENDSELECT 명령 48
GOTO 명령 36
IF(If) 명령 37
ITERATE 명령 46
LEAVE 명령 47
OTHERWISE(Otherwise) 48
QHST 파일의 삭제 338
QINSTAPP 프로그램 213
QSYSMSG로부터 메세지 수신 313
SELECT 명령 48
SST 기능 55
TOPGMQ(*PRV*) 259
WHEN(When) 48
*BCAT 값 253
*CTLBDY의 사용 270
*PGMBDY의 사용 264, 265, 267
예약된 매개변수 값
대체 29
변수 대체 19
예외 메세지
RMV 키워드의 사용 273
오류
명령 정의문 393
문자 길이 94
자료 유형 91
정밀도 93
컴파일러 74
프로그램 호출 91
프로시듀어 91
오류 *(계속)*
10진 길이 93
오브젝트
검사 20, 172
공통 기능 표 116
공통 속성 115
권한 검증 115
규정된 이름
설명 8
예 8
그룹화 8
단일 탐색 128
디폴트 감사 속성 134
디폴트 공용 권한 132
라이브러리 117
라이브러리 간의 이동 153
라이브러리에 놓기 134
라이브러리에 표시 136
명령 정의 341
명명 7
모듈
변경 427
보기 변경 428
보안 130, 131
복수 탐색 128
사본 156
사용 정보 148
사용하지 않는 검출 147
삭제 163
설명 115, 139
손상 발견 및 통지 115
수행 기능 115, 116
압축
사용 160
제한사항 160
표 160
압축 해제
오퍼레이팅 시스템 설치 후 162
일시적으로 161
제한사항 160
액세스
규정된 이름으로 117
라이브러리 리스트 117
오브젝트 재명명
제한사항 158
유형 6, 115
갱신 148
유형 검증 115

오브젝트 *(계속)*
이동
제한사항 154
작성 156
변수 사용 25
설명 제공 139
정보 147
잠금 강제 115
잠금 상태 164
재명명 158
정의 6
제한사항
사본 156
참조
오브젝트 169
CL 프로시듀어 169
총칭명 127
테스트 라이브러리에서 실행 라이브러리로 이동 208
특정 기능 116
특정 저장 209
프로그램
디버그 세션 준비 421
디버그 세션 추가 423
디버그 세션에서 제거 425
할당 164
할당 해제 166
CL 프로시듀어
에 대한 작업 169
TEXT(텍스트) 매개변수 139
오브젝트 검사(CHKOBJ) 명령 20, 172
오브젝트 권한 130
오브젝트 사본
작성 156
오브젝트 사본 작성(CRTDUPOBJ) 명령 156
오브젝트 설명
검색 68, 144
화면
기록부 버전 332
사용 139
온라인 도움말 115
오브젝트 설명 검색(RTVOBJD) 명령 68, 144
오브젝트 이동(MOVOBJ) 명령 153
오브젝트 잠금
에 대한 작업 167
오브젝트 잠금에 대한 작업(WRKOBJLCK) 명령 167
오브젝트 재명명(RNMOBJ) 명령 158
오브젝트 조작(*OBJOPR) 권한 130
오브젝트 존재(*OBJEXIST) 권한 130
오브젝트 할당 해제(DLCOBJ) 명령 166
온라인 도움말 정보
도움말 패널 그룹 414
명령 414
명령에 대한 제공 414
완료 메세지 217, 251
외부 단계화 디버그 명령 437
요구
종료 449
요구 메세지 217, 273
요구 종료(ENDRQS) 명령 449
요구 처리 프로그램
기록 275
존재 판별 276
요소
리스트 안의 정의 365
요소(ELEM)문
명령 정의 344
사용 365
예 365, 368
워크스테이션 메세지 대기행렬 237
유효성 검사
기록 404
응답 225
프로그램 341
응답
송신 20, 279
응답 메세지 217
이동
한 라이브러리에서 다른 라이브러리로 오브젝트 153
이력 기록부의 내용 표시 화면 333
이력 기록부(QHST)
메세지 대기행렬 331, 334
버전 331
설명 331
처리 336
형식 334
형식표 334
이전
제어 467, 468
이전 릴리스
소스 프로그램 컴파일 77
컴파일러 지원 설치 79
이탈 메세지
모니터링 280
송신 251
정의 217
CPF2469 232
인쇄
명령 사용 20
일괄처리 입력 15
일괄처리 작업
작업 대기행렬에 제출된, 디버그 462
작업 대기행렬에서 시작되지 않은 작업 디버그 463
중단점 프로그램 451
일괄처리 작업 기록부
고려사항 331
일시 중단 메세지
송신 216, 248
일시 중단 메세지 송신(SNDBRKMSG) 명령 248
일시 중단 처리 프로그램 287, 288
일시 중단(break) 처리 프로그램 241
읽기 공유(*SHRRD) 잠금 상태 164
읽기 권한 130
입력
대화식 15
일괄처리 15
입력 필드 길이 350

[자]

자국어 정렬 순서(NLSS) 433
자국어 지원 137, 443
정의 137
자동 변수
프로그램 461
자동 압축 해제 162
자료 권한 130
자료 대기행렬
기억장치 관리 101
사용 103
예 102
자료 송신 102
작성 101
프로그램 간의 통신 96
할당 102
자료 비교 시작 위치 316
자료 영역 111
검색 20, 112

자료 영역 *(계속)*
검색 예 112
그룹 109
변경 20, 112
삭제 20
설명 107
작성 20, 111
초기값 107
통신 107
화면 20, 112
참조 : 오브젝트
자료 영역 변경(CHGDTAARA) 명령 20, 112
자료 영역 삭제(DLTDTAARA) 명령 20
자료 영역 작성(CRTDTAARA) 명령 20, 111
자료 영역, 작업할 명령 20
자료 요구(RQSDTA) 매개변수 19
자료 유형 오류 91
자원
재생 447
할당 164
자원 재생(RCLRSC) 명령 447
자주 사용되는 오브젝트
설명 163
작성
라이브러리 129
메세지 대기행렬 238
메세지 파일 217, 219
명령
설명 341, 389
속성 389
예 343, 405
오브젝트 사본 156
오브젝트에 대한 정보 147
온라인 도움말 정보 414
유효한 유형 111
자료 영역 20, 111
작성 20
CL 프로시듀어 20, 70
작업
대화식
테스트 기능 444
변경 319
일괄처리
테스트 기능 444
제출 209
화면 167
작업 *(계속)*
참조 : 일괄처리 작업
작업 기록부
대화식인 경우 고려사항 330
사용 시의 제안 329
생성 금지 328
설명 319
지시 471
출력 파일 471
화면 328
1차 모델 471
작업 대기행렬
시작되지 않은 일괄처리 작업 디버그 463
제출된 일괄처리 작업 디버그 462
작업 메세지 대기행렬 237, 242
작업 변경(CHGJOB) 명령 319
작업 속성
검색 20, 67
작업 속성 검색(RTVJOBA) 명령 20, 66
작업 표시(DSPJOB) 명령 167
잠금 상태
갱신 공유(*SHRUPD) 164
배타적 읽기 허용(*EXCLRD) 164
비갱신 공유(*SHRNUP) 164
오브젝트 유형표 164
읽기 공유(*SHRRD) 164
조합 표 164
*EXCLRD(배타적 읽기 허용) 164
*EXCL(배타적) 164
*SHRNUP(공유된 갱신 불가) 164
*SHRRD(공유된 읽기) 164
*SHRUPD(공유된 갱신) 164
재개
중단점 451
재명명
오브젝트 158
재생
자원 447
전달 347
매개변수에 대한 속성 정보 346
유형
*DATE(날짜) 347
*TIME(시간) 347
CPP로 매개변수 값 347
경로명 값 347
규정된 이름 372
논리값 347
리스트 359
전달 *(계속)*
CPP로 매개변수 값 *(계속)*
문자 값 347
변수 347
이름 347
총칭명 347
10진값 347
정밀도 오류 93
정보용 메세지 216, 250
정의
규정된 이름 372
단순 리스트 360
대체 변수 223
리스트 안의 리스트로 368
리스트 안의 요소
단순 리스트 360
매개변수 346
매개변수 리스트 359
매개변수에 대한 리턴값 346
매개변수에 대한 제한값 345
매개변수의 프롬트 텍스트 346
명령
권한 343
매개변수 376
상태 344
정의 340
명령 도움말
정의 414
명령 문서
정의 417
선택적 매개변수 346
유효 매개변수 346
필수 매개변수 346
CL 명령표 340
정의 오브젝트, 명령 341
정의면, 명령 340
정적 변수
설명 461
제거
디버그 세션에서 프로그램 오브젝트 425
라이브러리 리스트 항목 124
메세지 20, 279
메세지 대기행렬로부터 메세지 279
메세지 설명 220
시스템으로부터 자료 추적 458
중단점 429, 435, 455
프로그램 446
프로그램으로부터 중단점 455

제거 *(계속)*
프로그램으로부터 추적 459
제어
이전
사용 468
설명 467
CL 프로그램 안의 논리 흐름 35
CL 프로시듀어의 처리 35
제어 언어 모듈 작성(CRTCLMOD) 명령 20, 70
제어 언어(CL)
메뉴
제어하기 위해 CL 프로그램을 사용 181
명령
구문 3
입력 1
정의 1
프로그램
메뉴 제어 181
메세지 서브파일 197
메세지 송신 248
메세지 수신 271
메세지 처리 241
서브스트링 내장 기능 (%SUBSTRING) 373
설명 13
소개 13
수행 시 사용자 변경 허용 197
예 18
자료 송신 179
자료 수신 179
지원된 파일 173
처리 제어 24
프로그램 간의 흐름 제어 81
프로그램 예 207
화면 서식 173
화면 파일, 사용 173
DBCS 자료 206
프로시듀어
메세지 모니터 280
부분 16
설명 3
오브젝트 참조 169
작성 20, 70
CL 안에서 사용된 3
참조: 명령, CL
제어 이전(TFRCTL) 명령 467, 468
제출
작업 209
제한사항
오브젝트 사본 156
오브젝트 압축 160
오브젝트 이동 154
CL 프로시듀어 13
조건 프롬팅 378
조건부 중단점
설정 431
예 434
제거 431
추가 454
조합 권한 130
조회 메세지 216, 250
종료
수신 184, 186
요구 449
프로그램 16, 20
종료, 비정상 209
주석 분리문자(/*) 35
준비
프로그램 오브젝트에 디버그 세션을 421
중단점
기능 429
무조건부
설명 429
설정 430
제거 430
설정
무조건부 430
설명 429
조건 431
제거
무조건부 430
설명 435
조건 431
프로그램으로부터 429
조건
설명 429
설정 431
예 434
제거 431
추가 454
특성 429
프로그램 처리의 재개 451
중단점 재개(RSMBKP) 명령 451
중단점 제거(RMVBKP) 명령 455
중단점 추가(ADDBKP) 명령
설명 450
예 452
중단점 표시(DSPBKP) 명령 459
중단점 프로그램
일괄처리 작업 451
즉시 메세지 215
진단 메세지 217, 251

[차]

참조 키
메세지 271
처리
디폴트 284
CL 명령 사용 24
CL 프로시듀어 35
초기화
라이브러리 리스트 124
총칭명
설명 127
최기 프로그램 모델(OPM)
송수신 326
최초 프로그램 모델(OPM) 프로그램
메세지 대기행렬
호출 스택 입력 243
추가
디버그 세션에 프로그램 오브젝트 423
라이브러리 리스트 항목 124
메세지 설명
값 215
예 230
파일 219
ADDMSGD(메세지 설명 추가) 명령 230
FMT(형식) 매개변수 223
프로그램에 대한 추적 455
프로그램에 중단점 450
프로그램에 추가 455
추가 권한 130
추적
화면 459
추적 자료
비움 455
화면 456
추적 자료 지우기(CLRTRCDTA) 명령 455, 456

추적 자료 표시(DSPTRCDTA) 명령 456, 458
추적 제거(RMVTRC) 명령 459
추적 추가(ADDTRC) 명령 456
  예 456
추적 표시(DSPTRC) 명령 459
취소
  테스트 시 요구 449

[카]

컴파일
  소스 프로그램 77
컴파일러 리스팅
  샘플 프로그램 72
  CL 프로시듀어 71
컴파일러 오류 74
컴파일러, CL
  지원 설치 79
키 매개변수
  사용 381
  식별 382
  정의 345

[타]

탐색
  오브젝트 127
테스트
  디버그 모드 445
  디폴트 프로그램 446
  요구 취소 449
테스트 기능
  설명 11
테스트 라이브러리 129, 208
통신
  자료 대기행렬로 사용 107
  프로시듀어 107
통지 메세지
  모니터링 280, 286
  송신 251
  정의 217
통합 언어 환경(ILE) 프로시듀어
  송신 326
  수신 326
  호출 스택 입력 메세지 대기행렬 243

[파]

파일
  데이터베이스
    닫기 176
    선언 176
    열기 176
  삭제 16, 409
  선언
    변수 16
    이름 25
    프로그램 20
    CL 프로그램 안의 177
  송신
    서브파일 레코드 179
    자료 175, 186
    CL 프로시듀어 20
  수신
    레코드 20, 179
    자료 175, 186
  이름
    매개변수 값으로 사용 345
  표시
    닫기 176
    선언 176
    열기 176
  CL 프로시듀어
    데이터베이스 파일 대체 187
    에 대한 작업 173
    참조 176
    화면 파일 대체 182
  참조 : 멤버
  참조 : 오브젝트
  참조 : 화면 파일
파일 멤버
  삭제 410
파일 삭제(DLTF) 명령 16
파일 선언(DCLF) 명령
  선언
    변수 27, 177
  설명 25
  CL 프로시듀어 16, 20
파일 송신(SNDF) 명령
  기능 175
  입력 요구 취소 186
  CL 프로시듀어 20
파일 송/수신(SNDRCVF) 명령
  기능 175
파일 송/수신(SNDRCVF) 명령 (계속)
  사용 179
  CL 프로시듀어 20
파일 수신(RCVF) 명령 175, 186
표시 452
  메뉴, 명령 입력에 사용 15
  명령 입력 15
  모니터되지 않은 메세지 중단점 448
  중단점 452
  추적 자료 456
  프로그래머 메뉴 18, 203
표현식
  관계 49
  논리 49
  이름을 같게 함 442
프로그래머 메뉴
  사용 203
  시작 203
프로그래머 메뉴 시작(STRPGMMNU) 명령 203
프로그램
  덤프 74
  동시에 디버그될 수 있는 수 446
  디버그 모드로 놓기 446
  디폴트, 테스트시 446
  명령 처리 프로그램 작성 401
  명령 처리 프로시듀어 작성 401
  변수
    화면 459
  삭제 20
  서비스 2
  설명 2
  에 중단점 추가 450
  에 추적 추가 455
  유효성 검사 작성 401, 404
  으로부터 추적 제거 459
  일시 중단(break) 처리 288
  제거 446
  종료 16, 20
  중단점 450
  중단점으로부터 제거 455
  추가 446
  프로그램 논리 명령 20
  프로그램 논리 제어 명령 20
  프롬트 대체 프로그램 342
  프롬트 제어 기록 377
  프롬트 제어 작성 383
  호출 447

프로그램 *(계속)*
사용 88
설명 81
CL 프로시듀어 20
활성화 447
CL 작성 70
QCMDCHK 195
QCMDEXC 198
참조: CL 프로그램
참조: 메세지 프로그램
참조: 오브젝트
참조: 프로그램 변수
프로그램 덤프
확보 74
프로그램 메세지
변경 216
송신
메세지 대기행렬 20, 248
CL 프로시듀어 16
프로그램 메세지 송신(SNDPGMMSG) 명령
사용 248
CL 프로시듀어 16, 20
프로그램 변수
변경 461
화면 459
프로그램 변수 변경(CHGPGMVAR) 명령 461
프로그램 변수 표시(DSPPGMVAR) 명령 459
프로그램 삭제(DLTPGM) 명령 20
프로그램 소스
보기 427
프로그램 속성
화면 76
프로그램 오브젝트
내부 단계화 436, 437
단계화 435
디버그 세션 준비 421
디버그 세션 추가 423
디버그 세션에서 제거 425
삭제 410
외부 단계화 436
프로그램 작성(CRTPGM) 명령 20
프로그램 제거(RMVPGM) 명령
사용 446
중단점 프로그램 459
추적된 프로그램 459
프로그램 제어 명령 16
프로그램 종료(ENDPGM) 명령
예 198
CL 프로시듀어 16, 20
프로그램 초기화 매개변수(PIP) 자료 영역
프로그램 초기화 매개변수(PIP) 110
프로그램 추가(ADDPGM) 명령 446
프로그램 호출
CALL 명령 사용 88
CALL 명령 설명 81
프로그램 호출(CALL) 명령
기능 20
사용 88
설명 81
프로그램 활성화 447
프로그램 흐름 81, 82
프로그램(PGM) 명령 16, 20
프로시듀어
메세지 수신 271
설명 1
제어 언어(CL) 소개 3
호출
설명 82
CL 3
CL의 프로그램 부분
설명 16
오브젝트에 대한 작업 169
프로시듀어 명령
기록 70
프로시듀어 제어 명령 16
프로시듀어 호출
CALLPRC 명령 설명 82
CALLPRC 명령 예 95
프로파일 속성
검색 20, 68
프롬터 6
도움말 15
사용 198
프롬트
텍스트
매개변수에 대한 정의 345
프롬트 대체 프로그램
기록 383
리턴된 정보 383
명령의 작성 및 변경시에 지정 386
사용을 위한 프로시듀어 382
설명 342
오류 허용 385
전달되는 정보 383
프롬트 대체 프로그램 *(계속)*
키 매개변수 사용 381
CL 샘플, 사용 386
프롬트 매개변수 345
프롬트 제어 377
프롬팅
명령 199
문자 설명 표 202
문자 표 201
선택 199
조건 378
CL 프로그램 안의 2바이트 자료에 대해 194
CL 프로시듀어
사용 198
QCMDEXC 203
QCMDEXC 사용에 대해 194
필드 정의
QMHCID 476
QMHCRP 479
QMHCSP 478
QMHDAT 473
QMHJBN 480
QMHJDT 472, 480
QMHJOB 478
QMHJTM 472
QMHJTS 480
QMHLIN 481
QMHLNN 480
QMHLSP 479
QMHMDT 478
QMHMF 474
QMHMID 473
QMHMKS 480
QMHMRK 472
QMHMSC 479
QMHPRL 476
QMHRLB 477
QMHRMD 477
QMHRPG 477
QMHRPR 477
QMHRPY 474
QMHRQS 474
QMHRSN 475
QMHRTM 478
QMHRTY 475
QMHSEV 473
QMHSID 480

필드 정의 *(계속)*
QMHSLB 477
QMHSMD 476
QMHSPG 476
QMHSPR 476
QMHSSN 475
QMHSTM 477
QMHSTY 474
QMHSYN 480
QMHSYS 478
QMHTID 479
QMHTIM 473
QMHTTY 481
QMHTYP 473
필수 매개변수 346
필터
메세지
SEV(심각도 코드 필터) 매개변수의 사용 239
설명 321

[하]

할당
자원 164
할당 해제
오브젝트 166
현재 라이브러리
변경 124
현재 라이브러리 변경(CHGCURLIB) 명령 124
현지 시간 사용(CEELOCT) 65
호출
내포 447
레벨
설명 447
설명 447
스택 447
호출 스택
설명 447
오류가 있는 요구의 제거 449
입력 ID
SNDPGMMSG 255
테스트 정보 표시 459
호출 제거 82, 83
CALL 명령에 대한 관계 82
CALLPRC 명령에 대한 관계 83
TFRCTL(제어 이전) 명령 467
호출 스택 입력 메세지 대기행렬 243
호출 스택 표시 화면 276
혼합 리스트
리스트 안의 요소
혼합 리스트 365
설명 365
정의 365
CL 또는 HLL 사용 366
CPP로 전달 365
REXX의 사용 367
화면 331
기록부 333
디버그 정보 459
라이브러리 136
라이브러리 설명 137
라이브러리 안의 오브젝트 136
메세지 238, 408
메세지 설명 220, 230
명령 393
명령 정의 393
모듈 속성 76
스풀 파일 327
오브젝트 설명
공통 속성 115
기록부 버전 선택 332
사용 139
오브젝트 잠금 167
이력 기록부(QHST) 332
일괄처리 작업 기록부 333
자료 영역 20, 112
작업 167
작업 기록부 327, 328
중단점 459
추적 459
추적 자료 456, 458
테스트 정보 459
프로그램 변수 459
프로그램 속성 76
프로그램 안의 변수 값 459
QHST 기록부 332
화면 파일
복수 장치 표시장치 사용 184
송신 179
수신 175, 179
작성 178
참조 176
CL 프로그램에서 사용 173
확보
프로그램 덤프 74
회복
시스템 비정상 종료 후 209
후미 공백
명령 매개변수 32
예 33

[숫자]

10진 길이 오류 93
1차 작업 기록부 모델 471
2바이트 메세지 231
2바이트 문자 세트(DBCS)
메세지 송신 228
메세지 정의 231
어플리케이션 프로그램 설계 205
DBCS 자료가 있는 CL 프로그램 작성 206
QCMDEXC를 사용 194
2바이트 자료
어플리케이션 프로그램 설계 205
즉시 송신 방법 228
2바이트 메세지 정의 231
2바이트 문자가 들어 있는 메세지의 전송 228
CL 프로그램에 대한 프롬팅 194
CL 프로그램에서 사용 206
QCMDEXC 프로그램의 사용에 대한 프롬팅 194
2진 기능 53

A

ADDBKP(중단점 추가) 명령
설명 450
예 452
ADDLIBLE(라이브러리 리스트 항목 추가) 명령 124
ADDMSGD(메세지 설명 추가) 명령
예 230
정보 지정 215
파일명 219
FMT(형식) 매개변수 223
ADDPGM(프로그램 추가) 명령 446
ADDTRC(추적 추가) 명령 456
예 456

ALROPT 코드
지정 230
항목 크기 219
API(어플리케이션 프로그래밍 인터페이스)
사용일 계수 152

C

Callprc(CALL PROCEDURE) 명령 20
CALLPRC(프로시듀어 호출) 명령 20
설명 82
예 95
CALL(프로그램 호출) 명령
기능 20
사용 88
설명 81
CCSID 249
메세지
CCSID 사용 239
CEELOCT 프로그램 65
CHGCMD(명령 변경) 명령 395
CHGCURLIB(현재 라이브러리 변경) 명령 124
CHGDBG(디버그 변경) 명령 446
CHGDTAARA(자료 영역 변경) 명령 20, 112
CHGJOB(작업 변경) 명령 319
CHGLIBL(라이브러리 리스트 변경) 명령 28, 124
CHGMSGD(메세지 설명 변경) 명령 220, 236
CHGMSGQ(메세지 대기행렬 변경) 명령 238, 241
CHGPGMVAR(프로그램 변수 변경) 명령 461
CHGSYSLIBL(시스템 라이브러리 리스트 변경) 명령 124
CHGVAR(변수 변경) 명령
예 30, 254
정의 20
CL 프로시듀어 16
%SWITCH 설정 59
CHKOBJ(오브젝트 검사) 명령 20, 172
CL 명령
참조 : 명령, CL
CL 명령으로 처리 제어 24
CL 변수
선언 16, 20
CL 변수 선언(DCL) 명령 16, 20
CL 프로그램
서브스트링 내장 기능(%SUBSTRING)
규정된 이름 처리에 사용 373
자료 송신 179
자료 수신 179
작성 14
지원된 파일 173
화면 서식 173
CL 프로시듀어
대화식 입력 15
덤프 확보 74
데이터베이스 파일 대체 187
명령
기록 70
사용된 19
목적 13
변수, 작업할 명령 20
부분 16
사용 21
사용 시 이점 13
설명 3
소개 13
소스 작성 14
에 대한 작업 70, 169
예 18
일괄처리 입력 15
작성
소스 사용 14
CRTCLMOD 명령 사용 14, 70
CRTCLMOD(제어 언어 모듈 작성) 명령 20
주석 작성 34
처리 제어 24, 35
컴파일러 리스팅 71
파일 참조 176
파일에 대한 작업 173
프로시듀어 작성 14
화면 파일 대체 182
CL 프로시듀어와 명령 기록 70
CLRLIB(라이브러리 비움) 명령 135
CLRTRCDTA(추적 자료 지우기) 명령 455
CMD(명령) 매개변수 19
CMD(명령)문
예 405
정의 344
CPP(명령 처리 프로그램)
기록 401
CPP(명령 처리 프로그램) *(계속)*
설명 342
예 406
정의 5
참조 : 명령 정의
CRTBNDCL(바인드 제어 언어 작성) 명령 20
CRTCLMOD(제어 언어 모듈 작성) 명령 20, 70
CRTCMD(명령 작성) 명령
관계 402
매개변수 389
예 406
처리 341
CL 프로그램 340
CRTDTAARA(자료 영역 작성) 명령 20, 111
CRTDUPOBJ(오브젝트 사본 작성) 명령 156
CRTLIB(라이브러리 작성) 명령 129
CRTMSGF(메세지 파일 작성) 명령 217, 219
CRTMSGQ(메세지 대기행렬 작성) 명령 238
CRTPGM(프로그램 작성) 명령 20
CRTSRVPGM(서비스 프로그램 작성) 명령 20
CVTDAT(날짜 변환) 명령 20, 63

D

DBCS(2바이트 문자 세트)
메세지 송신 228
메세지 정의 231
어플리케이션 프로그램 설계 205
DBCS 자료가 있는 CL 프로그램 작성 206
QCMDEXC를 사용 194
DCL(CL 변수 선언) 명령 16, 20
DCLF(파일 선언) 명령
선언
변수 177
설명 25
CL 프로시듀어 16, 20
DEP(종속)문
명령 정의 344
사용 375
예 376
DLCOBJ(오브젝트 할당 해제) 명령 166
DLTCMD(명령 삭제) 명령 390

DLTDTAARA(자료 영역 삭제) 명령 20
DLTF(파일 삭제) 명령 16
DLTLIB(라이브러리 삭제) 명령 135
DLTPGM(프로그램 삭제) 명령 20
Do For(DOFOR) 명령 20, 45
Do Until(DOUNTIL) 명령 20, 44
Do While(DOWHILE) 명령 20, 44
DO 그룹 43
DO(Do) 명령 20, 42
DOFOR(Do For) 명령 20, 45
DOUNTIL(Do Until) 명령 20, 44
DOWHILE(Do While) 명령 20, 44
DSPAUDJRNE 132
DSPBKP(중단점 표시) 명령 459
DSPCMD(명령 표시) 명령 393
DSPDBG(디버그 표시) 명령 459
DSPDTAARA(자료 영역 표시) 명령 20, 112
DSPJOBLOG(작업 기록부 표시) 명령 328
DSPJOB(작업 표시) 명령 167
DSPLIBD(라이브러리 설명 표시) 명령 137
DSPLIB(라이브러리 표시) 명령 136
DSPLOG(기록부 표시) 명령 333
DSPMSGD(메세지 설명 표시) 명령 220, 230
DSPMSG(메세지 표시) 명령 238
DSPOBJD(오브젝트 설명 표시) 명령
  기록부 버전 선택 332
  사용 139
  설명 115
DSPPGMVAR(프로그램 변수 표시) 명령 459
DSPSPLF(스풀 파일 표시) 명령 327
DSPTRCDTA(추적 자료 표시) 명령 456, 458
DSPTRC(추적 표시) 명령 459

E

ELSE(Else) 명령 20, 39
End Do(ENDDO) 명령 20, 42
ENDDO(End Do) 명령 20, 42
ENDPGM(프로그램 종료) 명령
  예 198
  CL 프로시듀어 16, 20
ENDRCV(수신 종료) 명령
  복수 장치 화면 파일 184, 186
ENDRQS(요구 종료) 명령 449
ENDSELECT(선택 종료) 명령 20, 48

G

GENCMDDOC(명령 문서 생성) 명령
  CL 프로그램 414, 417
GOTO(Go To) 명령 20, 36

H

HLL(고급 언어) 프로그램
  혼합 리스트 366
  QCMDEXC 프로그램 191
HTML(Hyper Text Markup Language) 417

I

IF(If) 명령 20
  내장 41
  설명 20
  예 37
  %SWITCH를 사용하여 59
IF(if) 명령
  CL 프로시듀어 20
ILE(통합 언어 환경) 모델
  메세지 대기행렬
    호출 스택 입력 243
  소스 디버거 420
  소스 디버거 시작 423
  통지 메세지 252
  프로시듀어
    송신 326
    수신 326
  CL 프로그램
    디버깅 모드 419
ITERATE(Iterate) 명령 20, 46
Iterate(ITERATE) 명령 20

L

LDA(내부 자료 영역) 108
LEAVE(Leave) 명령 20, 47
Leave(LEAVE) 명령 20
LODRUN(매체 프로그램 로드 및 수행) 명령 212

M

MONMSG(메세지 모니터) 명령
  사용 60
  CL 프로시듀어 280
MOVOBJ(오브젝트 이동) 명령 153
MRGMSGF(메세지 파일 병합) 명령 218, 220

O

OPM(초기 프로그램 모델)
  송수신 326
OPM(최초 프로그램 모델) 프로그램
  메세지 대기행렬
    호출 스택 입력 243
OS/400 언어 지원 137
OTHERWISE(Otherwise) 명령 20, 48
Otherwise(OTHERWISE) 명령 20
OVRDBF(데이터베이스 파일로 대체) 명령 16
OVRMSGF(메세지 파일로 대체) 명령 232

P

PARM(매개변수) 명령 정의문
  사용 345
  설명 344
  예 405
  참조 : 매개변수
PARM(매개변수)문
  사용 345
  예 351
PGM(프로그램) 명령 16, 20
PMTCTL(프롬트 제어) 명령 정의문 344
PRTCMDUSG(명령 사용 인쇄) 명령 20
PRV 78

Q

QBATCH 서브시스템 331
QCMDCHK 프로그램 195
QCMDEXC 프로그램
  명령 스트링의 처리 126
  프로그램으로부터 명령 수행 191
  호출 프롬터 198, 203
  2바이트 자료를 프롬팅 194
QGPL 라이브러리 136

QHST(이력 기록부) 메세지 대기행렬 332, 334
QHST(이력 기록부) 파일
  삭제 338
  작업 시작 메세지 336
  작업 완료 메세지 336
QPJOBLOG(작업 기록부) 파일 327
QRECOVERY 라이브러리 136
QSYS 라이브러리 123
QSYSMSG
  메세지 대기행렬
    정의 290
    CPF0907 291
    CPF1269 292
    CPF1393 292
    CPF1397 292
    CPF4070 291
    CPF510E 293
    CPF5167 293
    CPF5244 293
    CPF5248 293
    CPF5250 293
    CPF5251 294
    CPF5257 294
    CPF5260 294
    CPF5274 294
    CPF5341 295
    CPF5342 295
    CPF5344 295
    CPF5346 295
    CPF5355 296
    CPF8AC4 296
    CPF9E7C 296
    CPI091F 296
    CPI0948 297
    CPI0949 297
    CPI0950 297
    CPI0953 297
    CPI0954 297
    CPI0955 297
    CPI095A 298
    CPI0964 298
    CPI0965 298
    CPI0966 298
    CPI096B 298
    CPI096C 298
    CPI096D 298
    CPI096E 299
QSYSMSG *(계속)*
  메세지 대기행렬 *(계속)*
    CPI0970 299
    CPI0988 299
    CPI0989 300
    CPI0998 300
    CPI0999 300
    CPI099C 300
    CPI099D 301
    CPI099E 301
    CPI099F 302
    CPI1117 304
    CPI1136 304
    CPI1138 304
    CPI1139 304
    CPI1153 304
    CPI1154 304
    CPI1159 304
    CPI1160 304
    CPI1161 304
    CPI1162 305
    CPI1165 305
    CPI1166 305
    CPI1167 305
    CPI1168 305
    CPI1169 305
    CPI116A 302
    CPI116B 302
    CPI116C 303
    CPI1171 305
    CPI1468 306
    CPI2209 306
    CPI2239 306
    CPI2283 306
    CPI2284 306
    CPI22AA 306
    CPI8A13 306
    CPI8A14 307
    CPI9014 307
    CPI9490 307
    CPI94A0 307
    CPI94CE 307
    CPI94CF 307
    CPI94FC 307
    CPI96C0 307
    CPI96C1 308
    CPI96C2 308
    CPI96C3 308
QSYSMSG *(계속)*
  메세지 대기행렬 *(계속)*
    CPI96C4 308
    CPI96C5 308
    CPI96C6 308
    CPI96C7 308
    CPP0DD9 308
    CPP0DDA 308
    CPP0DDB 309
    CPP0DDC 309
    CPP0DDD 309
    CPP0DDE 309
    CPP0DDF 309
    CPP1604 309
    CPP29B0 309
    CPP29B8 309
    CPP29B9 310
    CPP29BA 310
    CPP951B 310
    CPP9522 310
    CPP955E 310
    CPP9575 310
    CPP9576 310
    CPP9589 310
    CPP9616 310
    CPP9617 311
    CPP9618 311
    CPP961F 311
    CPP9620 311
    CPP9621 311
    CPP9622 311
    CPP9623 311
    CPP962B 312
    CPPEA02 312
    CPPEA04 312
    CPPEA05 312
    CPPEA12 312
    CPPEA13 312
    CPPEA26 312
    CPPEA32 312
    CPPEA38 313
    CPPEA39 313
  샘플 프로그램 291
QSYSMSG로부터 메세지를 수신하는 샘플 프로그램 313
QSYSOPR
  메세지 대기행렬
    CPP0DDD 309

QSYSOPR *(계속)*
메세지 대기행렬 *(계속)*
CPP1604 309
CPPEA02 312
CPPEA04 312
CPPEA05 312
CPPEA12 312
CPPEA13 312
CPPEA26 312
CPPEA32 312
CPPEA38 313
CPPEA39 313
QSYSOPR 메세지 대기행렬 237
QUAL(규정자)문
사용 372
예 372, 408
정의 344

R

RCLRSC(자원 재생) 명령 447
RCVF(파일 수신) 명령 175, 186
RCVMSG(메세지 수신) 명령 271, 272
RETURN(리턴) 명령 20, 84
REXX 프로시듀어
리스트 안의 리스트로 371
명령 처리 프로시듀어 작성 403
사용
규정된 이름 375
단순 리스트의 예 363
혼합 리스트 367
RMVBKP(중단점 제거) 명령 455
RMVLIBLE(라이브러리 리스트 항목 제거) 명령 124
RMVMSGD(메세지 설명 제거) 명령 220
RMVMSG(메세지 제거) 명령 20, 279
RMVPGM(프로그램 제거) 명령
사용 446
중단점 프로그램 459
추적된 프로그램 459
RMVTRC(추적 제거) 명령 459
RNMOBJ(오브젝트 재명명) 명령 158
RQSDTA(자료 요구) 매개변수 19
RSMBKP(중단점 재개) 명령 451
RTNCDE(리턴 코드) 매개변수 49
RTVCFGSRC(구성 상태 검색) 명령 20, 65
RTVCFGSRC(구성 소스 검색) 명령 20, 65
RTVDTAARA(자료 영역 검색) 명령 20, 112
RTVJOBA(작업 속성 검색) 명령 20, 66
RTVLIBD(라이브러리 설명 검색) 명령 137
RTVMBRD(멤버 설명 검색) 명령 20, 69
RTVMSG(메세지 검색) 명령 20, 277
RTVNETA(네트워크 속성 검색) 명령 66
RTVOBJD(오브젝트 설명 검색) 명령 68, 144
RTVSYSVAL(시스템 값 검색) 명령 20, 62
RTVUSRPRF(사용자 프로파일 검색) 명령 20, 68

S

see='메세지 대기행렬'.메세지 448
see='멤버'.데이터베이스 파일
실행(production) 라이브러리 안의 갱신 방지 446
see='사용자 프로파일'.보안
디버그 고려사항 465
see='중단점 프로그램'.중단점 451, 452, 455
추적 안에서 사용 458
프로그램에 추가 450
see='중단점'.디버그 모드 446, 459, 465
see='중단점'.추적 455, 456
설명 455
시스템에서 정보 제거 458
추적 안에서의 중단점 사용 458
프로그램으로부터 제거 459
see='중단점 프로그램'.중단점 표시 459
see='추적'.디버그 모드 446
보안 고려사항 465
see='추적'.중단점 450, 451, 452, 458
위치 표시 459
프로그램으로부터 제거 455
see='테스팅'.디버그 모드 459, 465
프로그램 놓기 446
프로그램 추가 446
SELECT(Select) 명령 20, 48
Select(SELECT) 명령 20
SNDBRKMSG(일시 중단 메세지 송신) 명령 248
SNDF(파일 송신) 명령
기능 175
입력 요구 취소 186
CL 프로시듀어 20
SNDMSG(메세지 송신) 명령 247
SNDPGMMSG(프로그램 메세지 송신) 명령
사용 248
CL 프로시듀어 16, 20
SNDRCVF(파일 송/수신) 명령
기능 175
사용 179
CL 프로시듀어 20
SNDRPY(응답 송신) 명령 20, 279
SNDUSRMSG(사용자 메세지 송신) 명령 20, 250
STRDBG(디버그 시작) 명령
예 445
파일 갱신 방지 446
프로그램 추가 446
STRPGMMNU(프로그래머 메뉴 시작) 명령
사용 203

T

TFRCTL(제어 이전) 명령 467, 468

W

WAIT(대기) 명령 20, 184
WHEN(When) 명령 20, 48
When(WHEN) 명령 20
WRKOBJLCK(오브젝트 잠금에 대한 작업) 명령 167

[특수 문자]

*ALL 권한 130
*AND 연산자 49
*CHANGE 권한 130
*EXCLRD(배타적 읽기 허용) 잠금 상태 164
*EXCLUDE 권한 130
*EXCL(배타적) 잠금 상태 164
*INT2 값 403
*INT4 값 403
*LDA 값
로컬 108
*NOT 연산자 49
*OBJMGT(오브젝트 관리) 권한 130
*OR 연산자 49
*SHRNUP(공유된 갱신 불가) 잠금 상태 164
*SHRRD(공유된 읽기) 잠금 상태 164
*SHRUPD(공유된 갱신) 잠금 상태 164
*UINT2 403

*UINT4 403
*USE 권한 130
*(별표)
  프로그램 안의 주석 35
  OUTPUT(출력) 매개변수 137
/*(주석) 분리문자 35
%BIN(2진) 기능 53
%BINARY(2진) 내장 기능
  설명 53
%SST(서브스트링) 기능
  규정된 이름 처리 373
  설명 55
%SWITCH(스위치) 기능 58

SA30-0246-06