HITACHI
Inspire the Next

OpenTP1 Version 7

Programming Reference COBOL Language

3000-3-D55-30(E)

■ Relevant program products

Note: In the program products listed below, those marked with an asterisk (*) might be released later than the other program products.

For AIX 5L V5.1, AIX 5L V5.2, AIX 5L V5.3, AIX V6.1

P-1M64-2131 uCosminexus TP1/Server Base 07-03*

P-1M64-2331 uCosminexus TP1/FS/Direct Access 07-03*

P-1M64-2431 uCosminexus TP1/FS/Table Access 07-03*

P-1M64-2531 uCosminexus TP1/Client/W 07-02

P-1M64-2631 uCosminexus TP1/Offline Tester 07-00

P-1M64-2731 uCosminexus TP1/Online Tester 07-00

P-1M64-2831 uCosminexus TP1/Multi 07-00

P-1M64-2931 uCosminexus TP1/High Availability 07-00

P-1M64-3131 uCosminexus TP1/Message Control 07-03

P-1M64-3231 uCosminexus TP1/NET/Library 07-04

P-1M64-8131 uCosminexus TP1/Shared Table Access 07-00

P-1M64-8331 uCosminexus TP1/Resource Manager Monitor 07-00

P-1M64-8531 uCosminexus TP1/Extension 1 07-00

P-1M64-C371 uCosminexus TP1/Message Queue 07-01

P-1M64-C771 uCosminexus TP1/Message Queue - Access 07-01

P-F1M64-31311 uCosminexus TP1/Message Control/Tester 07-00

P-F1M64-32311 uCosminexus TP1/NET/User Agent 07-00

P-F1M64-32312 uCosminexus TP1/NET/HDLC 07-00

P-F1M64-32313 uCosminexus TP1/NET/X25 07-00

P-F1M64-32314 uCosminexus TP1/NET/OSI-TP 07-00

P-F1M64-32315 uCosminexus TP1/NET/XMAP3 07-01

P-F1M64-32316 uCosminexus TP1/NET/HSC 07-00

P-F1M64-32317 uCosminexus TP1/NET/NCSB 07-00

P-F1M64-32318 uCosminexus TP1/NET/OSAS-NIF 07-01

P-F1M64-3231B uCosminexus TP1/NET/Secondary Logical Unit - TypeP2 07-00

P-F1M64-3231C uCosminexus TP1/NET/TCP/IP 07-02

P-F1M64-3231D uCosminexus TP1/NET/High Availability 07-00

P-F1M64-3231U uCosminexus TP1/NET/User Datagram Protocol 07-00

R-1M45F-31 uCosminexus TP1/Web 07-00

For AIX 5L V5.3 and AIX V6.1

P-1M64-1111 uCosminexus TP1/Server Base(64) 07-03*

P-1M64-1311 uCosminexus TP1/FS/Direct Access(64) 07-03*

P-1M64-1411 uCosminexus TP1/FS/Table Access(64) 07-03*

P-1M64-1911 uCosminexus TP1/High Availability(64) 07-00

P-1M64-1L11 uCosminexus TP1/Extension 1(64) 07-00

For HP-UX 11i V1 (PA-RISC) and HP-UX 11i V2 (PA-RISC)

P-1B64-3F31 uCosminexus TP1/NET/High Availability 07-00

P-1B64-8531 uCosminexus TP1/Extension 1 07-00

P-1B64-8931 uCosminexus TP1/High Availability 07-00

R-18451-41K uCosminexus TP1/Client/W 07-00

R-18452-41K uCosminexus TP1/Server Base 07-00

R-18453-41K uCosminexus TP1/FS/Direct Access 07-00
R-18454-41K uCosminexus TP1/FS/Table Access 07-00
R-18455-41K uCosminexus TP1/Message Control 07-03*
R-18456-41K uCosminexus TP1/NET/Library 07-04*
R-18459-41K uCosminexus TP1/Offline Tester 07-00
R-1845A-41K uCosminexus TP1/Online Tester 07-00
R-1845C-41K uCosminexus TP1/Shared Table Access 07-00
R-1845D-41K uCosminexus TP1/Resource Manager Monitor 07-00
R-1845E-41K uCosminexus TP1/Multi 07-00
R-1845F-41K uCosminexus TP1/Web 07-00
R-F18455-411K uCosminexus TP1/Message Control/Tester 07-00
R-F18456-411K uCosminexus TP1/NET/User Agent 07-00
R-F18456-415K uCosminexus TP1/NET/XMAP3 07-01*
R-F18456-41CK uCosminexus TP1/NET/TCP/IP 07-02*
For HP-UX 11i V2 (IPF) and HP-UX 11i V3 (IPF)
P-1J64-3F21 uCosminexus TP1/NET/High Availability 07-00
P-1J64-4F11 uCosminexus TP1/NET/High Availability(64) 07-00
P-1J64-8521 uCosminexus TP1/Extension 1 07-00
P-1J64-8611 uCosminexus TP1/Extension 1(64) 07-00
P-1J64-8921 uCosminexus TP1/High Availability 07-00
P-1J64-8A11 uCosminexus TP1/High Availability(64) 07-00
P-1J64-C371 uCosminexus TP1/Message Queue 07-01
P-1J64-C571 uCosminexus TP1/Message Queue(64) 07-01
P-1J64-C871 uCosminexus TP1/Message Queue - Access(64) 07-00
R-18451-21J uCosminexus TP1/Client/W 07-02
R-18452-21J uCosminexus TP1/Server Base 07-03*
R-18453-21J uCosminexus TP1/FS/Direct Access 07-03*
R-18454-21J uCosminexus TP1/FS/Table Access 07-03*
R-18455-21J uCosminexus TP1/Message Control 07-03*
R-18456-21J uCosminexus TP1/NET/Library 07-04*
R-18459-21J uCosminexus TP1/Offline Tester 07-00
R-1845A-21J uCosminexus TP1/Online Tester 07-00
R-1845C-21J uCosminexus TP1/Shared Table Access 07-00
R-1845D-21J uCosminexus TP1/Resource Manager Monitor 07-00
R-1845E-21J uCosminexus TP1/Multi 07-00
R-1845F-21J uCosminexus TP1/Web 07-00
R-1B451-11J uCosminexus TP1/Client/W(64) 07-02
R-1B452-11J uCosminexus TP1/Server Base(64) 07-03*
R-1B453-11J uCosminexus TP1/FS/Direct Access(64) 07-03*
R-1B454-11J uCosminexus TP1/FS/Table Access(64) 07-03*
R-1B455-11J uCosminexus TP1/Message Control(64) 07-03*
R-1B456-11J uCosminexus TP1/NET/Library(64) 07-04*
R-F18455-211J uCosminexus TP1/Message Control/Tester 07-00
R-F18456-215J uCosminexus TP1/NET/XMAP3 07-01*

R-F18456-21CJ uCosminexus TP1/NET/TCP/IP 07-02*

R-F1B456-11CJ uCosminexus TP1/NET/TCP/IP(64) 07-02*

For Solaris 8, Solaris 9, and Solaris 10

P-9D64-3F31 uCosminexus TP1/NET/High Availability 07-00

P-9D64-8531 uCosminexus TP1/Extension 1 07-00

P-9D64-8931 uCosminexus TP1/High Availability 07-00

R-19451-216 uCosminexus TP1/Client/W 07-00

R-19452-216 uCosminexus TP1/Server Base 07-00

R-19453-216 uCosminexus TP1/FS/Direct Access 07-00

R-19454-216 uCosminexus TP1/FS/Table Access 07-00

R-19455-216 uCosminexus TP1/Message Control 07-03*

R-19456-216 uCosminexus TP1/NET/Library 07-04*

R-19459-216 uCosminexus TP1/Offline Tester 07-00

R-1945A-216 uCosminexus TP1/Online Tester 07-00

R-1945C-216 uCosminexus TP1/Shared Table Access 07-00

R-1945D-216 uCosminexus TP1/Resource Manager Monitor 07-00

R-1945E-216 uCosminexus TP1/Multi 07-00

R-F19456-2156 uCosminexus TP1/NET/XMAP3 07-01*

R-F19456-21C6 uCosminexus TP1/NET/TCP/IP 07-02*

For Red Hat Enterprise Linux AS 4 (AMD64 & Intel EM64T), Red Hat Enterprise Linux AS 4 (x86), Red Hat Enterprise Linux ES 4 (AMD64 & Intel EM64T), and Red Hat Enterprise Linux ES 4 (x86)

P-9S64-2161 uCosminexus TP1/Server Base 07-00

P-9S64-2351 uCosminexus TP1/FS/Direct Access 07-00

P-9S64-2451 uCosminexus TP1/FS/Table Access 07-00

P-9S64-2551 uCosminexus TP1/Client/W 07-00

P-9S64-3151 uCosminexus TP1/Message Control 07-00

P-9S64-3251 uCosminexus TP1/NET/Library 07-00

P-9S64-C371 uCosminexus TP1/Message Queue 07-01

P-F9S64-3251C uCosminexus TP1/NET/TCP/IP 07-00

P-F9S64-3251U uCosminexus TP1/NET/User Datagram Protocol 07-00

R-1845F-A15 uCosminexus TP1/Web 07-00

For Red Hat Enterprise Linux AS 4 (AMD64 & Intel EM64T), Red Hat Enterprise Linux AS 4 (x86), Red Hat Enterprise Linux ES 4 (AMD64 & Intel EM64T), Red Hat Enterprise Linux ES 4 (x86), Red Hat Enterprise Linux 5 (AMD/Intel 64), Red Hat Enterprise Linux 5 (x86), Red Hat Enterprise Linux 5 Advanced Platform (AMD/Intel 64), and Red Hat Enterprise Linux 5 Advanced Platform (x86)

P-9S64-2951 uCosminexus TP1/High Availability 07-00

P-9S64-8551 uCosminexus TP1/Extension 1 07-00

P-9S64-C771 uCosminexus TP1/Message Queue - Access 07-01

P-F9S64-3251D uCosminexus TP1/NET/High Availability 07-00

For Red Hat Enterprise Linux 5 (AMD/Intel 64), Red Hat Enterprise Linux 5 (x86), Red Hat Enterprise Linux 5 Advanced Platform (AMD/Intel 64), and Red Hat Enterprise Linux 5 Advanced Platform (x86)

P-9S64-2171 uCosminexus TP1/Server Base 07-03

P-9S64-2361 uCosminexus TP1/FS/Direct Access 07-03

P-9S64-2461 uCosminexus TP1/FS/Table Access 07-03

P-9S64-2561 uCosminexus TP1/Client/W 07-02

P-9S64-3161 uCosminexus TP1/Message Control 07-03*

P-9S64-3261 uCosminexus TP1/NET/Library 07-04*

P-9S64-C571 uCosminexus TP1/Message Queue 07-01

P-F9S64-32611 uCosminexus TP1/NET/User Agent 07-00

P-F9S64-3261C uCosminexus TP1/NET/TCP/IP 07-02

P-F9S64-3261U uCosminexus TP1/NET/User Datagram Protocol 07-00

For Red Hat Enterprise Linux 5 (AMD/Intel 64) and Red Hat Enterprise Linux 5 Advanced Platform (AMD/Intel 64)

P-9W64-2111 uCosminexus TP1/Server Base(64) 07-03

P-9W64-2311 uCosminexus TP1/FS/Direct Access(64) 07-03

P-9W64-2411 uCosminexus TP1/FS/Table Access(64) 07-03

P-9W64-2911 uCosminexus TP1/High Availability(64) 07-02

P-9W64-8511 uCosminexus TP1/Extension 1(64) 07-02

For Red Hat Enterprise Linux AS 4 (IPF)

P-9V64-2121 uCosminexus TP1/Server Base 07-00

P-9V64-2321 uCosminexus TP1/FS/Direct Access 07-00

P-9V64-2421 uCosminexus TP1/FS/Table Access 07-00

P-9V64-2521 uCosminexus TP1/Client/W 07-00

P-9V64-3121 uCosminexus TP1/Message Control 07-00

P-9V64-3221 uCosminexus TP1/NET/Library 07-00

P-9V64-C371 uCosminexus TP1/Message Queue(64) 07-01

P-9V64-C771 uCosminexus TP1/Message Queue - Access(64) 07-00

P-F9V64-3221C uCosminexus TP1/NET/TCP/IP 07-00

P-F9V64-3221U uCosminexus TP1/NET/User Datagram Protocol 07-00

For Red Hat Enterprise Linux AS 4 (IPF), Red Hat Enterprise Linux 5 (Intel Itanium), and Red Hat Enterprise Linux 5 Advanced Platform (Intel Itanium)

P-9V64-2921 uCosminexus TP1/High Availability 07-00

P-9V64-8521 uCosminexus TP1/Extension 1 07-00

P-F9V64-3221D uCosminexus TP1/NET/High Availability 07-00

For Red Hat Enterprise Linux 5 (Intel Itanium) and Red Hat Enterprise Linux 5 Advanced Platform (Intel Itanium)

P-9V64-2131 uCosminexus TP1/Server Base 07-02

P-9V64-2331 uCosminexus TP1/FS/Direct Access 07-02

P-9V64-2431 uCosminexus TP1/FS/Table Access 07-02

P-9V64-2531 uCosminexus TP1/Client/W 07-02

P-9V64-3131 uCosminexus TP1/Message Control 07-03*

P-9V64-3231 uCosminexus TP1/NET/Library 07-04*

P-F9V64-3231C uCosminexus TP1/NET/TCP/IP 07-02*

P-F9V64-3231U uCosminexus TP1/NET/User Datagram Protocol 07-00

For Windows 2000, Windows Server 2003, Windows Server 2003 x64 Editions, Windows Server 2003 R2, Windows Server 2003 R2 x64 Editions, Windows XP, Windows Vista, and Windows Vista x64

P-2464-2144 uCosminexus TP1/Client/P 07-02

For Windows 2000, Windows Server 2003, Windows Server 2003 x64 Editions, Windows Server 2003 R2, Windows Server 2003 R2 x64 Editions, and Windows XP

R-1845F-8134 uCosminexus TP1/Web 07-00

For Windows 2000, Windows Server 2003, Windows Server 2003 x64 Editions, Windows Server 2003 R2, Windows Server 2003 R2 x64 Editions, Windows XP, Windows Vista, Windows Vista x64, Windows Server 2008, and Windows Server 2008 x64

P-2464-7824 uCosminexus TP1/Client for .NET Framework 07-03

R-15451-21 uCosminexus TP1/Connector for .NET Framework 07-03

For Windows Server 2003, Windows Server 2003 x64 Editions, Windows Server 2003 R2, Windows Server 2003 R2 x64 Editions, Windows XP, Windows Vista, Windows Vista x64, Windows Server 2008, and Windows Server 2008 x64

P-2464-2274 uCosminexus TP1/Server Base 07-03*

P-2464-2374 uCosminexus TP1/FS/Direct Access 07-03*

P-2464-2474 uCosminexus TP1/FS/Table Access 07-03*

P-2464-2544 uCosminexus TP1/Extension 1 07-00

P-2464-3154 uCosminexus TP1/Message Control 07-03*

P-2464-3254 uCosminexus TP1/NET/Library 07-04*

P-2464-3354 uCosminexus TP1/Messaging 07-00

P-2464-C374 uCosminexus TP1/Message Queue 07-01

P-2464-C774 uCosminexus TP1/Message Queue - Access 07-00

P-F2464-3254C uCosminexus TP1/NET/TCP/IP 07-02*

R-15452-21 uCosminexus TP1/Extension for .NET Framework 07-00

R-1945B-24 uCosminexus TP1/LiNK 07-02

For Windows Server 2003, Windows Server 2003 x64 Editions, Windows Server 2003 R2, Windows Server 2003 R2 x64 Editions, and Windows XP

P-F2464-32545 uCosminexus TP1/NET/XMAP3 07-01*

For Windows Server 2003, Windows Server 2003 x64 Editions, Windows Server 2003 R2, Windows Server 2003 R2 x64 Editions, Windows Server 2008, and Windows Server 2008 x64

P-2464-2934 uCosminexus TP1/High Availability 07-00

P-F2464-3254D uCosminexus TP1/NET/High Availability 07-00

For Java VM

P-2464-7394 uCosminexus TP1/Client/J 07-02

P-2464-73A4 uCosminexus TP1/Client/J 07-02

This manual can be used for products other than the products shown above. For details, see the *Release Notes*.

This product was developed under a quality management system that has received ISO9001 and TickIT certification.

■ Trademarks

■ Restrictions

■ Edition history

Edition 1 (3000-3-D55(E)): June 2006

Edition 3 (3000-3-D55-30(E)): October 2010

■ Copyright

Summary of amendments

The following table lists changes in this manual (3000-3-D55-30(E)) and product changes related to this manual for uCosminexus TP1/Server Base 07-03, uCosminexus TP1/Server Base(64) 07-03, uCosminexus TP1/Message Control 07-03, uCosminexus TP1/Message Control(64) 07-03, uCosminexus TP1/NET/Library 07-04, uCosminexus TP1/NET/Library(64) 07-04.

Changes	Location
Explanations have been added about the maximum length of segments that can be sent or received.	*Message exchange (CBLDCMCF)* in *Chapter 2* *CBLDCMCF('EXECAP '), CBLDCMCF('RECEIVE '), CBLDCMCF('RECVSYNC'), CBLDCMCF('REPLY '), CBLDCMCF('SEND '), CBLDCMCF('SENDRECV'), CBLDCMCF('SENDSYNC')* *Data communication facility* in *Chapter 3* *RECEIVE - Receive a message, SEND - Send a message* *Service facility* in *Chapter 3* *SEND - Activate an application program*
A method for declaring unique names in the DATA DIVISION (communication description entry) specification has been added for the following data manipulation language programs: • `RECEIVE`: Receive a message • `RECEIVE`: Accept temporarily-stored data • `SEND`: Activate an application program • `SEND`: Update temporarily-stored data • `SEND`: Execute an operation command • `SEND`: Acquire a user journal	*Data communication facility* in *Chapter 3* *RECEIVE - Receive a message* *Service facility* in *Chapter 3* *RECEIVE - Accept temporarily-stored data* *SEND - Activate an application program* *SEND - Update temporarily-stored data* *SEND - Execute an operation command* *SEND - Acquire a user journal*

The following table lists changes in this manual (3000-3-D55-30(E)) and product changes related to this manual for uCosminexus TP1/Message Control 07-02 and uCosminexus TP1/NET/Library 07-03

Changes	Location
A library function can be used to delete application timer startup requests. To support this change, the following function has been added: • `CBLDCMCF('ADLTAP  ')`	*1.1.1*, *1.1.1(2)*, *1.1.1(3)* *Message exchange (CBLDCMCF)* in *Chapter 2* *CBLDCMCF('ADLTAP ')*
Library functions can be used to display the status of connections and to establish and release connections. To support this change, the following functions have been added: • `CBLDCMCF('TACTCN  ')` • `CBLDCMCF('TDCTCN  ')` • `CBLDCMCF('TLSCN   ')`	*1.1.1*, *1.1.1(2)*, *1.1.1(3)* *Message exchange (CBLDCMCF)* in *Chapter 2* *CBLDCMCF('TACTCN ')*, *CBLDCMCF('TDCTCN ')*, *CBLDCMCF('TLSCN ')*
A library function can be used to display the status of MCF communication services and application startup services. To support this change, the following function has been added: • `CBLDCMCF('TLSCOM  ')`	*1.1.1*, *1.1.1(2)*, *1.1.1(3)* *Message exchange (CBLDCMCF)* in *Chapter 2* *CBLDCMCF('TLSCOM ')*
Library functions can be used to display the status of logical terminals, to shut down logical terminals, to release logical terminals from shutdown status, and to delete the output queue of logical terminals. To support this change, the following functions have been added: • `CBLDCMCF('TACTLE  ')` • `CBLDCMCF('TDCTLE  ')` • `CBLDCMCF('TDLQLE  ')` • `CBLDCMCF('TLSLE   ')`	*1.1.1*, *1.1.1(2)*, *1.1.1(3)* *Message exchange (CBLDCMCF)* in *Chapter 2* *CBLDCMCF('TACTLE ')*, *CBLDCMCF('TDCTLE ')*, *CBLDCMCF('TDLQLE ')*, *CBLDCMCF('TLSLE ')*
A library function can be used to acquire the acceptance status of connection establishment requests. To support this change, the following function has been added: • `CBLDCMCF('TLSLN   ')`	*1.1.1*, *1.1.1(2)*, *1.1.1(3)* *Message exchange (CBLDCMCF)* in *Chapter 2* *CBLDCMCF('TLSLN ')*
Library functions can be used to start and stop acceptance of server-type connection establishment requests. To support this change, the following functions have been added: • `CBLDCMCF('TOFLN   ')` • `CBLDCMCF('TONLN   ')`	*1.1.1*, *1.1.1(2)*, *1.1.1(3)* *Message exchange (CBLDCMCF)* in *Chapter 2* *CBLDCMCF('TOFLN ')*, *CBLDCMCF('TONLN ')*
MHPs can use the facility for dynamic loading of service functions.	*1.2.1(3)*, *1.2.5(3)(d)*

In addition to the above changes, minor editorial corrections have been made.

The following table lists changes in the manual (3000-3-D55-20(E)) and product changes related to that manual for uCosminexus TP1/Server Base 07-02, uCosminexus TP1/Message Control 07-01, and uCosminexus TP1/NET/Library 07-01.

Changes
An audit log output function was added. With this addition, the `CBLDCADT('PRINT  ')` function was added.
A function that enables service functions to be loaded dynamically was added.
A function that allows the system to operate without using system journal files (journal fileless mode) was added. With this addition, status codes and return values for some functions were changed.
The description of the remote API facility was changed. With this change, status codes were changed or added.

The following table lists changes in the manual (3000-3-D55-20(E)) and product changes related to that manual for uCosminexus TP1/Server Base 07-01

Changes
Notes and status codes were added.

Preface

This manual explains how to create application programs which can be used with the following program products of OpenTP1:

- Distributed transaction processing facility TP1/Server Base
- Distributed application server TP1/LiNK

In this manual, an application program which is created by the user is abbreviated to a UAP (User Application Program).

Products described in this manual, other than those for which the manual is released, may not work with OpenTP1 Version 7 products. You need to confirm that the products you want to use work with OpenTP1 Version 7 products.

Intended readers

This manual is intended for programmers who create user application programs (UAPs) used with TP1/Server Base or TP1/LiNK.

Readers of this manual are assumed to have knowledge about operating systems, online systems, handling of the machine to be used, and the syntax of the COBOL language used for coding application programs.

This manual assumes that the reader has read the *OpenTP1 Programming Guide*.

Organization of this manual

This manual is organized into the following chapters and an appendix:

1. Creating Application Programs

This chapter explains the procedure for writing application programs to be used with the OpenTP1.

2. Syntax of OpenTP1 Programs for COBOL-UAP Creation Programs

This chapter explains the syntax of OpenTP1 programs for COBOL-UAP creation.

3. Syntax of OpenTP1 Programs for COBOL-UAP Creation Programs (DML Interface)

This chapter explains the syntax of the data manipulation language (DML) for OpenTP1 COBOL-UAP creation programs.

4. X/Open-compliant Application Programming Interface

This chapter explains the syntax of the library functions complying with X/Open.

5. Syntax of OpenTP1 COBOL-UAP Creation Programs (Association Status Notification)

This chapter explains the syntax of the COBOL-UAP creation program used by SPPs to process communication events and the format of received communication events.

6. Coding Samples

This chapter gives coding samples for OpenTP1 application programs.

7. Reference for Application Activation

This chapter explains the communication facilities in the message exchange configuration, focusing on user exit routines relating to application program activate and MCF event (ERREVT4) reference information.

A. Using OpenTP1 Remote Procedure Calls and XATMI-interfaced API Functions in Combination

This chapter explains the procedures for creating UAPs that use OpenTP1 remote procedure calls and XATMI-Interfaced API functions in combination.

Related publications

This manual is part of a related set of manuals. The manuals in the set are listed below (with the manual numbers):

OpenTP1 products

- *OpenTP1 Version 7 Description* (3000-3-D50(E))
- *OpenTP1 Version 7 Programming Guide* (3000-3-D51(E))
- *OpenTP1 Version 7 System Definition* (3000-3-D52(E))
- *OpenTP1 Version 7 Operation* (3000-3-D53(E))
- *OpenTP1 Version 7 Programming Reference C Language* (3000-3-D54(E))
- *OpenTP1 Version 7 Programming Reference COBOL Language* (3000-3-D55(E))
- *OpenTP1 Version 7 Messages* (3000-3-D56(E))
- *OpenTP1 Version 7 Tester and UAP Trace User's Guide* (3000-3-D57(E))
- *OpenTP1 Version 7 TP1/Client User's Guide TP1/Client/W, TP1/Client/P* (3000-3-D58(E))
- *OpenTP1 Version 7 TP1/Client User's Guide TP1/Client/J* (3000-3-D59(E))

- *OpenTP1 Version 7 TP1/LiNK User's Guide* (3000-3-D60(E))[1]
- *OpenTP1 Version 7 Protocol TP1/NET/TCP/IP* (3000-3-D70(E))
- *OpenTP1 Version 7 TP1/Message Queue User's Guide* (3000-3-D90(E))[1]
- *OpenTP1 Version 7 TP1/Message Queue Messages* (3000-3-D91(E))[1]
- *OpenTP1 Version 7 TP1/Message Queue Application Programming Guide* (3000-3-D92(E))[1]
- *OpenTP1 Version 7 TP1/Message Queue Application Programming Reference* (3000-3-D93(E))[1]

Other OpenTP1 products

- *TP1/Web User's Guide and Reference* (3000-3-D62(E))[1]

Other related products

- *Indexed Sequential Access Method ISAM* (3000-3-046(E))
- *XP/W* (3000-3-047(E))
- *Extended Mapping Service 2/Workstation XMAP2/W DESCRIPTION/USER'S GUIDE* (3000-7-421(E))
- *SEWB 3 General Information* (3000-7-450(E))
- *Job Management Partner 1/Base User's Guide* (3020-3-K06(E))
- *Job Management Partner 1/Base Messages* (3020-3-K07(E))
- *Job Management Partner 1/Base Software Developer's Guide* (3020-3-K08(E))

For OpenTP1 protocol manuals, please check whether English versions are available.

Note

[1] If you want to use this manual, confirm that it has been published. (Some of these manuals might not have been published yet.)

Conventions: Abbreviations

This manual uses the following abbreviations for product names:

Abbreviation	Full name or meaning
AIX	AIX 5L V5.1
	AIX 5L V5.2
	AIX 5L V5.3

Abbreviation			Full name or meaning
			AIX V6.1
Client .NET		TP1/Client for .NET Framework	uCosminexus TP1/Client for .NET Framework
Connector .NET		TP1/Connector for .NET Framework	uCosminexus TP1/Connector for .NET Framework
DPM			JP1/ServerConductor/Deployment Manager
HI-UX/WE2			HI-UX/workstation Extended Version 2
HP-UX	HP-UX (IPF)		HP-UX 11i V2 (IPF)
			HP-UX 11i V3 (IPF)
	HP-UX (PA-RISC)		HP-UX 11i V1 (PA-RISC)
			HP-UX 11i V2 (PA-RISC)
IPF			Itanium(R) Processor Family
Java			Java^{TM}
JP1	JP1/AJS2	JP1/AJS2 - Agent	JP1/Automatic Job Management System 2 - Agent
		JP1/AJS2 - Manager	JP1/Automatic Job Management System 2 - Manager
		JP1/AJS2 - View	JP1/Automatic Job Management System 2 - View
	JP1/AJS2 - Scenario Operation	JP1/AJS2 - Scenario Operation Manager	JP1/Automatic Job Management System 2 - Scenario Operation Manager
		JP1/AJS2 - Scenario Operation View	JP1/Automatic Job Management System 2 - Scenario Operation View
		JP1/NETM/Audit	JP1/NETM/Audit - Manager
Linux			Linux(R)
Linux (AMD64/Intel EM64T/x86)			Red Hat Enterprise Linux AS 4 (AMD64 & Intel EM64T)
			Red Hat Enterprise Linux AS 4 (x86)
			Red Hat Enterprise Linux ES 4 (AMD64 & Intel EM64T)
			Red Hat Enterprise Linux ES 4 (x86)
			Red Hat Enterprise Linux 5 (AMD/Intel 64)
			Red Hat Enterprise Linux 5 (x86)

Abbreviation		Full name or meaning
		Red Hat Enterprise Linux 5 Advanced Platform (AMD/Intel 64)
		Red Hat Enterprise Linux 5 Advanced Platform (x86)
Linux (IPF)		Red Hat Enterprise Linux AS 4 (IPF)
		Red Hat Enterprise Linux 5 (Intel Itanium)
		Red Hat Enterprise Linux 5 Advanced Platform (Intel Itanium)
MS-DOS		Microsoft$^{(R)}$ MS-DOS$^{(R)}$
NETM/DM		JP1/NETM/DM Client
		JP1/NETM/DM Manager
		JP1/NETM/DM SubManager
Oracle		Oracle 10g
		Oracle9i
Solaris		Solaris 8
		Solaris 9
		Solaris 10
TP1/Client	TP1/Client/J	uCosminexus TP1/Client/J
	TP1/Client/P	uCosminexus TP1/Client/P
	TP1/Client/W	uCosminexus TP1/Client/W
		uCosminexus TP1/Client/W(64)
TP1/EE		uCosminexus TP1/Server Base Enterprise Option
		uCosminexus TP1/Server Base Enterprise Option(64)
TP1/Extension 1		uCosminexus TP1/Extension 1
		uCosminexus TP1/Extension 1(64)
TP1/FS/Direct Access		uCosminexus TP1/FS/Direct Access
		uCosminexus TP1/FS/Direct Access(64)
TP1/FS/Table Access		uCosminexus TP1/FS/Table Access

Abbreviation		Full name or meaning
		uCosminexus TP1/FS/Table Access(64)
TP1/High Availability		uCosminexus TP1/High Availability
		uCosminexus TP1/High Availability(64)
TP1/LiNK		uCosminexus TP1/LiNK
TP1/Message Control		uCosminexus TP1/Message Control
		uCosminexus TP1/Message Control(64)
TP1/Message Control/Tester		uCosminexus TP1/Message Control/Tester
TP1/Message Queue		uCosminexus TP1/Message Queue
		uCosminexus TP1/Message Queue(64)
TP1/Message Queue - Access		uCosminexus TP1/Message Queue - Access
		uCosminexus TP1/Message Queue - Access(64)
TP1/Messaging		uCosminexus TP1/Messaging
TP1/Multi		uCosminexus TP1/Multi
TP1/NET/HDLC		uCosminexus TP1/NET/HDLC
TP1/NET/High Availability		uCosminexus TP1/NET/High Availability
		uCosminexus TP1/NET/High Availability(64)
TP1/NET/HSC		uCosminexus TP1/NET/HSC
TP1/NET/Library		uCosminexus TP1/NET/Library
		uCosminexus TP1/NET/Library(64)
TP1/NET/NCSB		uCosminexus TP1/NET/NCSB
TP1/NET/OSAS-NIF		uCosminexus TP1/NET/OSAS-NIF
TP1/NET/OSI-TP		uCosminexus TP1/NET/OSI-TP
TP1/NET/SLU - TypeP2	TP1/NET/ Secondary Logical Unit - TypeP2	uCosminexus TP1/NET/Secondary Logical Unit - TypeP2
TP1/NET/TCP/IP		uCosminexus TP1/NET/TCP/IP
		uCosminexus TP1/NET/TCP/IP(64)
TP1/NET/UDP		uCosminexus TP1/NET/User Datagram Protocol

Abbreviation	Full name or meaning
TP1/NET/User Agent	uCosminexus TP1/NET/User Agent
TP1/NET/X25	uCosminexus TP1/NET/X25
TP1/NET/X25-Extended	uCosminexus TP1/NET/X25-Extended
TP1/NET/XMAP3	uCosminexus TP1/NET/XMAP3
TP1/Offline Tester	uCosminexus TP1/Offline Tester
TP1/Online Tester	uCosminexus TP1/Online Tester
TP1/Resource Manager Monitor	uCosminexus TP1/Resource Manager Monitor
TP1/Server Base	uCosminexus TP1/Server Base
	uCosminexus TP1/Server Base(64)
TP1/Shared Table Access	uCosminexus TP1/Shared Table Access
TP1/Web	uCosminexus TP1/Web
Windows 2000	Microsoft$^{(R)}$ Windows$^{(R)}$ 2000 Advanced Server Operating System
	Microsoft$^{(R)}$ Windows$^{(R)}$ 2000 Datacenter Server Operating System
	Microsoft$^{(R)}$ Windows$^{(R)}$ 2000 Professional Operating System
	Microsoft$^{(R)}$ Windows$^{(R)}$ 2000 Server Operating System
Windows Server 2003	Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003, Datacenter Edition
	Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003, Enterprise Edition
	Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003, Standard Edition
Windows Server 2003 R2	Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003 R2, Enterprise Edition
	Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003 R2, Standard Edition
Windows Server 2003 x64 Editions	Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003, Datacenter x64 Edition
	Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003, Enterprise x64 Edition
	Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003, Standard x64 Edition

Abbreviation	Full name or meaning
Windows Server 2003 R2 x64 Editions	Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition
	Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition
Windows Server 2008	Microsoft(R) Windows Server(R) 2008 Datacenter (x86)
	Microsoft(R) Windows Server(R) 2008 Enterprise (x86)
	Microsoft(R) Windows Server(R) 2008 Standard (x86)
Windows Server 2008 x64 Editions	Microsoft(R) Windows Server(R) 2008 Datacenter (x64)
	Microsoft(R) Windows Server(R) 2008 Enterprise (x64)
	Microsoft(R) Windows Server(R) 2008 Standard (x64)
Windows Vista	Microsoft(R) Windows Vista(R) Business (x86)
	Microsoft(R) Windows Vista(R) Enterprise (x86)
	Microsoft(R) Windows Vista(R) Ultimate (x86)
Windows Vista x64 Editions	Microsoft(R) Windows Vista(R) Business (x64)
	Microsoft(R) Windows Vista(R) Enterprise (x64)
	Microsoft(R) Windows Vista(R) Ultimate (x64)
Windows XP	Microsoft(R) Windows(R) XP Professional Operating System

- The term Windows is used to indicate Windows Server 2003, Windows XP and Windows Vista if the difference in functions among them need not be considered.
- The term UNIX is used to indicate AIX, HP-UX, Linux, and Solaris.

Conventions: Acronyms

This manual also uses the following acronyms:

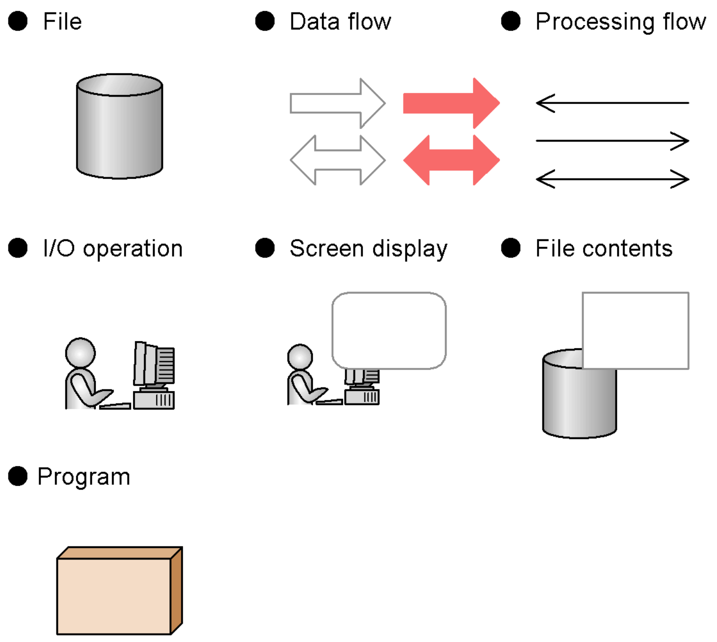
Acronym	Full name or meaning
ACL	Access Control List
ANSI	American National Standards Institute
AP	Application Program

Acronym	Full name or meaning
API	Application Programming Interface
C/S	Client/Server
CPU	Central Processing Unit
CRM	Communication Resource Manager
CUP	Client User Program
DAM	Direct Access Method
DBMS	Database Management System
DML	Data Manipulation Language
DNS	Domain Name System
FEP	Front End Processor
GUI	Graphical User Interface
HA	High Availability
ISAM	Indexed Sequential Access Method
IST	Internode Shared Table
LAN	Local Area Network
MCF	Message Control Facility
MHP	Message Handling Program
MQA	Message Queue Access
MQI	Message Queue Interface
OS	Operating System
OSI	Open Systems Interconnection
OSI TP	Open Systems Interconnection Transaction Processing
PC	Personal Computer
PRF	Performance
RM	Resource Manager
RPC	Remote Procedure Call
SPP	Service Providing Program

Acronym	Full name or meaning
SUP	Service Using Program
TAM	Table Access Method
TCP/IP	Transmission Control Protocol/Internet Protocol
UAP	User Application Program
UOC	User Own Coding
VM	Virtual Machine
WAN	Wide Area Network
WS	Workstation

Conventions: Diagrams

This manual uses the following conventions in diagrams:



Conventions: Differences between JIS and ASCII keyboards

The JIS code and ASCII code keyboards are different in the input characters represented by the following codes. In this manual, the use of a JIS keyboard is assumed for these characters.

Code	JIS keyboard	ASCII keyboard
$(5c)_{16}$	¥ (yen symbol)	\ (backslash)
$(7e)_{16}$	‾ (overline)	~ (tilde)

Conventions: Fonts and symbols

The following table explains the fonts used in this manual:

Font	Convention
Bold	**Bold** type indicates text on a window, other than the window title. Such text includes menus, menu options, buttons, radio box options, or explanatory labels. For example: • From the **File** menu, choose **Open**. • Click the **Cancel** button. • In the **Enter name** entry box, type your name.
Italics	*Italics* are used to indicate a placeholder for some actual text to be provided by the user or system. For example: • Write the command as follows: `copy` *source-file target-file* • The following message appears: `A file was not found. (file =` *file-name*`)` *Italics* are also used for emphasis. For example: • Do *not* delete the configuration file.
`Code font`	A `code font` indicates text that the user enters without change, or text (such as messages) output by the system. For example: • At the prompt, enter `dir`. • Use the `send` command to send mail. • The following message is displayed: `The password is incorrect.`
`SD`	Bold code-font characters indicate the abbreviation for a command.
<u>`perm`</u>	Underlined characters indicate the default value.

The following table explains the symbols used in this manual:

Symbol	Convention
|	In syntax explanations, a vertical bar separates multiple items, and has the meaning of OR. For example: `A|B|C` means A, or B, or C.
{ }	In syntax explanations, curly brackets indicate that only one of the enclosed items is to be selected. For example: `{A|B|C}` means only one of A, or B, or C.

Symbol	Convention
`[ ]`	In syntax explanations, square brackets indicate that the enclosed item or items are optional. For example: `[A]` means that you can specify `A` or nothing. `[B|C]` means that you can specify `B`, or `C`, or nothing.
`...`	In coding, an ellipsis (...) indicates that one or more lines of coding are not shown for purposes of brevity. In syntax explanations, an ellipsis indicates that the immediately preceding item can be repeated as many times as necessary. For example: `A, B, B, ...` means that, after you specify `A, B,` you can specify B as many times as necessary.
Δ	Indicates a space character.
~	The item preceding this symbol must be specified according to the rule given in the angle brackets (< >) following this symbol.
< >	Information between these symbols indicates the syntax of the item.

Conventions for permitted characters

In most cases, only the following characters are permitted as syntax elements (if other characters are permitted, the manual will state this explicitly):

Type	Definition
Upper-case alphabetic characters	`A` to `Z`
Lower-case alphabetic characters	`a` to `z`
Alphabetic characters	`A` to `Z`, `a` to `z`
Numeric characters	`0` to `9`
Alphanumeric characters	`A` to `Z`, `a` to `z`, `0` to `9`
Symbols	`!`, `#`, `$`, `%`, `&`, `'`, `(`, `)`, `*`, `+`, `-`, `.`, `/`, `:`, `;`, `<`, `=`, `>`, `?`, `@`, `[`, `\`, `]`, `^`, `_`, `` ` ``, `{`, `|`, `}`, `~`
Hexadecimal	Numeric values `0` to `9`, `A` to `F`, and `a` to `f`
Pathname	Symbolic names, slashes (`/`), and periods (`.`), depending on the operating system being used.

Conditions for values to be specified for data areas

The table below lists the conditions for values to be specified for data areas.

Value to be specified	Condition
Service group name	Must be an ASCII character string of up to 31 bytes. Note that null characters, blanks, at marks (`@`), and periods cannot be used. When a service group name is specified in a data area, it must end with a blank. This blank will not be included in the length of the character string.
Service name	Must be an ASCII character string of up to 31 bytes. Note that null and blank characters cannot be used. When a service name is specified in a data area, it must end with a blank. This blank will not be included in the length of the character string.
Physical file name	Must be a pathname consisting of the special file name followed by a name of 14 or less bytes. The entire pathname must not exceed 63 characters.
Logical file name	Must be an alphanumeric character string of 1 to 8 bytes that begins with an alphabetic character.

Conventions: KB, MB, GB, and TB

This manual uses the following conventions:

- 1 KB (kilobyte) is 1,024 bytes.
- 1 MB (megabyte) is 1,024^2 bytes.
- 1 GB (gigabyte) is 1,024^3 bytes.
- 1 TB (terabyte) is 1,024^4 bytes.

Conventions: Platform-specific notational differences

For the Windows version of OpenTP1, there are some notational differences from the description in the manual. The following table describes these differences.

Item	Description in the manual	Change to:
Environment variable	`$`*aaaaaa* Example: `$DCDIR`	`%`*aaaaaa*`%` Example: `%DCDIR%`
Path name separator	Colon (`:`)	Semicolon (`;`)
Directory name separator	Slash (`/`)	Backslash (`\`)
Absolute path name	A path from the root directory Example: `/tmp`	A path name from a drive letter and the root directory Example: `C:\tmp`
Executable file name	File name only (without an extension) Example: `mcfmngrd`	File name with an extension Example: `mcfmngrd.exe`

Item	Description in the manual	Change to:
`make` command	`make`	`nmake`

Conventions: Version numbers

The version numbers of Hitachi program products are usually written as two sets of two digits each, separated by a hyphen. For example:

- Version 1.00 (or 1.0) is written as 01-00.
- Version 2.05 is written as 02-05.
- Version 2.50 (or 2.5) is written as 02-50.
- Version 12.25 is written as 12-25.

The version number might be shown on the spine of a manual as *Ver. 2.00,* but the same version number would be written in the program as *02-00*.

Acknowledgments

Quotations from X/Open CAE Specification Distributed Transaction Processing: The XATMI Specification published by X/Open Company Limited

The following section comes from Chapter *7. COBOL Reference Manual Pages* of the above document.

Chapter *4. X/Open-Compliant Application Programming Interface*

4.1 XATMI-Interfaced Application Programming Interface (TP~)

Quotations from X/Open CAE Specification Distributed Transaction Processing: The TX (Transaction Demarcation) Specification published by X/Open Company Limited

The following section comes from Chapter *6. COBOL Reference Manual Pages* of the above document.

Chapter *4. X/Open-Compliant Application Programming Interface*

4.2 TX-Interfaced Application Programming Interface (TX~)

COBOL

COBOL was developed by CODASYL (the Conference on Data Systems Languages). Of the OpenTP1 application programming interface specifications, the data manipulation language (DML) specification was developed by relying on the communication section in CODASYL COBOL (1981) as well as the RECEIVE, SEND, COMMIT, and ROLLBACK statements and adding original specifications and interpretations made by Hitachi, Ltd. The publisher of this manual expresses acknowledgment to the original developer and presents the following acknowledgment statement as requested by CODASYL. This statement is quoted from

Important note on this manual

Please check the availability of the products and manuals for HAmonitor, ServerConductor/DeploymentManager, Cosminexus, and Job Management Partner 1/ Automatic Job Management System 2.

Contents

List of figures

List of tables

Chapter

1. Creating Application Programs

This chapter outlines how to write OpenTP1 application programs in the COBOL language.

This chapter contains the following sections:

1.1 Coding application program

1.1.1 Relationship between UAPs and programs

Table 1-1 gives the relationship between OpenTP1 facilities for use with OpenTP1 UAPs and programs for creating UAPs in COBOL.

Table 1-1: Relationship between OpenTP1 facilities and programs for COBOL-UAP creation

OpenTP1 facility		Program called with CALL statement
System operation management	Execute an operation command.	`CBLDCADM('COMMAND ')`
	Report the completion of user server start processing.	`CBLDCADM('COMPLETE')`
	Report the status of a user server.	`CBLDCADM('STATUS  ')`
DAM file service	Close a logical file.	`CBLDCDAM('DCDAMSVC','CLOS')`
	Terminate using an unrecoverable DAM file.	`CBLDCDAM('DCDAMSVC','END ')`
	Shut down a logical file.	`CBLDCDAM('DCDAMSVC','HOLD')`
	Open a logical file.	`CBLDCDAM('DCDAMSVC','OPEN')`
	Input a logical file block.	`CBLDCDAM('DCDAMSVC','READ')`
	Update a logical file block.	`CBLDCDAM('DCDAMSVC','REWT')`
	Release a logical file from the shutdown state.	`CBLDCDAM('DCDAMSVC','RLES')`
	Reference the status of a logical file.	`CBLDCDAM('DCDAMSVC','STAT')`
	Start using an unrecoverable DAM file.	`CBLDCDAM('DCDAMSVC','STRT')`
	Output a logical file block.	`CBLDCDAM('DCDAMSVC','WRIT')`
	Seek a physical file block.	`CBLDCDMB('DCDAMINT','BSEK')`
	Close a physical file.	`CBLDCDMB('DCDAMINT','CLOS')`
	Allocate a physical file.	`CBLDCDMB('DCDAMINT','CRAT')`
	Input directly a physical file block.	`CBLDCDMB('DCDAMINT','DGET')`
	Output directly a physical file block.	`CBLDCDMB('DCDAMINT','DPUT')`

OpenTP1 facility		Program called with CALL statement
	Input a physical file block.	`CBLDCDMB('DCDAMINT','GET ')`
	Open a physical file.	`CBLDCDMB('DCDAMINT','OPEN')`
	Output a physical file block.	`CBLDCDMB('DCDAMINT','PUT ')`
IST service	Close an internode shared table.	`CBLDCIST('CLOS')`
	Open an internode shared table.	`CBLDCIST('OPEN')`
	Input an internode shared table record.	`CBLDCIST('READ')`
	Output an internode shared table record.	`CBLDCIST('WRIT')`
User journal acquisition	Acquire a user journal.	`CBLDCJNL('UJPUT ')`
Journal data editing	Close the jnlrput output file.	`CBLDCJUP('CLOSERPT')`
	Open the jnlrput output file.	`CBLDCJUP('OPENRPT ')`
	Input journal data of the jnlrput output file.	`CBLDCJUP('RDGETRPT')`
Look for resources	Enable locking of a resource.	`CBLDCLCK('GET  ')`
	Release all the resources from lock.	`CBLDCLCK('RELALL ')`
	Release resource from lock specified by name.	`CBLDCLCK('RELNAME ')`
Audit log output	Output audit log data.	`CBLDCADT('PRINT   ')`
Output message log	Output message log.	`CBLDCLOG('PRINT ')`
Message exchange processing	Report the application information.	`CBLDCMCF('APINFO ')`
	Close the MCF environment.	`CBLDCMCF('CLOSE ')`
	Commit an MHP.	`CBLDCMCF('COMMIT ')`
	Terminate continuous-inquiry-response processing.	`CBLDCMCF('CONTEND ')`
	Activate an application program.	`CBLDCMCF('EXECAP ')`

OpenTP1 facility		Program called with CALL statement
	Start an MHP service.	`CBLDCMCF('MAINLOOP')`
	Open the MCF environment.	`CBLDCMCF('OPEN   ')`
	Receive a message.	`CBLDCMCF('RECEIVE ')`
	Receive a synchronous message.	`CBLDCMCF('RECVSYNC')`
	Send a response message.	`CBLDCMCF('REPLY  ')`
	Resend a message.	`CBLDCMCF('RESEND ')`
	Enable MHP rollback.	`CBLDCMCF('ROLLBACK')`
	Send a message.	`CBLDCMCF('SEND   ')`
	Exchange a synchronous message.	`CBLDCMCF('SENDRECV')`
	Send a synchronous message.	`CBLDCMCF('SENDSYNC')`
	Accept temporary-stored data.	`CBLDCMCF('TEMPGET')`
	Update temporary-stored data.	`CBLDCMCF('TEMPPUT')`
	Cancel user timer monitoring.	`CBLDCMCF('TIMERCAN')`
	Set user timer monitoring.	`CBLDCMCF('TIMERSET')`
Performance verification trace	Report the sequential number for an acquired performance verification trace.	`CBLDCPRF('PRFGETN ')`
	Acquire user-specific performance verification traces.	`CBLDCPRF('PRFPUT ')`

Remote API facility	Establish a connection with a RAP-processing listener.	`CBLDCRAP('CONNECT ')`
		`CBLDCRAP('CONNECTX')`

	Release a connection with a RAP-processing listener.	`CBLDCRAP('DISCNCT ')`
Remote procedure call	Request a remote service.	`CBLDCRPC('CALL    ')`
	Terminate an application program.	`CBLDCRPC('CLOSE   ')`
	Report data to CUP unidirectionally.	`CBLDCRPC('CLTSEND ')`
	Reject the receiving of processing results.	`CBLDCRPC('DISCARDF ')`

OpenTP1 facility		Program called with CALL statement
	Reject acceptance of specific processing results.	CBLDCRPC('DISCARDS')
	Acquire the node address of a client UAP.	CBLDCRPC('GETCLADR')
	Acquire the descriptor of an asynchronous response-type RPC request which has encountered an error	CBLDCRPC('GETERDES')
	Acquire the node address of a gateway.	CBLDCRPC('GETGWADR')
	Reference the schedule priority of a service request.	CBLDCRPC('GETSVPRI ')
	Reference the service response waiting interval.	CBLDCRPC('GETWATCH')
	Start an application program.	CBLDCRPC('OPEN ')
	Receive processing results in asynchronous mode.	CBLDCRPC('POLLANYR')
	Set a schedule priority of a service request.	CBLDCRPC('SETSVPRI ')
	Update the service response waiting interval.	CBLDCRPC('SETWATCH ')
	Retry a service program.	CBLDCRPC('SVRETRY ')
	Start an SPP service.	CBLDCRSV('MAINLOOP')
Real-time statistical information service	Acquire real-time statistical information for arbitrary section.	CBLDCRTS('RTSPUT ')
TAM file service#	Delete a TAM table record.	CBLDCTAM('ERS ')('ERSR')('ZRS ')('ZRSR')
	Input a TAM table record.	CBLDCTAM('FxxR')('FxxU')('VxxR')('VxxU')
	Acquire TAM table status.	CBLDCTAM('GST ')
	Acquire TAM table information.	CBLDCTAM('INFO')
	Update/add a TAM table record.	CBLDCTAM('MFY ')('MFYS')('STR ')('WFY ')('WFYS')('YTR ')

OpenTP1 facility		Program called with CALL statement
Transaction control	Start a transaction.	`CBLDCTRN('BEGIN   ')`
	Enable commitment in chained mode.	`CBLDCTRN('C-COMMIT')`
	Enable rollback in chained mode.	`CBLDCTRN('C-ROLL  ')`
	Report the information about the current transaction.	`CBLDCTRN('INFO    ')`
	Enable commitment in unchained mode.	`CBLDCTRN('U-COMMIT')`
	Enable rollback in unchained mode.	`CBLDCTRN('U-ROLL  ')`
Online tester management	Report the test status of a user server.	`CBLDCUTO('T-STATUS')`

#: The APIs of the following TAM file services are not supported in COBOL language:

- Open a TAM table.
- Close a TAM table.
- Cancel the input of a TAM table record.

(1) Facilities and programs available with SUPs

Table 1-2 lists the facilities which can be used with SUPs and their request codes.

Table 1-2: Facilities available with SUPs and their request codes

Facility available with SUP	Names of facilities available with SUPs and request codes specified as data name at the beginning of COBOL-UAP creation program		SUP operating conditions	
			Outside	Inside
System operation management	Execute an operation command.	`'COMMAND'`	Y	Y
	Report the completion of user server start processing.	`'COMPLETE'`	Y	N
	Report the status of a user server.	`'STATUS  '`	Y	Y
DAM file service	Close a logical file.	`'DCDAMSVC', 'CLOS'`	Y	Y

Facility available with SUP	Names of facilities available with SUPs and request codes specified as data name at the beginning of COBOL-UAP creation program		SUP operating conditions	
			Outside	Inside
	Terminate using an unrecoverable DAM file.	`'DCDAMSVC', 'END '`	Y	Y
	Shut down a logical file.	`'DCDAMSVC', 'HOLD'`	Y	Y
	Open a logical file.	`'DCDAMSVC', 'OPEN'`	Y	Y
	Input a logical file block.	`'DCDAMSVC', 'READ'`	Y	Y
	Update a logical file block.	`'DCDAMSVC', 'REWT'`	(Y)	Y
	Release a logical file from the shutdown state.	`'DCDAMSVC', 'RLES'`	Y	Y
	Reference the status of a logical file.	`'DCDAMSVC', 'STAT'`	Y	Y
	Start using an unrecoverable DAM file.	`'DCDAMSVC', 'STRT'`	Y	Y
	Output a logical file block.	`'DCDAMSVC', 'WRIT'`	(Y)	Y
IST service	Close an internode shared table.	`'DCISTSVC', 'CLOS'`	Y	Y
	Open an internode shared table.	`'DCISTSVC', 'OPEN'`	Y	Y
	Input an internode shared table record.	`'DCISTSVC', 'READ'`	Y	Y
	Output an internode shared table record.	`'DCISTSVC', 'WRIT'`	Y	Y
User journal acquisition	Acquire a user journal.	`'UJPUT   '`	Y	Y
Lock for resources	Enable locking of a resource.	`'GET   '`	N	Y
	Release all the resources from lock.	`'RELALL '`	N	Y
	Release resource from lock specified by name.	`'RELNAME'`	N	Y
Audit log output	Output audit log data.	`'PRINT   '`	Y	Y

Facility available with SUP	Names of facilities available with SUPs and request codes specified as data name at the beginning of COBOL-UAP creation program		SUP operating conditions	
			Outside	Inside
Output message log	Output message log.	`'PRINT   '`	Y	Y
Performance verification trace	Report the sequential number for an acquired performance verification trace.	`'PRFGETN'`	Y	Y
	Acquire user-specific performance verification traces.	`'PRFPUT  '`	Y	Y
Remote API facility	Establish a connection with a RAP- processing listener.	`'CONNECT'` `'CONNECTX'`	Y	N
	Release a connection with a RAP- processing listener.	`'DISCNCT'`	Y	N
Remote procedure call	Request a remote service.	`'CALL    '`	Y	Y
	Terminate an application program.	`'CLOSE   '`	Y	N
	Reject the receiving of processing results.	`'DISCARDF'`	Y	Y
	Reject acceptance of specific processing results.	`'DISCARDS'`	Y	Y
	Acquire the descriptor of an asynchronous response-type RPC request which has encountered an error.	`'GETERDES'`	Y	Y
	Reference the schedule priority of a service request.	`'GETSVPRI'`	Y	Y
	Reference the service response waiting interval.	`'GETWATCH'`	Y	Y
	Start an application program.	`'OPEN    '`	Y	N
	Receive processing results in asynchronous mode.	`'POLLANYR'`	Y	Y
	Set a schedule priority of a service request.	`'SETSVPRI'`	Y	Y

Facility available with SUP	**Names of facilities available with SUPs and request codes specified as data name at the beginning of COBOL-UAP creation program**		**SUP operating conditions**	
			Outside	**Inside**
	Update the service response waiting interval.	'SETWATCH'	Y	Y
Real-time statistical information service	Acquire real-time statistical information for arbitrary section.	'RTSPUT '	Y	Y
TAM file service	Delete a TAM table record.	'ERS'/'ERSR'/'ZRS '/'ZRSR'	N	Y
	Input a TAM table record.	'FxxR'/'FxxU'/ 'VxxR'/'VxxU'	N	Y
	Acquire TAM table status.	'GST'	Y	Y
	Acquire TAM table information.	'INFO '	Y	Y
	Update/add a TAM table record.	'MFY'/'MFYS'/'STR '/'WFY'/'WFYS'/ 'YTR'	N	Y
Transaction control	Start a transaction.	'BEGIN'	Y	N
	Enable commitment in chained mode.	'C-COMMIT'	N	Y
	Enable rollback in chained mode.	'C-ROLL '	N	Y
	Report the information about the current transaction.	'INFO '	Y	Y
	Enable commitment in unchained mode.	'U-COMMIT'	N	Y
	Enable rollback in unchained mode.	'U-ROLL '	N	Y
Online tester management	Report the test status of a user server.	'T-STATUS'	Y	Y

Legend:

Outside: Outside the transaction processing range

Inside: Inside the transaction processing range

Y: Can be used with SUPs.

(Y): Can be used only when accessing an unrecoverable DAM file.

N: Cannot be used with SUPs.

(2) Facilities and programs available with SPPs

Table 1-3 lists the facilities which can be used with SPPs and their request codes.

Table 1-3: Facilities available with SPPs and their request codes

Facility available with SPP	**Names of facilities available with SUPs and request codes specified as data name at the beginning of COBOL-UAP creation program**		**SPP operating conditions**		
			Outside	**Inside**	
				Root	**Not root**
System operation management	Execute an operation command.	`'COMMAND '`	Y	Y	Y
	Report the status of a user server.	`'STATUS  '`	Y	Y	Y
DAM file service	Close a logical file.	`'DCDAMSVC','CLOS'`	Y	Y	Y
	Terminate using an unrecoverable DAM file.	`'DCDAMSVC','END '`	Y	Y	Y
	Shut down a logical file.	`'DCDAMSVC','HOLD'`	Y	Y	Y
	Open a logical file.	`'DCDAMSVC','OPEN'`	Y	Y	Y
	Input a logical file block.	`'DCDAMSVC','READ'`	Y	Y	Y
	Update a logical file block.	`'DCDAMSVC','REWT'`	(Y)	Y	Y
	Release a logical file from the shutdown state.	`'DCDAMSVC','RLES'`	Y	Y	Y
	Reference the status of a logical file.	`'DCDAMSVC','STAT'`	Y	Y	Y
	Start using an unrecoverable DAM file.	`'DCDAMSVC','STRT'`	Y	Y	Y
	Output a logical file block.	`'DCDAMSVC','WRIT'`	(Y)	Y	Y

Facility available with SPP	**Names of facilities available with SUPs and request codes specified as data name at the beginning of COBOL-UAP creation program**		**SPP operating conditions**		
			Outside	**Inside**	
				Root	**Not root**
IST service	Close an internode shared table.	`'DCISTSVC','CLOS'`	Y	Y	Y
	Open an internode shared table.	`'DCISTSVC','OPEN'`	Y	Y	Y
	Input an internode shared table record.	`'DCISTSVC','READ'`	Y	Y	Y
	Output an internode shared table record.	`'DCISTSVC','WRIT'`	Y	Y	Y
User journal acquisition	Acquire a user journal.	`'UJPUT   '`	Y	Y	Y
Lock for resources	Enable locking of a resource.	`'GET     '`	N	Y	Y
	Release all the resources from lock.	`'RELALL  '`	N	Y	Y
	Release resource from lock specified by name.	`'RELNAME '`	N	Y	Y
Audit log output	Output audit log data.	`'PRINT   '`	Y	Y	Y
Output message log	Output message log.	`'PRINT   '`	Y	Y	Y
Message exchange processing	Close the MCF environment.	`'CLOSE   '`	O	N	N
	Activate an application program.	`'EXECAP  '`	N	Y	Y
	Open the MCF environment.	`'OPEN    '`	O	N	N
	Receive a synchronous message.	`'RECVSYNC'`	Y	Y	Y
	Resend a message.	`'RESEND  '`	N	Y	Y
	Send a message.	`'SEND    '`	N	Y	Y

Facility available with SPP	Names of facilities available with SUPs and request codes specified as data name at the beginning of COBOL-UAP creation program		SPP operating conditions		
			Outside	Inside	
				Root	Not root
	Exchange a synchronous message.	`'SENDRECV'`	Y	Y	Y
	Send a synchronous message.	`'SENDSYNC'`	Y	Y	Y
	Cancel user timer monitoring.	`'TIMERCAN'`	Y	Y	Y
	Set user timer monitoring.	`'TIMERSET'`	Y	Y	Y
Performance verification trace	Report the sequential number for an acquired performance verification trace.	`'PRFGETN '`	Y	Y	Y
	Acquire user-specific performance verification traces.	`'PRFPUT  '`	Y	Y	Y

Remote API facility	Establish a connection with a RAP- processing listener.	`'CONNECT '`
		`'CONNECTX'`
			Y	N	N

	Release a connection with a RAP-processing listener.	`'DISCNCT '`	Y	N	N
Remote procedure call	Request a remote service.	`'CALL    '`	Y	Y	Y
	Terminate an application program.	`'CLOSE   '`	O	N	N
	Report data to CUP unidirectionally.	`'CLTSEND '`	Y	Y	Y
	Reject the receiving of processing results.	`'DISCARDF'`	Y	Y	Y
	Reject acceptance of specific processing results.	`'DISCARDS'`	Y	Y	Y
	Acquire the node address of a client UAP.	`'GETCLADR'`	Y	Y	Y
	Acquire the descriptor of an asynchronous response-type RPC request which has encountered an error.	`'GETERDES'`	Y	Y	Y

Facility available with SPP	Names of facilities available with SUPs and request codes specified as data name at the beginning of COBOL-UAP creation program		SPP operating conditions		
			Outside	Inside	
				Root	Not root
	Acquire the node address of a gateway.	`'GETGWADR'`	Y	Y	Y
	Reference the schedule priority of a service request.	`'GETSVPRI'`	Y	Y	Y
	Reference the service response waiting interval.	`'GETWATCH'`	Y	Y	Y
	Start an application program.	`'OPEN    '`	O	N	N
	Receive processing results in asynchronous mode.	`'POLLANYR'`	Y	Y	Y
	Set a schedule priority of a service request.	`'SETSVPRI'`	Y	Y	Y
	Update the service response waiting interval.	`'SETWATCH'`	Y	Y	Y
	Retry a service program.	`'SVRETRY '`	Y	N	N
	Start an SPP service.	`'MAINLOOP'`	O	N	N
Real time statistical information service	Acquire real-time statistical information for arbitrary section.	`'RTSPUT  '`	Y	Y	Y
TAM file service	Delete a TAM table record.	`'ERS '/'ERSR'/'ZRS '/'ZRSR'`	N	Y	Y
	Input a TAM table record.	`'FxxR'/'FxxU'/ 'VxxR'/'VxxU'`	N	Y	Y
	Acquire TAM table status.	`'GST '`	Y	Y	Y
	Acquire TAM table information.	`'INFO    '`	Y	Y	Y
	Update/add a TAM table record.	`'MFY '/'MFYS'/'STR '/'WFY'/'WFYS'/'YTR '`	N	Y	Y

Facility available with SPP	Names of facilities available with SUPs and request codes specified as data name at the beginning of COBOL-UAP creation program		SPP operating conditions		
			Outside	Inside	
				Root	Not root
Transaction control	Start a transaction.	`'BEGIN   '`	Y	N	N
	Enable commitment in chained mode.	`'C-COMMIT'`	N	Y	N
	Enable rollback in chained mode.	`'C-ROLL  '`	N	Y	N
	Report the information about the current transaction.	`'INFO    '`	Y	Y	Y
	Enable commitment in unchained mode.	`'U-COMMIT'`	N	Y	N
	Enable rollback in unchained mode.	`'U-ROLL  '`	N	Y	Y
Online tester management	Report the test status of a user server.	`'T-STATUS'`	Y	Y	Y

Legend:

Outside: Outside the transaction processing range

Inside: Inside the transaction processing range

Y: Can be used with SPPs.

(Y): Can be used only when accessing an unrecoverable DAM file.

O: Can be used only from the main program.

N: Cannot be used with SPPs.

Note

Root in the table indicates a root transaction branch. *Not root* indicates a transaction branch other than the root transaction branch.

(3) Facilities and programs available with MHPs

Table 1-4 lists the facilities which can be used with MHPs and their request codes.

Table 1-4: Facilities available with MHPs and their request codes

Facility available with MHP	Names of facilities available with MHPs and request codes specified as data name at the beginning of COBOL-UAP creation program		MHP operating conditions	
			Outside	Inside
System operation management	Execute an operation command.	`'COMMAND '`	Y	Y
	Report the status of a user server.	`'STATUS  '`	Y	Y
DAM file service	Close a logical file.	`'DCDAMSVC','CLOS'`	Y	Y
	Terminate using an unrecoverable DAM file.	`'DCDAMSVC','END '`	Y	Y
	Shut down a logical file.	`'DCDAMSVC','HOLD'`	Y	Y
	Open a logical file.	`'DCDAMSVC','OPEN'`	Y	Y
	Input a logical file block.	`'DCDAMSVC','READ'`	Y	Y
	Update a logical file block.	`'DCDAMSVC','REWT'`	(Y)	Y
	Release a logical file from the shutdown state.	`'DCDAMSVC','RLES'`	Y	Y
	Reference the status of a logical file.	`'DCDAMSVC','STAT'`	Y	Y
	Start using an unrecoverable DAM file.	`'DCDAMSVC','STRT'`	Y	Y
	Output a logical file block.	`'DCDAMSVC','WRIT'`	(Y)	Y
IST service	Close an internode shared table.	`'DCISTSVC','CLOS'`	Y	Y
	Open an internode shared table.	`'DCISTSVC','OPEN'`	Y	Y
	Input an internode shared table record.	`'DCISTSVC','READ'`	Y	Y
	Output an internode shared table record.	`'DCISTSVC','WRIT'`	Y	Y
User journal acquisition	Acquire a user journal.	`'UJPUT   '`	Y	Y

Facility available with MHP	Names of facilities available with MHPs and request codes specified as data name at the beginning of COBOL-UAP creation program		MHP operating conditions	
			Outside	Inside
Look for resources	Enable locking of a resource.	`'GET     '`	N	Y
	Release all the resources from lock.	`'RELALL  '`	N	Y
	Release resource from lock specified by name.	`'RELNAME '`	N	Y
Audit log output	Output audit log data.	`'PRINT   '`	Y	Y
Output message log	Output message log.	`'PRINT   '`	Y	Y
Message exchange processing	Report the application information.	`'APINFO  '`	NO	Y
	Close the MCF environment.	`'CLOSE   '`	O	O
	Commit an MHP.	`'COMMIT  '`	N	Y
	Terminate continuous-inquiry-response processing.	`'CONTEND '`	NO	Y
	Activate an application program.	`'EXECAP  '`	NO	Y
	Start an MHP service.	`'MAINLOOP'`	O	N
	Open the MCF environment.	`'OPEN    '`	O	O
	Receive a message.	`'RECEIVE '`	NO	Y
	Receive a synchronous message.	`'RECVSYNC'`	Y	Y
	Send a response message.	`'REPLY   '`	NO	Y
	Resend a message.	`'RESEND  '`	N	Y
	Enable MHP rollback.	`'ROLLBACK'`	N	Y
	Send a message.	`'SEND    '`	NO	Y

Facility available with MHP	Names of facilities available with MHPs and request codes specified as data name at the beginning of COBOL-UAP creation program		MHP operating conditions	
			Outside	Inside
	Exchange a synchronous message.	`'SENDRECV'`	Y	Y
	Send a synchronous message.	`'SENDSYNC'`	Y	Y
	Accept temporary-stored data.	`'TEMPGET '`	NO	Y
	Update temporary-stored data.	`'TEMPPUT '`	NO	Y
	Cancel user timer monitoring.	`'TIMERCAN'`	Y	Y
	Set user timer monitoring.	`'TIMERSET'`	Y	Y
Performance verification trace	Report the sequential number for an acquired performance verification trace.	`'PRFGETN '`	Y	Y
	Acquire user-specific performance verification traces.	`'PRFPUT  '`	Y	Y
Remote API facility	Establish a connection with a RAP- processing listener.	`'CONNECT '` `'CONNECTX'`	Y	N
	Release a connection with a RAP-processing listener.	`'DISCNCT '`	Y	N
Remote procedure call	Request a remote	'CALL '	O	Y
	Terminate an application program	`'CLOSE   '`	O	N
	Report data to CUP unidirectionally.	`'CLTSEND '`	Y	Y
	Reject the receiving of processing results.	`'DISCARDF'`	Y	Y
	Reject acceptance of specific processing results.	`'DISCARDS'`	Y	Y
	Acquire the descriptor of an asynchronous-response type RPC request which has encountered an error.	`'GETERDES'`	Y	Y

Facility available with MHP	**Names of facilities available with MHPs and request codes specified as data name at the beginning of COBOL-UAP creation program**		**MHP operating conditions**	
			Outside	**Inside**
	Reference the schedule priority of a service request.	`'GETSVPRI'`	Y	Y
	Reference the service response waiting interval.	`'GETWATCH'`	Y	Y
	Start an application program.	`'OPEN    '`	O	N
	Receive processing results in asynchronous mode.	`'POLLANYR'`	O	Y
	Set a schedule priority of a service request.	`'SETSVPRI'`	Y	Y
	Update the service response waiting interval.	`'SETWATCH'`	Y	Y
Real time statistical information service	Acquire real-time statistical information for arbitrary section.	`'RTSPUT  '`	Y	Y
TAM file service	Delete a TAM table record.	`'ERS '/'ERSR'/'ZRS '/'ZRSR'`	N	Y
	Input a TAM table record.	`'FxxR'/'FxxU'/ 'VxxR'/'VxxU'`	N	Y
	Acquire TAM table status.	`'GST '`	Y	Y
	Acquire TAM table information.	`'INFO'`	Y	Y
	Update/add a TAM table record.	`'MFY '/'MFYS'/'STR '/'WFY'/'WFYS'/'YTR '`	N	Y
Transaction control	Start a transaction.	`'BEGIN   '`	O	N
	Enable commitment in unchained mode.	`'U-COMMIT'`	N	O
	Report the information about the current transaction	`'INFO    '`	Y	Y
	Enable rollback in unchained mode.	`'U-ROLL  '`	N	O

Facility available with MHP	Names of facilities available with MHPs and request codes specified as data name at the beginning of COBOL-UAP creation program		MHP operating conditions	
			Outside	Inside
Online tester management	Report the test status of a user server.	`'T-STATUS'`	Y	Y

Legend:

Outside: Outside the transaction processing range

Inside: Inside the transaction processing range

Y: Can be used with an MHP.

(Y): Can be used only when accessing an unrecoverable DAM file.

O: Can be used only from the main program.

NO: The function can be used only in the service-program range of nontransaction attribute MHPs.

N: Cannot be used with an MHP.

Note

Outside the transaction processing range means the range of nontransaction attribute MHPs or MHP main programs.

(4) Facilities and programs available with UAPs that handles offline work

Table 1-5 lists the facilities which can be used with UAPs that handle offline work and their request codes.

Table 1-5: Facilities available with UAPs that handle offline work and their request codes

Facility available with UAP That handles offline work	Names of facilities available with UAP that handles offline work and request codes specified as data name at the beginning of COBOL-UAP creation program	
DAM file service	Seek a physical file block.	`'DCDAMINT','BSEK'`
	Close a physical file.	`'DCDAMINT','CLOS'`
	Allocate a physical file.	`'DCDAMINT','CRAT'`
	Input directly a physical file block.	`'DCDAMINT','DGET'`
	Output directly a physical file block.	`'DCDAMINT','DPUT'`

Facility available with UAP That handles offline work	Names of facilities available with UAP that handles offline work and request codes specified as data name at the beginning of COBOL-UAP creation program	
	Input a physical file block.	`'DCDAMINT','GET '`
	Open a physical file.	`'DCDAMINT','OPEN'`
	Output a physical file block.	`'DCDAMINT','PUT '`
Journal data editing	Close the jnlrput output file.	`'CLOSERPT'`
	Open the jnlrput output file.	`'OPENRPT '`
	Input journal data of the jnlrput output file.	`'RDGETRPT'`
Performance verification trace	Report the sequential number for an acquired performance verification trace.	`'PRFGETN '`
	Acquire user-specific performance verification traces.	`'PRFPUT  '`

1.1.2 Coding rules

(1) Notes on coding

Write UAPs to be used with OpenTP1 by coding them in COBOL/2# or COBOL85. OpenTP1 facilities are made available by using COBOL-UAP creation programs residing in the OpenTP1 library.

In addition, any system calls and program libraries can also be used. However, it is recommendable to use OS-provided standard statements and system calls when writing UAPs in order to assure high portability of the UAPs.

When creating UAPs in the COBOL language which use system calls and arbitrary program libraries, note the following:

1. When issuing a signal from the UAP, do not register the type of a signal handler (`SIGILL` or `SIGBUS`) which creates a core file during operation with the signal default specified. If the signal handler is registered, a core file is not created even when the program terminates abnormally. As a result, troubleshooting is impossible.
2. When issuing a signal from the UAP, do not use COBOL-UAP creation programs in the OpenTP1 library from the signal handler.
3. Do not use the following system call:
   - `chdir` (change of the current working directory)

4. Do not use the following system calls after the UAP start statement:
    - `fork` (new process creation)
    - `exec` (file execution)
    - `system` (shell command issuance)
5. Do not use a jump statement (e.g., `GOTO` statement) which extends over service programs.
6. When using another program library, do not use Xlib and OSF/Motif programs which control event-driven dispatching.
7. When creating UAPs using the COBOL language, make sure that data areas such as unique-name-1 always begin at an even address. If a data area such as unique-name-1 begins at an odd address, a bus error will occur.

    Suppose that a unique-name-3 data area that is to be used in `CBLDCMCF` is defined in an array, the send data length in bytes is an odd number, and `SYNC` is not specified for the structure (in other words, no boundary alignment will be performed). In this situation, if a second data item in the array is set as an argument in `CBLDCMCF`, a bus error will occur when function processing is attempted.
8. The MCF event information area for MCF event processing MHPs (C system or V system) that was created in the COBOL language must have a spare area length at least 2 bytes longer than the spare area length of the MCF event information area for C.

If the OS is HP-UX, the bind mode for linkage must be specified as `immediate`. If an executable file created in another mode is used as an OpenTP1 UAP, the system operation is unpredictable. To check that the bind mode of the created UAP is `immediate`, use the `chatr` command of the OS.

#

COBOL/2 cannot be used depending on the OS.

(2) Notes on naming

We recommend that you include a certain prefix character string in names of any variables or definitions coded by the user. If any names duplicate those used by the OS or OpenTP1, system operation is unpredictable.

(a) Service program names, program names, and entry names

Service programs must be given names which are 20 or less alphanumeric characters in length and begin with an alphabetic character. Do not give service programs or entries the following names:

- Names beginning with `dc`

- Names beginning with `CBLDC`
- Names beginning with `tx` or `TX`
- Names beginning with `tp` or `TP`

(b) External variable names

Do not give external variables the following names except when such names are used according to the instructions in this manual:

- Names beginning with `dc`
- Names beginning with `CBLDC`
- Names beginning with `tx` or `TX`
- Names beginning with `tp` or `TP`

(3) Termination method

Programs (main programs) which are directly activated by OpenTP1 must terminate with `STOP RUN`. Service programs must terminate with `EXIT PROGRAM`. If a main program does not terminate with `STOP RUN`, COBOL85 count and other information will not be output for the program.

(4) When using Windows

When using OpenTP1 (TP1/LiNK) with Windows, compile and link-edit a UAP conforming to the specifications of the COBOL complier for Windows.

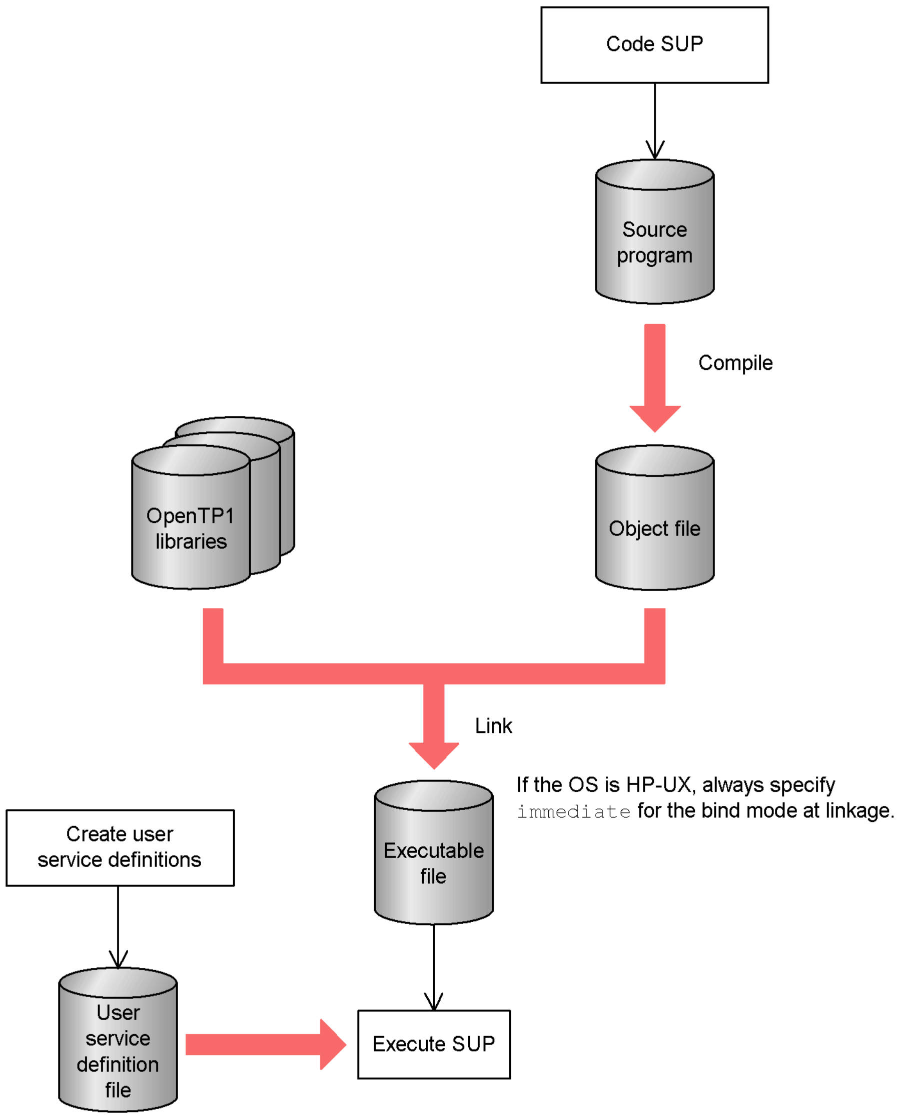
1.2 Creating application programs (TCP/IP)

1.2.1 Procedure for creating application programs

(1) Procedure for creating an SUP

The figure below shows the procedure for creating an SUP.

Figure 1-1: Procedure for creating SUPs


Code SUP
Source program
Compile
OpenTP1 libraries
Object file
Link
If the OS is HP-UX, always specify `immediate` for the bind mode at linkage.
Create user service definitions
Executable file
User service definition file
Execute SUP

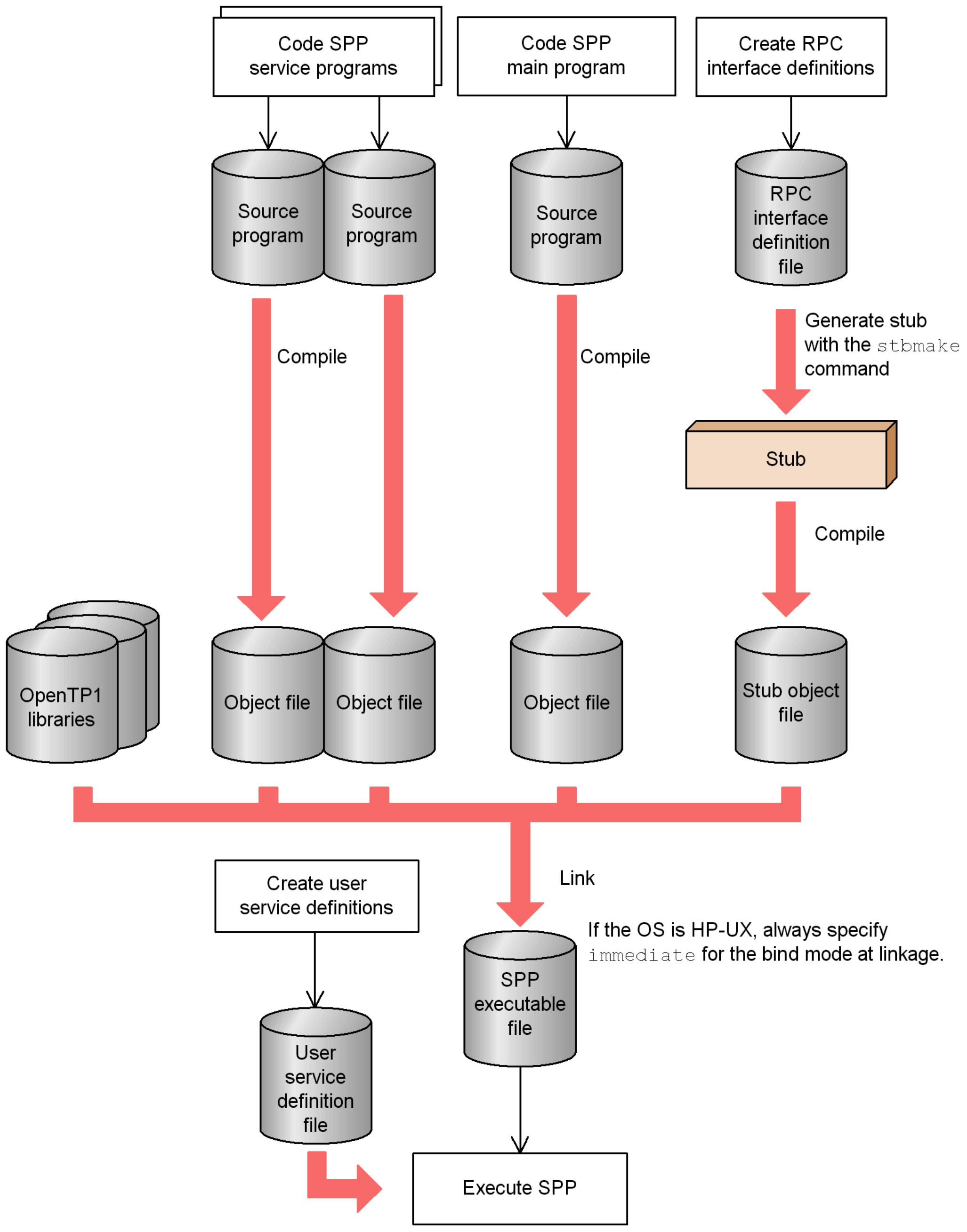
(2) Procedure for creating an SPP

The SPP creation procedure depends on whether the SPP uses a stub or uses dynamic loading of service functions.

(a) Creating an SPP by using a stub

The figure below shows the procedure for creating an SPP by using a stub.

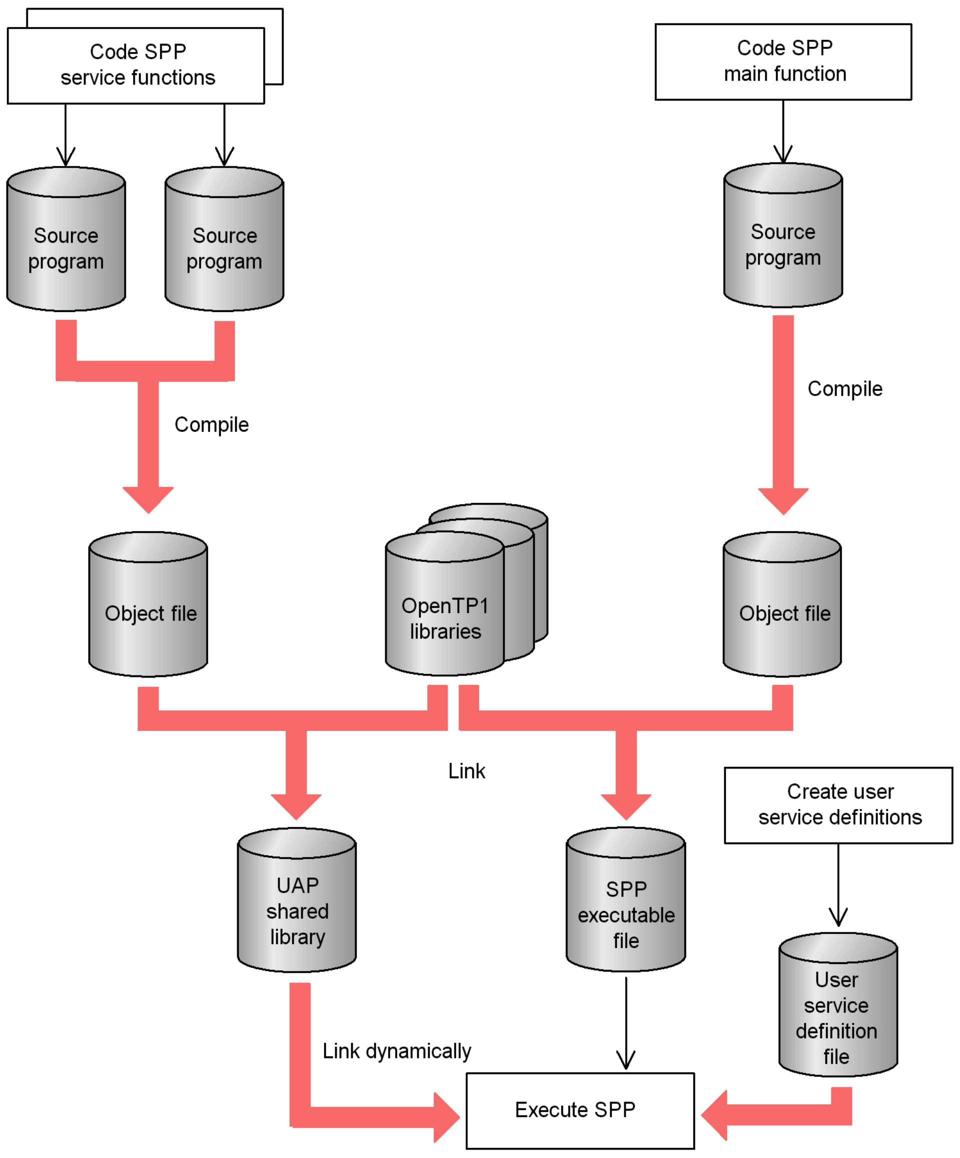
Figure 1-2: Procedure for creating an SPP by using a stub


Code SPP service programs
Code SPP main program
Create RPC interface definitions
Source program
Source program
Source program
RPC interface definition file
Compile
Compile
Generate stub with the stbmake command
Stub
Compile
OpenTP1 libraries
Object file
Object file
Object file
Stub object file
Link
If the OS is HP-UX, always specify immediate for the bind mode at linkage.
Create user service definitions
SPP executable file
User service definition file
Execute SPP

(b) Creating an SPP that dynamically loads service functions

The figure below shows the procedure for creating an SPP that dynamically loads service functions.

Figure 1-3: Procedure for creating an SPP that dynamically loads service functions



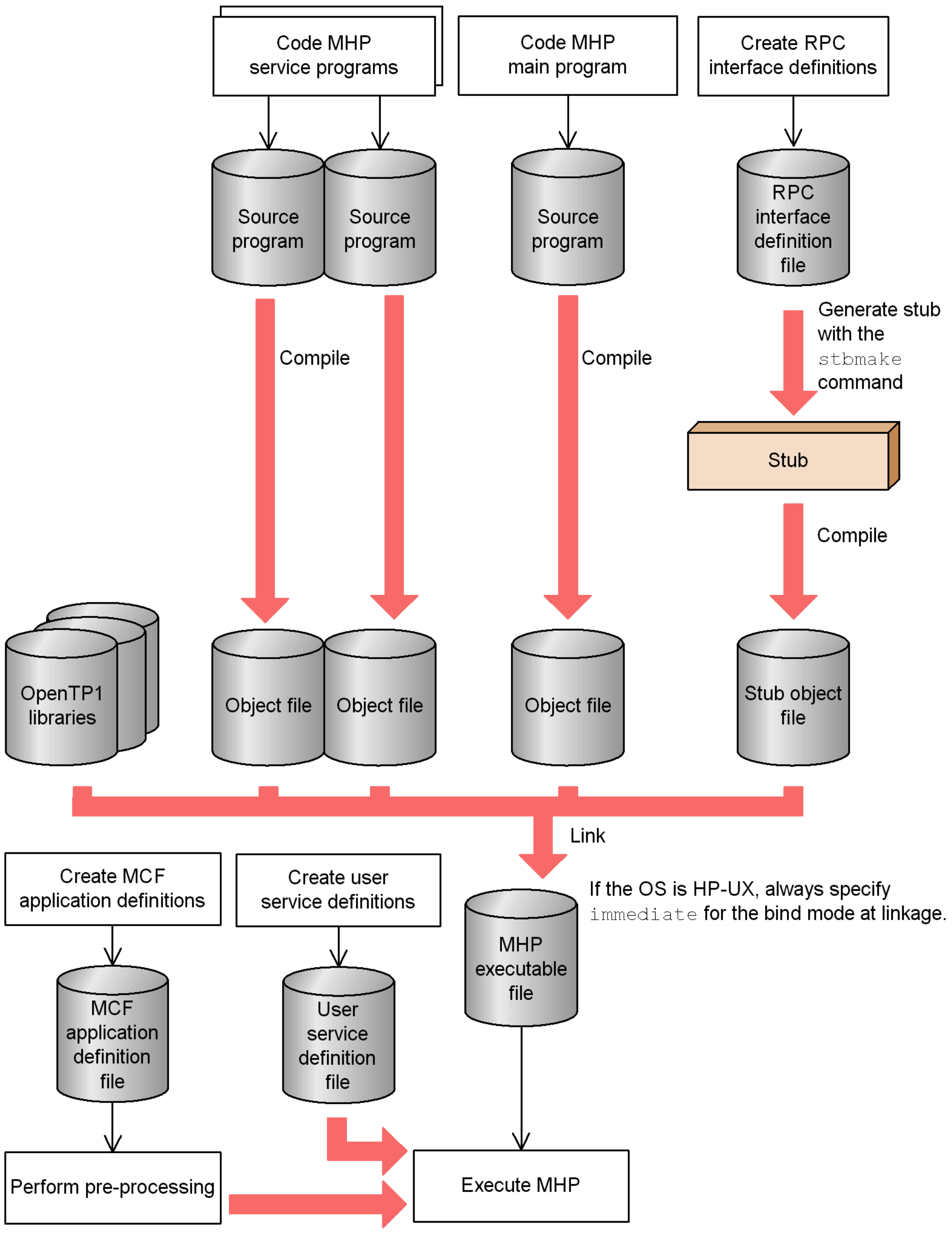
(3) Procedure for creating an MHP

The MHP creation procedure depends on whether the MHP uses a stub or uses dynamic loading of service functions.

(a) Procedure for creating an MHP (when using a stub)

The figure below shows the procedure for creating an MHP that uses a stub.

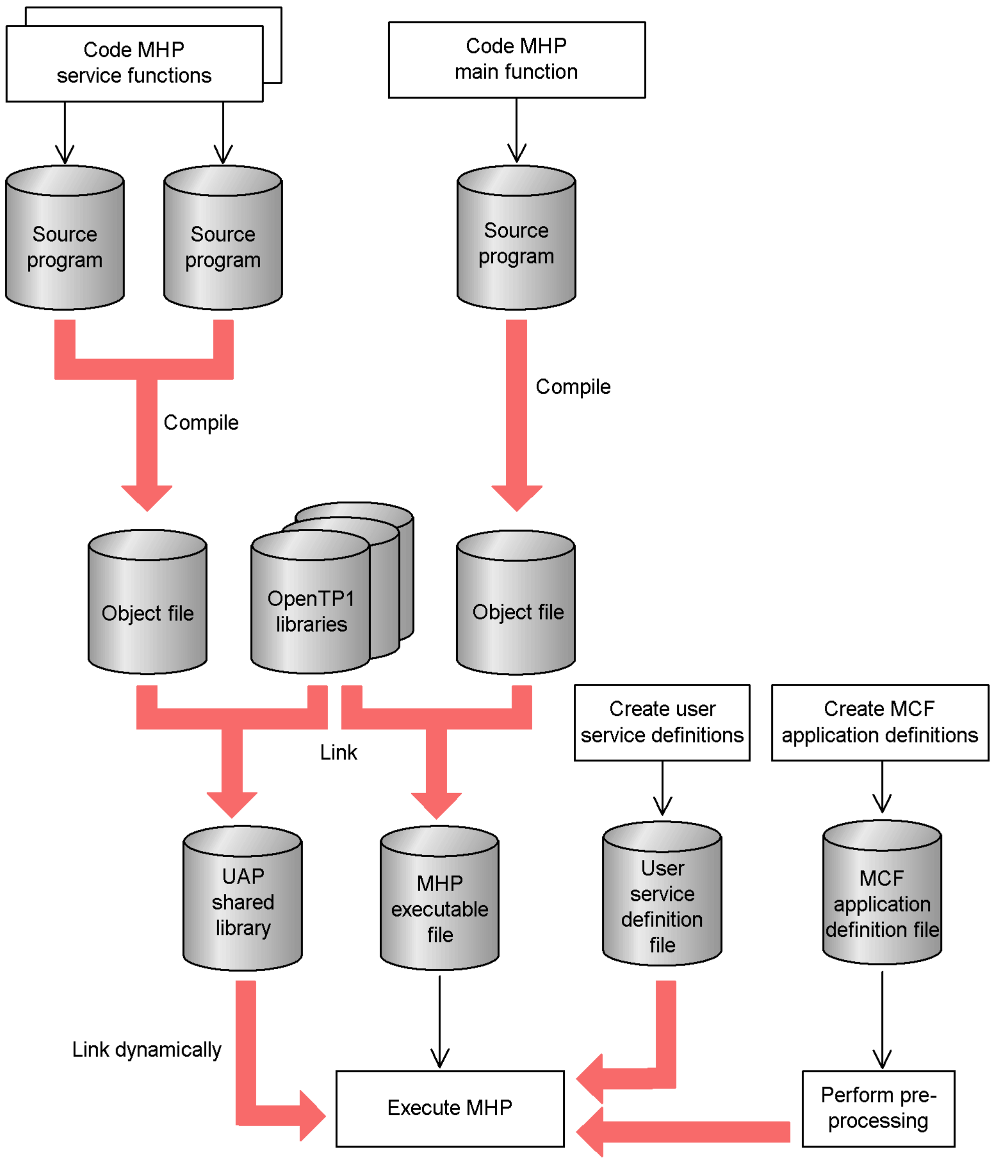
Figure 1-4: Procedure for creating an MHP (when using a stub)


Code MHP service programs
Code MHP main program
Create RPC interface definitions
Source program
Source program
Source program
RPC interface definition file
Compile
Compile
Generate stub with the stbmake command
Stub
Compile
OpenTP1 libraries
Object file
Object file
Object file
Stub object file
Link
Create MCF application definitions
Create user service definitions
If the OS is HP-UX, always specify immediate for the bind mode at linkage.
MHP executable file
MCF application definition file
User service definition file
Perform pre-processing
Execute MHP

(b) Procedure for creating an MHP (when using dynamic loading of service functions)

The figure below shows the procedure for creating an MHP that uses dynamic loading of service functions.

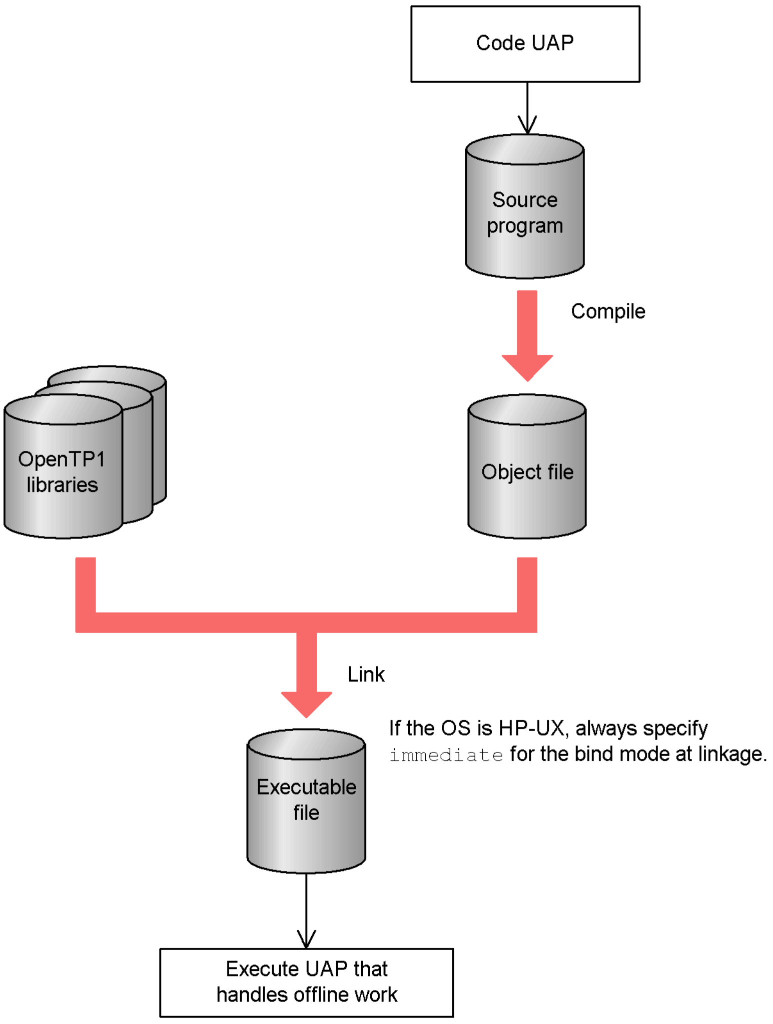
Figure 1-5: Procedure for creating an MHP (when using dynamic loading of service functions)



(4) Procedure for creating a UAP that handles offline work

The figure below shows the procedure for creating a UAP that handles offline work.

Figure 1-6: Procedure for creating UAPs that handle offline work


Code UAP
Source program
Compile
OpenTP1 libraries
Object file
Link
If the OS is HP-UX, always specify immediate for the bind mode at linkage.
Executable file
Execute UAP that handles offline work

1.2.2 Creating stubs

UAPs used with the OpenTP1 require libraries for fulfilling inter-UAP service requests. One of these libraries is called a *stub*.

(1) UAPs requiring stubs

Of all UAPs used with OpenTP1, UAPs having service programs (SPP, MHP) require a stub. However, a stub is not required when all service functions are put in the UAP shared library from where they are dynamically loaded. The UAP shared library is a shared library created by linking the UAP object files compiled from UAP source files.

Note that neither UAPs that handle offline work nor SUPs require a stub because they do not have a service function.

(2) Stub creation procedure

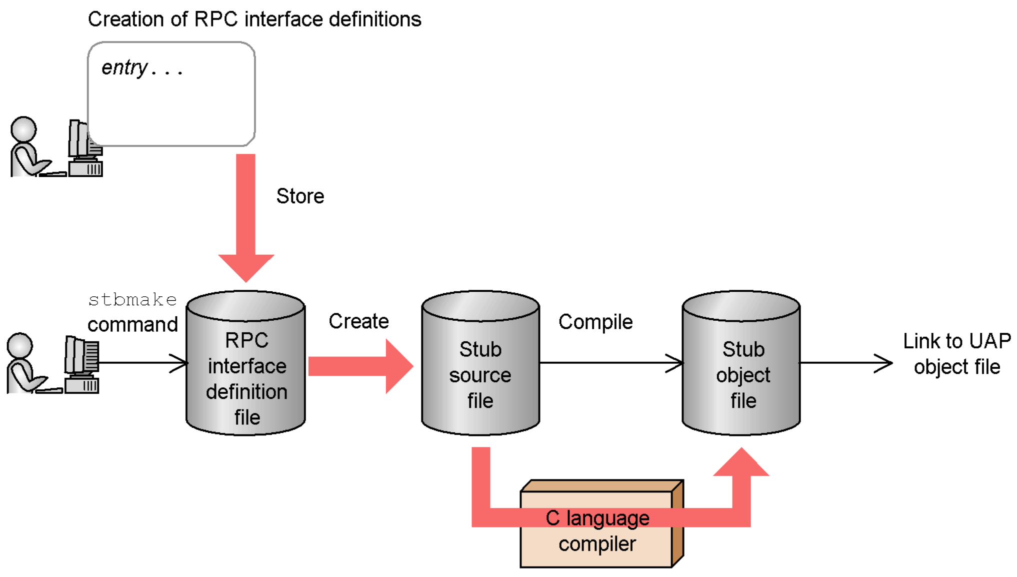
Before creating a stub, create a file (RPC interface definition file) in which UAP service programs are defined. Execute the `stbmake` command with this file as the argument.

When the `stbmake` command is executed, a source file (C-language source file) for the stub is created. Compile this file with the C-language compiler and link it to the object file of the UAP.

When modifying the stub, create the UAP from scratch. Modify the RPC interface definition file, recreate the stub, and link it to the object file of the recompiled UAP.

The figure below shows the stub creation procedure.

Figure 1-7: Stub creation procedure



(3) Creation of RPC interface definition file

When creating a stub, create a file which defines program IDs to the SPP and MHP services. What is defined here is called the *RPC interface definition*. The file containing this definition is called the *RPC interface definition file*.

Create an RPC interface definition file for each executable file of the SPP or MHP.

(a) Format of RPC interface definition

Write the RPC interface definition in the following format:

Format

```
entry "program-ID"["program-ID"...];
```

Description

This statement specifies the names of the program-IDs to the SPP and MHP service programs. Each program ID name must be a COBOL program name. Use 20 characters or fewer to specify each program ID.

The program IDs must correspond to the service names as specified in the user service definition.

Comments can be added to the RPC interface definition. Begin each comment with a forward slash asterisk combination (`/*`) and terminate it with an asterisk forward-slash combintaion (`*/`). Comments cannot be nested. Comments cannot be written within a keyword, identifier, or other character string.

More than one `entry` statement can be written in one file. An example of RPC interface definition is given below.

Example

Specification of RPC interface definition for a UAP which has service programs with their program IDs identified by `sv01` and `sv02` (use either format below)

```
entry "sv01";
entry "sv02";
```

```
entry "sv01" "sv02";
```

(4) Name of RPC interface definition file

The name of an RPC interface definition file must be appended with a suffix `.def` which indicates that the file is an RPC interface definition file. The RPC interface definition file may be placed under any directory that the `stbmake` command can search.

The name of an RPC interface definition file is up to 255 characters long. Note, however, that a lower upper limit is used under some OSs.

After the `stbmake` command is executed, stub source files are created under names different from those in the RPC interface definition file. Therefore, the RPC interface definition file will not be used when the OpenTP1 is active.

1.2.3 Creating stub source file

To create the source file of the stub, execute the `stbmake` command with the RPC interface definition file name as the argument.

(1) File created by stbmake command

When the `stbmake` command is executed, the following file is created (*xxxxx* is the RPC interface definition file name minus the suffix`.def`).

- Stub source file (file name: *xxxxx*`_sstb.c`)

The name of the source file can be changed using an option to the command.

The name of a source file is up to 255 characters long. Note, however, that a lower upper limit is used under some OSs.

Compile the stub source file with the C-language compiler and link it with the UAP object file.

1.2.4 stbmake - Stub source file creation

(1) Format

```
stbmake [-s [stub-source-file-name]] definition-file-name
```

(2) Description

Creates a stub source file from the RPC interface definition file.

To create a UAP that will use both OpenTP1 remote procedure calls and the XATMI interface, see the explanation about the `stbmake` command in *A. Using OpenTP1 Remote Procedure Calls and XATMI-interfaced API Functions in Combination*.

(3) Options

`-s` *stub-source-file-name* ~ <pathname>

Specify the pathname of the stub source file to be created. If no pathname is specified here, the source file name is the same as the RPC interface definition file name except that the suffix`.def` is replaced with `_sstb.c` and the source file is created in the current directory.

If a source file with the specified file name is already present, it is replaced with the created source file and is lost.

(4) Flags

definition-file-name ~ <pathname>

Specify the pathname of the RPC interface definition file.

(5) Note

The names of files that the `stbmake` command can take as input or create as output are up to 255 characters long. Note, however, that a lower upper limit is used under some OSs.

(6) Example

An example of using the `stbmake` command is given below.

Example:

Creating a stub source file from an RPC interface definition file `test.def` in the current directory.

Format 1:

```
stbmake test.def
```

A stub source file `test_sstb.c` is created from an RPC interface definition file `test.def` in the current directory.

Format 2:

```
stbmake -s stub/test.c test.def
```

A directory `stub` is created under the current directory and a stub source file `test.c` is created in the created directory.

1.2.5 Compiling and linking application program

For details on how to compile and link UAPs, see the reference manual of the OS to be used.

- Note on UAP creation

  Be careful of the OpenTP1 version in creating a UAP. Some system services do not accept statements from UAPs in old versions. To use a UAP created in an old version, the UAP should be recompiled in the OpenTP1 version.

(1) UAP compilation

To create the object file of a UAP written in COBOL language, compile the source program with the COBOL compiler.

See the *COBOL Language* manual for details on how to compile UAPs.

(2) Stub compilation

To create the object file of a stub, compile the stub source file with the C compiler.

(3) Linkage

The following notes (#1 to #3) apply to files treated in (a) to (d) below.

#1:

The object file for transaction control is required to execute transactions that access the resource manager via the XA interface. Note that any resource manager provided by OpenTP1 is accessed by the XA interface. An object file for transaction control is created by using an OpenTP1 command (`trnmkobj` command). For details on the `trnmkobj` command, see the manual *OpenTP1 Operation*.

#2:

The object file provided by resource manager is required to access the resource manager. The following arguments can be specified in the linkage command to link object files provided by OpenTP1:

Arguments for using the message exchange facility: `-lmcf` and `-lmnet`

Argument for using the DAM access facility: `-ldam`

Argument for using the TAM access facility: `-ltam`

Arguments for using the ISAM facility: `-lismb`, `-lisam`, and `-lrsort`

Argument specified for using message queuing: `-lmqacb`

For details on how to link object files for a non-Hitachi resource manager, see the documentation for the resource manager.

#3:

The object file provided by online tester is required to use the `CBLDCUTO('T-STATUS')` function, which reports the user server test status. The following argument is specified to link the object file for the online tester:

Argument for reporting the user server test status: `-luto`

(a) Files to be linked to SPP and MHP

The executable file of an SPP or MHP is linked to the following files when it is created:

- UAP object file (main and service programs)
- Stub object file
- Object file for transaction control#1
- Object file provided by resource manager#2
- Object file provided by online tester#3

- OpenTP1 library
- COBOL library (COBOL85 library if the UAP was created in the COBOL85 language)

(b) Files to be linked to SUP

The executable file of a UAP that handles offline work is linked to the following files when it is created:

- UAP object file (main program)
- Object file for transaction control#1
- Object file provided by resource manager#2
- Object file provided by online tester#3
- OpenTP1 library
- COBOL library (COBOL85 library if the UAP was created in the COBOL85 language)

(c) Files to be linked to UAP that handles offline work

The executable file of a UAP that handles offline work is linked to the following files when it is created:

- UAP object file (main program)
- OpenTP1 library
- COBOL library (COBOL85 library if the UAP was created in the COBOL85 language)

(d) Files to be linked to an SPP or MHP that performs dynamic loading of service functions

The executable file of an SPP that dynamically loads service functions is linked to the following files when the file is created:

- UAP object file (main function)
- OpenTP1 library
- Object file for transaction control#1
- Object file provided by resource manager#2
- Object file provided by online tester#3

In addition to the above files, the following files are required when the SPP also uses a service search that employs a stub:

- UAP object file (service function)
- Stub object file

(4) Note

If the OS is HP-UX, the bind mode for linkage must be specified as `immediate`. If an executable file created in another mode is used as an OpenTP1 UAP, the system operation is unpredictable. To check that the bind mode of the created UAP is `immediate`, use the chatr command of the OS.

1.3 Creating XATMI interface application programs (TCP/IP, OSI TP)

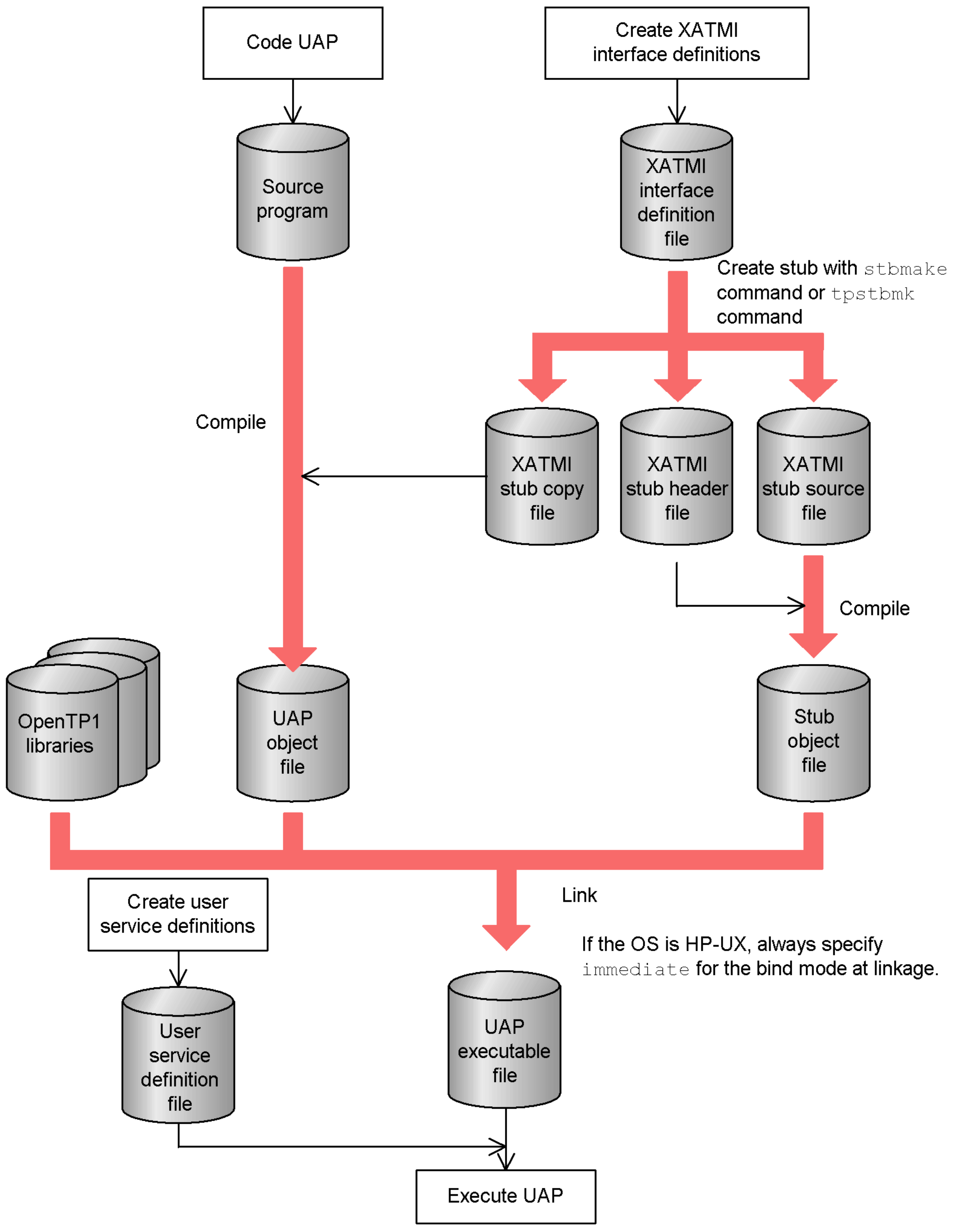
This section explains how to create a UAP that uses the XATMI interface when the communication protocol is TCP/IP or OSI TP.

A UAP that uses the XATMI interface can be created in the same manner as for OpenTP1 UAPs except for the following two points: (1) stub creation method (execution formats of `stbmake` and `tpstbmk` commands) and (2) files to be linked with the UAP. For the UAP creation procedure, see *1.1 Coding application program* and *1.4 Executing application programs*.

1.3.1 Procedure for creating XATMI-Interfaced application programs

The figure below shows the procedure for creating UAP.

Figure 1-8: Procedure for creating UAP (XATMI interface TCP/IP, OSI TP)


Code UAP
Create XATMI interface definitions
Source program
XATMI interface definition file
Create stub with stbmake command or tpstbmk command
Compile
XATMI stub copy file
XATMI stub header file
XATMI stub source file
Compile
OpenTP1 libraries
UAP object file
Stub object file
Link
Create user service definitions
If the OS is HP-UX, always specify immediate for the bind mode at linkage.
User service definition file
UAP executable file
Execute UAP

1.3.2 Creating stubs for XATMI interface

This subsection explains how to create the stub for the XATMI interface. For UAP communication through the XATMI interface, stubs are necessary on both the client and server UAPs.

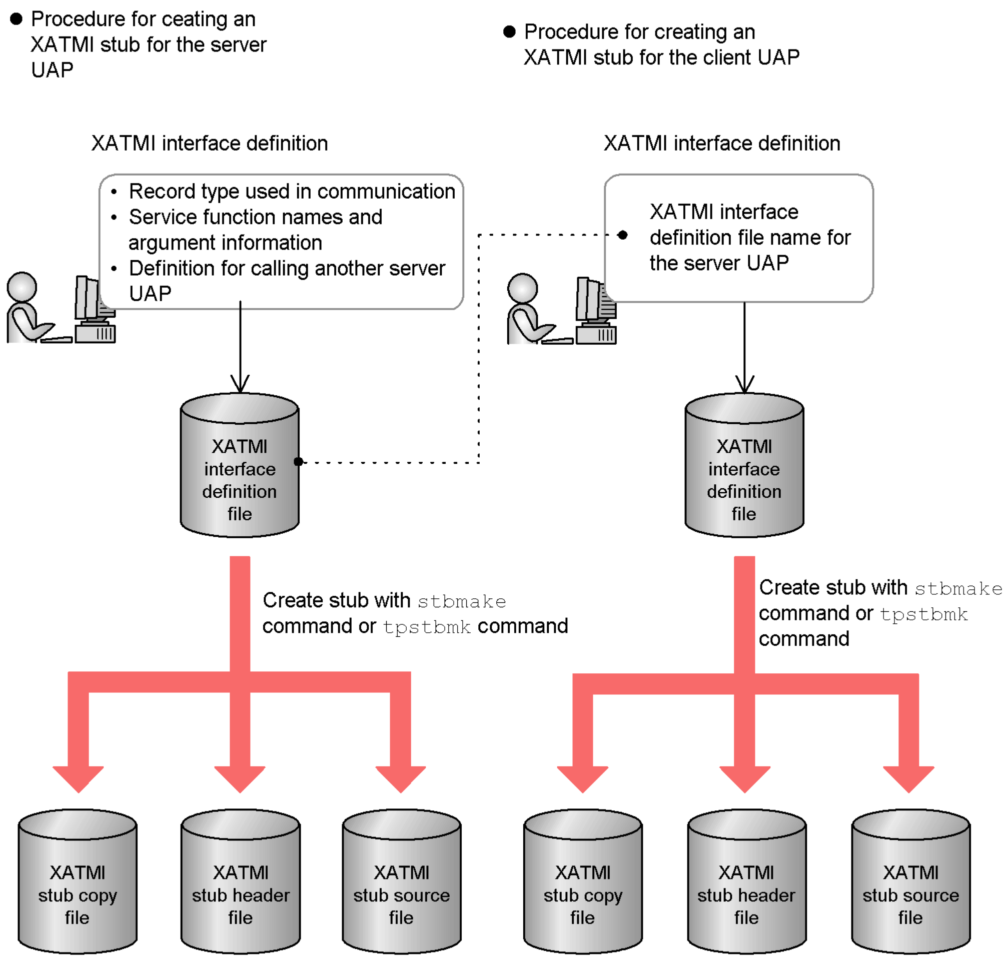
Before creating a stub, create a file (XATMI interface definition file) in which information about the XATMI interface is defined, then execute one of the following stub creation commands:

- For a UAP that supports TCP/IP communication: `stbmake` command
- For a UAP that supports OSI TP communication: `tpstbmk` command

Compile the created stub source file with the C-language compiler and link it to the UAP object file.

The figure below outlines the procedure for creating the stub for the XATMI interface.

Figure 1-9: Procedure for creating stub for XATMI interface


● Procedure for ceating an XATMI stub for the server UAP
● Procedure for creating an XATMI stub for the client UAP
XATMI interface definition
• Record type used in communication
• Service function names and argument information
• Definition for calling another server UAP
XATMI interface definition
XATMI interface definition file name for the server UAP
XATMI interface definition file
XATMI interface definition file
Create stub with stbmake command or tpstbmk command
Create stub with stbmake command or tpstbmk command
XATMI stub copy file
XATMI stub header file
XATMI stub source file
XATMI stub copy file
XATMI stub header file
XATMI stub source file

(1) XATMI interface definition (for client UAP)

The XATMI interface definition for the client UAP (SUP or SPP) is in the format explained below.

Format

```
called_servers={"server-definition-file-name"
                [,"server-definition-file-name"]...};
```

Description

Specify all XATMI interface definition file names defined in the server UAP. When a server UAP definition file is specified, the typed record defined in the server definition file can be used by the client UAP process.

Parameters

- *server-definition-file-name*

  Specify the file name of the XATMI interface definition file of the server UAP. The definition file name must have a suffix `.def`.

  Multiple definition files names can be specified in braces `{}` in one `called_servers` statement. It is also possible to write multiple `called_servers` statements in one XATMI interface definition file.

Example

Defining a client UAP which communicates with server UAP1 and server UAP2 through the XATMI interface (assuming that the server UAP1 definition file name is `serv1.def` and the server UAP2 definition file name is `serv2.def`).

Format 1:

```
called_servers = {"serv1.def" ,"serv2.def"};
```

Format 2:

```
called_servers = {"serv1.def" };
called_servers = {"serv2.def" };
```

(2) XATMI interface definition (for server UAP)

For the XATMI interface definition of a server UAP, the following items must be specified in any order:

- Definition of the typed record to be used
- Definition of service function name and argument information
- `called_servers` statement (if the server UAP is to call another server UAP)

(a) Definition of the typed record to be used

Format

```
type-name  subtype-name{
              data-type  data-name;
             [data-type  data-name;]
                     :
                     :
             } ;
```

Description

Define the type, subtype, and structure of the typed record to be used with the server UAP. If the server UAP is to call service from another server UAP process, the typed record which can be used by the calling process can also be used by any local process. Therefore, define here only the typed record to be used for I/O by the service function within the local process. However, `X_OCTET` will always be recognized. If `X_OCTET` is defined, the execution of the stub creation command (`stbmake` or `tpstbmk`) will encounter an error.

`X_C_TYPE` cannot be used for COBOL APIs. If `X_C_TYPE` is defined, the execution of the stub creation command (`stbmake` or `tpstbmk`) with the `-b` option specified results in an error.

Parameters

- *type-name*

  Specify the type name of the typed record to be used with the server UAP.

- *subtype-name*

  Specify the subtype name of the typed record to be used with the server UAP.

- *data-type*

  Specify the data type of the data contained in the structure of the typed record to be used with the server UAP.

- *data-name*

  Specify the data name of the data contained in the structure of the typed record to be used with the server UAP.

Data types that can be used for type arguments

Table 1-6 lists the data types that can be used as *types*. The *Identifier* column indicates the data type to be specified in the XATMI interface definition. The *COBOL data* column indicates the typed record to be actually defined for the stub. When converting the data type for communication with a non-OpenTP1 system, include the identifier to be converted in the XATMI interface definition.

Table 1-6: Data types that can be used for type arguments

Type	**Identifier**	**COBOL data**	**Communication protocol**		**Remarks**
			TCP/IP	**OSI TP**	
`X_OCTET`	--[#1]	--[#1]	Y	Y	None

Type	Identifier	COBOL data	Communication protocol		Remarks
			TCP/IP	OSI TP	
X_COMMON	short a	PIC S9(9) COMP-5	Y	Y	None
	short a[*n*]	PIC S9(9) COMP-5 OCCURS *n* TIMES	Y	Y	None
	long a	PIC S9(9) COMP-5	Y	Y	None
	long a[*n*]	PIC S9(9) COMP-5 OCCURS *n* TIMES	Y	Y	None
	char a#2	PIC X	Y	Y	Array not to be converted
	octet a	PI C X	Y	Y	Array not to be converted
	tchar a	PIC X	O	Y	Array to be converted
X_COMMON	char a[*n*]#2	PIC X(*n*)	Y	Y	Array not to be converted
	octet a[*n*]	PIC X(*n*)	Y	Y	Array not to be converted
	tchar a[*n*]	PIC X(*n*)	O	Y	Array to be converted
X_C_TYPE	--#3	--#3	N	N	None

Legend:

Y: Available with the communication protocol

N: Unavailable with the communication protocol

--: Always treated as an identifier not to be converted

#1: `X_OCTET` will always be recognized, regardless of whether it is defined. If `X_OCTET` is specified in the XATMI interface definition, the execution of a stub creation command will encounter an error.

#2: This identifier is available. However, the following identifier should be used for new stub creation:

`octet` or `tchar` for `X_COMMON`

`str` or `tstr` for `X_C_TYPE`

#3: X_C_TYPE cannot be used for COBOL APIs. If X_C_TYPE is defined, the execution of a stub creation command (stbmake or tpstbmk) with the -b option specified results in an error.

Example

```
X_COMMON subtype1 {
                  char name[8];
                  long data[10];
                  long flags;
                  };
```

(b) Definition of service function name and argument information

Format

```
service service-program-name {
               (type-name [subtype-name])(ALL)( [void] )};
```

Description

Specify the program name of the service program in the server UAP and the type name and subtype name of the typed record to be passed as the arguments. The argument is the data member of the svc_info structure which is the actual argument to the service program.

For the X_OCTET type, specify only the type name because there is no subtype. If intended processing does not involve reference to the data member of the svc_info structure in the service program, assign nothing or void to the argument.

The TPCALL, TPACALL, and TPCONNECT can call a service program without sending the typed record. If data indicated by a member of the svcinfo structure with a service program is not to be referenced explicitly, assign nothing or void to the argument.

To call a specified service program, set NULL for the pointer to the typed record sent with the TPCALL, TPACALL, or TPCONNECT functions at the client side. For the X_OCTET type, a specified service program can be called even if NULL is not set for the pointer or the length of the sent data is zero.

If specification is not to limit the typed record to be received as an argument, assign ALL to the argument. The service program defined with argument ALL can receive any type of typed records as long as they are recognizable in the local process.

Parameters

- *service-program-name*

  Specify the program name in the server UAP.

- *type-name*

  Specify the type name given to the argument to the function.
- *subtype-name*

  Specify the subtype name given to the argument to the function.

Examples

Example 1:

```
service svc_func1(X_COMMON subtype1);
```

Example 2 (argument type is X_OCTET):

```
service svc_func2(X_OCTET);
```

Example 3 (service program without argument reception):

```
service svc_func3(void); or service svc_func3();
```

Example 4 (service program without argument limitation):

```
service svc_func4(ALL);
```

(c) If the server UAP is to call another server UAP:

Specify the XATMI interface definition (`called_servers` statement) of the client UAP.

(3) Name of XATMI interface definition file

The name of an XATMI interface definition file must be appended with a suffix `.def` which indicates that the file is an XATMI interface definition file. The XATMI interface definition file may be placed under any directory that the stub creation command (`stbmake` or `tpstbmk`) can search.

The name of an XATMI interface definition file is up to 255 characters long. Note, however, that a lower upper limit is used under some OSs.

After the stub creation command (`stbmake` or `tpstbmk`) is executed, stub source files are created under names different from those in the XATMI interface definition file. Therefore, the XATMI interface definition file will not be used when the OpenTP1 is active.

(4) Including the definition file

If the same typed record is to be used by different processes, the user can create a definition file for the shared typed record and include it in the definition file for each process.

The statement for including the definition file is in the same format as in the C language as follows:

```
#include <file-name> or #include "file-name"
```

The include file will be read through the search path specified by the `-i` option to the stub creation command (`stbmake` or `tpstbmk` command). If the appropriate file is not found in the search path, the current directory will finally be searched.

The file to be included may be given any name (the suffix need not be `.h`). However, if the file is directly specified in the stub creation command (`stbmake` or `tpstbmk`) as the XATMI interface definition file, observe the definition naming convention.

The contents of the file to be included are the same as those of the XATMI interface definition file. However, the file should not contain the definition of a service function within the local process in order to avoid name duplication.

(5) Naming conventions

1. Service programs and subtypes must be named according to the OpenTP1 rules as follows:
   - Any name cannot begin with `dc`, `DC`, `CBLDC`, `tx`, `TX`, `tp`, or `TP`.
   - Service program names must be 20 characters or less long.
   - The maximum subtype name length is 32 characters. Of these characters, the first 16 characters are valid. These 16 characters are checked for duplication.
   - Up to 32 characters can be used for the data names of data used in the structures of typed records.
2. Service program names must be unique within the same process.
3. Subtype names may be duplicate in the same process only if the types and structures are identical. Otherwise, the stub creation command (`stbmake` or `tpstbmk`) returns with an error.
4. Identical service program names or subtype names may be used in different processes. However, processes treated as different servers will be regarded as the same process by the client if they are called from one client.

1.3.3 Creating stub source files for XATMI interface

Create a stub for the XATMI from the created XATMI interface definition file.

Before creating a stub, create a file (XATMI interface definition file) in which information about the XATMI interface is defined, and then execute one of the following stub creation commands:

- `stbmake` command (when the UAP is for TCP/IP communication)

- `tpstbmk` command (when the UAP is for OSI TP communication)

Create stubs for the client and server UAPs in the following way:

(1) Files created by the stbmake command or tpstbmk command

The following three files are created by executing the `stbmake` command (*xxxxx* is the XATMI interface definition file name minus the suffix `.def`):

- XATMI stub source file (default file name: *xxxxx*`_stbx.c`)
- XATMI stub header file (default file name: *xxxxx*`_stbx.h`)
- XATMI stub copy file (the file name consists of the subtype name followed by `.cbl`)

(a) XATMI stub source file

The XATMI stub source file will be compiled with the C-language compiler and linked to the UAP object file.

(b) XATMI stub header file

The XATMI stub header file will be included in the UAP source file and XATMI stub source file.

(c) XATMI stub copy file

The created XATMI stub copy file is invoked by the `COPY` statement from UAPs written in COBOL. It makes typed records available.

The name of an XATMI stub copy file is up to 255 characters long. Note, however, that a lower upper limit is used under some OSs.

The file name and the directory containing the file can be changed using appropriate command options.

1.3.4 stbmake - Stub source file creation for XATMI interface

(1) Format

```
stbmake [-x] [-b] [-S stub-source-file-name]
        [-H stub-header-file-name]
        [-i include-file-pathname]
        [-m server-definition-file-pathname]
        [-p] definition-file-name
```

(2) Description

Creates XATMI stub source files needed when XATMI-interfaced communication is to be used under TCP/IP. The `stbmake` command creates the following files by referring to the XATMI interface definition file:

- XATMI stub source file

- XATMI stub header file (used with UAPs written in C)
- XATMI stub copy file (used with UAPs written in COBOL)

When creating a UAP that uses OpenTP1 remote procedure calls and XATMI interface functions in combination, see the descriptions of the `stbmake` command in *A. Using OpenTP1 Remote Procedure Calls and XATMI-interfaced API Functions in Combination*.

(3) Options

■ `-x`

Indicates that the stub created will serve the UAP which uses the XATMI interface. The `-x` option can be omitted.

■ `-b`

Specify this option when creating an XATMI stub copy file which is to be used with COBOL UAPs. If this option is omitted, any XATMI stub copy file will not be created.

The name of an XATMI stub copy file is output in the format of *subtype-name*`.cbl`. The XATMI interface for the COBOL language does not support `X_C_TYPE` as a record type. If `X_C_TYPE` is specified in the XATMI interface definition, the stbmake command with the `-b` option specified will return with an error.

■ `-S` *stub-source-file-name* ~ <pathname>

Specify this option if the XATMI stub source file created is to be renamed. The relative or absolute pathname may be used for this file name.

If this option is omitted, the file will be created with name *xxxxx*`_stbx.c` in the current directory.

■ `-H` *stub-header-file-name* ~ <pathname>

Specify this option if the XATMI stub header file created is to be renamed. The relative or absolute pathname may be used for this file name.

If this option is omitted, the file will be created with name *xxxxx*`_stbx.h` in the current directory.

■ `-i` *include-file-pathname* ~ <pathname>

Specify the search path containing the include file specified by the `#include` statement to be used. The `stbmake` command searches the directory identified by the `-i` option for the include file.

If the `-i` option is omitted, the current directory is searched for the include file.

The `-i` option can be specified only once. If more than one search path is needed,

`-i` must be followed by the desired paths separated by colons (:). The search order is the order in which the paths are written as the argument to the `-i` option.

Use alphanumeric characters, underscore (_), slash (/), and period (.) when specifying a search pathname.

- `-m` *server-definition-file-pathname* ~ <pathname>

  Specify the search path containing the server definition file to be used. The stbmake command searches the directory identified by the `-m` option for the server definition file specified by the called_servers statement.

  If the `-m` option is omitted, the current directory is searched for the definition file.

  The `-m` option can be specified only once. If more than one search path is needed, `-m` must be followed by the desired paths separated by colons (:). The search order is the order in which the paths are written as the argument to the `-m` option.

  Use alphanumeric characters, underscore (_), slash (/), and period (.) when specifying a search pathname.

- `-p`

  Specify this option to output the allocation status of the typed record in memory to the standard output. In the case of testing by the online tester, use the `-p` option to learn about how XATMI structure members are allocated in memory.

  When the `-p` option is specified, the `stbmake` command creates no files. Thus, output file names specified in the `-S` and `-H` option are ignored. Specify the `-m` and `-i` options to search for files as needed.

(4) Command argument

- `definition-file-name`

  Specify the XATMI interface definition file name. Its suffix must be`.def`.

(5) Notes

- Each option to the `stbmake` command for XATMI stub creation can be specified only once. If an option is specified more than once, the last specified value will be valid.
- The names of files that the `stbmake` command can take as input or create as output are up to 255 characters long. Note, however, that a lower upper limit is used under some OSs.

1.3.5 tpstbmk - XATMI-interfaced stub creation for OSI TP communication

(1) Format

```
tpstbmk [-b] [-S stub-source-file-name]
             [-H stub-header-file-name]
             [-i include-file-pathname]
             [-m server-definition-file-pathname]
             definition-file-name
```

(2) Description

Creates XATMI stub source files needed when XATMI-interfaced communication is to be used under OSI TP. The `tpstbmk` command creates the following files by referring to the XATMI interface definition file:

- XATMI stub source file
- XATMI stub header file (used with UAPs written in C)
- XATMI stub copy file (used with UAPs written in COBOL)

To create a UAP that will use both OpenTP1 remote procedure calls and the XATMI interface, see the explanation about the `tpstbmk` command in *A. Using OpenTP1 Remote Procedure Calls and XATMI-interfaced API Functions in Combination*.

(3) Options

■ `-b`

Specify this option when creating an XATMI stub copy file which is to be used with COBOL UAPs. If this option is omitted, any XATMI stub copy file will not be created.

The name of an XATMI stub copy file is output in the format of *subtype-name*`.cbl`. The XATMI interface for the COBOL language does not support `X_C_TYPE` as a record type. If `X_C_TYPE` is specified in the XATMI interface definition, the tpstbmk command with the -b option specified will return with an error.

■ `-S` *stub-source-file-name* ~ <pathname>

Specify the relative or absolute pathname of the XATMI stub source file to be created.

If this option is omitted, an XATMI stub source file named *XXXXX*`_stbx.c` will be created in the current directory.

■ `-H` *stub-header-file-name* ~ <pathname>

Specify the relative or absolute pathname of the XATMI stub header file to be created.

If this option is omitted, an XATMI stub header file named *XXXXX*`_stbx.h` will be created in the current directory.

- `-i` *include-file-pathname* ~ <pathname>

  Specify the search path of the include file assigned to the `#include` statement in the XATMI interface definition file. The `tpstbmk` command searches the directory identified by the `-i` option for the include file.

  If the `-i` option is omitted, the current directory is searched for the include file.

  The `-i` option can be specified only once. If more than one search path is needed, `-i` must be followed by the desired paths separated by colons (:). The paths are searched in the order in which they are specified.

  When specifying a search path, you can use alphanumeric characters, underscores (_), slashes (/), and periods (.).

- `-m` *server-definition-file-pathname* ~ <pathname>

  Specify the search path of the server definition file given in the `called_servers` statement in the XATMI interface definition file. The `tpstbmk` command searches the directory identified by the `-m` option for the server definition file.

  If the `-m` option is omitted, the current directory is searched for the server definition file.

  The `-m` option can be specified only once. If more than one search path is needed, `-m` must be followed by the desired paths separated by colons (:). The paths are searched in the order in which they are specified.

  When specifying a search path, you can use alphanumeric characters, underscores (_), slashes (/), and periods (.).

(4) Command argument

- *definition-file-name* ~ <pathname>

  Specify the XATMI interface definition file name. Its suffix must be `.def`.

(5) Notes

- Each option to the `tpstbmk` command can be specified only once. If an option is specified more than once, the last specified value will be in effect.
- The names of files that the `tpstbmk` command can take as input or create as output are up to 255 characters long. Note, however, that a lower upper limit is used under some OSs.

1.4 Executing application programs

This section explains how to start and terminate UAPs and what environments are needed for executing UAPs.

1.4.1 Starting and terminating application programs

(1) Starting and terminating SUP

(a) Starting

The SUP is started when:

- OpenTP1 starts if the server name of the SUP is specified in the user service configuration definition, or
- The `dcsvstart` command is executed if the server name of the SUP is not specified in the user service configuration definition.

Before the SUP can request an SPP for service, the SPP must start. The SPP must have started before the SUP has.

(b) Terminating

Once an SUP has been started, it cannot be terminated normally by OpenTP1. Even when a command to exit OpenTP1 normally is executed, OpenTP1 will not terminate until all the SUPs in OpenTP1 termination.

When coding the SUP, design it so that it will terminate by itself. To cause the SUP to terminate abnormally if a problem occurs, design the SUP using a COBOL statement to exit the program so that it will terminate by itself.

The SUP cannot be terminated normally by the `dcsvstop` command. However, the SUP can be brought into forced termination by the `dcsvstop -f` command.

Do not terminate any SUP process by the `kill` command.

(2) Starting and terminating SPP and MHP

(a) Starting

The SPPs and MHPs belonging to one user server (service group) start at once. They start when:

- OpenTP1 starts if the server name of the SPPs and MHPs is specified in the user service configuration definition, or
- The `dcsvstart` command is executed if the server name of the SPPs and MHPs is not specified in the user service configuration definition.

If the multiserver facility is in use, the same number of user server processes as the specified number of resident processes are acquired. If the number of service requests increases, nonresident processes will start as well.

(b) Terminating

The SPP or MHP terminates when:

- Termination processing begins because one of the following OpenTP1 terminate commands is executed:

  `dcstop` (normal termination)

  `dcstop -n` (forced normal termination)

  `dcstop -a` (planned termination A)

  `dcstop -b` (planned termination B)

  `dcstop -f` (forced termination)

- The active online process enters termination steps because one of the following server termination commands is executed:

  `dcsvstop` (normal termination)

  `dcsvstop -f` (normal termination)

- The active online process is brought into termination by OpenTP1 because the maximum number of processes in the user service definition is exceeded;
- The SPP or MHP which is executing as a nonresident process finishes service processing; or
- The number of requests addressed to the service group decreases if loads on SPPs are distributed using a multiserver configuration.

Do not terminate any SPP or MHP process by the `kill` command.

(3) Starting and terminating UAPs that handle offline work

Users can start UAPs that handle offline work by any method. The UAPs are terminated by terminating the processes by the shell. Users are responsible for starting and terminating UAPs that handle offline work.

1.4.2 Operating environment of application programs started by OpenTP1

- The standard input (`stdin`), standard output (`stdout`), and standard error output (`stderr`) of SUPs, SPPs, and MHPs are redirected by OpenTP1.
- Input from an operator is not accepted even when the operator uses the COBOL `STOP` instruction. The `DISPLAY` instruction is available. However, the output data might be mixed with data output from another UAP if the `DISPLAY` instruction is used.

- When a UAP is activated, a directory `$DCDIR/tmp/home/`*user-server-name*`.`*XX* (where *XX* is a sequence number) is created. The UAP runs with this directory as the current working directory.

  You can change this directory by setting the `prc_current_work_path` operand in the system common definition.
- The user ID (UID) and group ID (GID) have the values specified at environment setup for the user server.
- The root directory remains as a forward slash (/).
- The following file descriptors are open during UAP execution:

  File descriptor 0: Standard input file descriptor

  File descriptor 1: Standard output file descriptor

  File descriptor 2: Standard error output file descriptor
- umask is 000.
- No control terminal is used.
- When creating a UAP process, OpenTP1 sets a UAP signal for the process automatically. Table 1-7 lists UAP signals which are set by OpenTP1.

Table 1-7: UAP signals set by OpenTP1

Signal name	Setting at the time of UAP process creation	Operation
`SIGHUP`	`SIG_DFL` (default)	`exit`
`SIGINT`	`SIG_IGN` (ignored)	`ignore`
`SIGQUIT`	`SIG_DFL` (default)	`core`
`SIGILL`	`SIG_DFL` (default)	`core`
`SIGTRAP`	`SIG_IGN` (ignored)	`ignore`
`SIGIOT`#	`SIG_DFL` (default)	`core`
`SIGABRT`#	`SIG_DFL` (default)	`core`
`SIGEMT`	`SIG_DFL` (default)	`core`
`SIGFEP`	`SIG_DFL` (default)	`core`
`SIGKILL`	--	`exit`
`SIGBUS`	`SIG_DFL` (default)	`core`
`SIGSEGV`	`SIG_DFL` (default)	`core`

Signal name	Setting at the time of UAP process creation	Operation
`SIGSYS`	`SIG_DFL` (default)	`core`
`SIGPIPE`#	`SIG_IGN` (ignored)	`ignore`
`SIGALRM`	`SIG_IGN` (ignored)	`ignore`
`SIGTERM`	`SIG_DFL` (default)	`exit`
`SIGUSR1`	`SIG_IGN` (ignored)	`ignore`
`SIGUSR2`	`SIG_IGN` (ignored)	`ignore`
`SIGCLD`	`SIG_DFL` (default)	`ignore`

Legend:

--: Not applicable.

Note

When specifying signal operations using UAP, do not stop the process by invoking `exit()` or `abort()` within the specified signal handler. When the process is stopped in the signal handler, the OpenTP1 system will shut down even if the signal interruption occurs during critical OpenTP1 processing. Furthermore, do not overwrite the value of the external variable `errno` in the signal handler.

#: These signals cannot be reset. When creating a UAP, do not reset the operation of the signal within the program.

1.4.3 Environment variables of application programs

UAP environment variables can be set for each user server at environment setup for the user server. However, the following environment variables are set by OpenTP1.

OpenTP1 sets the environment variables listed below:

- `DCDIR`: OpenTP1 home directory
- `DCCONFPATH`: Directory containing the OpenTP1 system definition file.
- `DCSVNAME`: User server name
- `DCSVGNAME`: Service group name (can be referenced only for SPP or MHP)
- `DCUAPCONFPATH`: Directory containing OpenTP1 user service definition files (only set when the files are to be stored in a different directory from `DCCONFPATH`)

In addition to the above, environment variables beginning with `DC` are used by OpenTP1. Since these environment variables are for reference only, do not change them. If changed, the system operation is undefined.

SUPs, SPPs, and MHPs that run under OpenTP1 do not inherit the environment variables set when the user logs in as OpenTP1 system administrator using telnet or other means. Set these environment variables again in the user service definition.

1.4.4 Troubleshooting

When using COBOL, specify environment variables as follows:

With COBOL/2:

Specify `0x00000efc` for the environment variable

`COBSIGMASK`.

With COBOL85:

Specify `1` for the environment variable `CBLCORE`.

(This value is set by OpenTP1 as default.)

If the environment variables are not specified as shown in the above, troubleshooting is impossible because a core file is not created when the UAP process terminates abnormally.

Chapter

2. Syntax of OpenTP1 Programs for COBOL-UAP Creation Programs

This chapter explains the syntax of UAP creation programs to be used when creating OpenTP1 UAPs in the COBOL language.

This chapter contains the following sections:

Format for explaining COBOL-UAP creation programs

To create an OpenTP1 UAP in COBOL language, call the COBOL-UAP creation program corresponding to the function in the OpenTP1 library by issuing the `CALL` statement. COBOL/2 and COBOL85 are available for coding in COBOL language.

COBOL-UAP creation programs are explained in the following format:

Format

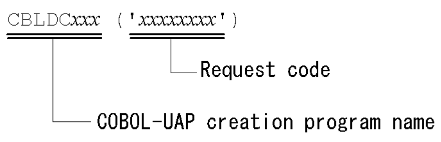
Explanations given under "Format" cover the format in which the COBOL-UAP creation program corresponding to a function in the OpenTP1 library is called with the `CALL` statement as well as how to specify the area.

The format is common to COBOL/2 and COBOL85. Specify a value for a data name according to the data format of the `PICTURE` clause shown here. If the value to be specified is determined, the value is written in the `VALUE` clause. For files and data which must have unique names, assign a specific name to each file and data unless otherwise specified. The length of a character string specified as a data name must comply with the specifications of COBOL in use.

When writing a program in COBOL, the COBOL language templates, which are OpenTP1 samples, can be used. Modifying the templates according to the program to be coded saves programmers the trouble of coding `DATA DIVISION` from the beginning. The COBOL language templates are stored in the `/$DCDIR/examples/COBOL/` directory. Each template is stored in a file named according to each system service. The template file name is `DC`*xxx*`.cbl` (*xxx* is the last three characters of the COBOL-UAP creation program name.)

Description

Describes the facilities of the COBOL-UAP creation programs. From here on, the format of a COBOL-UAP creation program is represented as follows:



Data areas set in the UAP

Indicates the names of data which is specified in `DATA DIVISION` and for which values are specified in data areas when the COBOL-UAP creation program is called.

Specify a value for each data name according to the explanation. If a value is not always specified in a data area, the explanation of the data name is enclosed in brackets [ ] when the value is specified for the argument.

Data area(s) to which a value(s) is returned from OpenTP1

Indicates the names of data which is specified in `DATA DIVISION` and to which a value(s) is returned from OpenTP1 after the `CALL` statement is executed. After executing the `CALL` statement, reference the contents of the data area indicated with the data name. If a value is not always returned to a data area from OpenTP1, the explanation of the data name is enclosed in brackets [ ] when the value is returned.

Data area(s) to which a value(s) is passed from the client UAP

Indicates a data area(s) to which a value(s) is returned from the client UAP when the service program is used. Execute service program processing referencing the contents of the data area.

Data area(s) to which a value(s) is returned from the server UAP

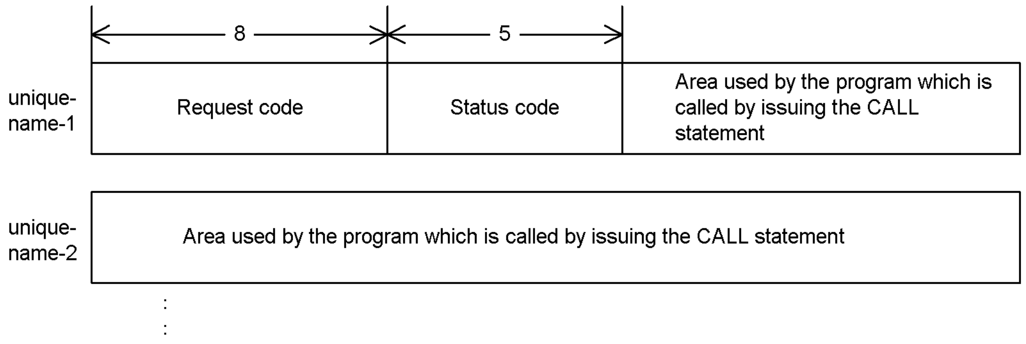
Indicates the names of data whose a value(s) is returned from the service program when a synchronous-response-type RPC or asynchronous-response-type RPC is used. The UAP that called `CBLDCRPC('CALL    ')` or `CBLDCRPC('POLLANYR')` can reference the value of the data area shown here.

Status codes

Status codes returned when the `CALL` statement is executed are explained in a table. The status code indicates whether the COBOL-UAP creation program was executed normally. If an error occurs, the status code indicates the error status.

A COBOL language status code comprises five digits. It is included in the first unique name specified in the `USING` clause. The following explains the relationship between the status code and unique names specified in the `USING` clause of the `CALL` statement:

```
CALL  'name-of-program-to-be-called'  USING  unique-name-1 unique-name-2...
```


8
5
unique-name-1
Request code
Status code
Area used by the program which is called by issuing the CALL statement
unique-name-2
Area used by the program which is called by issuing the CALL statement
:
:

Example

Provided only for COBOL-UAP creation programs with which specification samples are necessary.

Note(s)

Explains a note(s) on using the facilities of COBOL-UAP creation programs.

Creating main and service programs

This section explains the syntax of main and service programs in OpenTP1 UAPs. Main and service programs must be created for an SPP or MHP; only a main program must be created for an SUP.

- Create a main program (SUP, SPP, MHP)
- Create a service program (SPP)
- Create a service program (MHP)

TP1/LiNK can use only SUPs and SPPs as OpenTP1 UAPs. MHPs, CGWs, and SGWs are not available for TP1/LiNK.

Create a main program (SUP, SPP, MHP)

Format

When creating main programs, comply with the specifications of the COBOL language for coding. To terminate processing, issue `STOP RUN`. OpenTP1 does not limit creation of main programs except that `STOP RUN` must be issued to terminate processing.

Description

After the UAP process starts, the OS first calls the main program.

- SUP main program

  The following OpenTP1 COBOL-UAP creation programs are always issued in the SUP main program:

  1. `CBLDCRPC('OPEN    ')`

     Start an application program
  2. `CBLDCADM('COMPLETE')`

     Report the completion of user server start processing
  3. `CBLDCRPC('CLOSE   ')`

     Terminate an application program (called after job termination)

  In addition to the above COBOL-UAP creation programs, those for requesting the initialization processing of UAP processes required for jobs, termination processing, and remote procedure call (`CBLDCRPC('CALL    ')`) can also be used in the SUP main program.

- SPP main program

  Service programs created as services which are provided by an SPP are grouped into one executable file. An executable file comprising one main program and multiple service programs corresponds to a service group.

  The following OpenTP1 COBOL-UAP creation programs are always called from the SPP main program:

  1. `CBLDCRPC('OPEN    ')`

     Start an application program
  2. `CBLDCRSV('MAINLOOP')`

     Start an SPP service

3. `CBLDCRPC('CLOSE   ')`

   Terminate an application program (called after job termination)

To use an MCF facility (call CBLDCMCF) with an SPP service, call the following COBOL-UAP creation programs:

- `CBLDCMCF('OPEN    ')` - Open the MCF environment (between (1)and (2))
- `CBLDCMCF('CLOSE   ')` - Close the MCF environment (between (2)and (3))

After initialization processing, the main program stops when `CBLDCRSV('MAINLOOP')` is called. Meanwhile, the main program performs processing requested by service programs. In addition to the above OpenTP1 `CALL` statements, `CALL` statements for requesting the initialization of SPP processes required for jobs, termination processing, and remote procedure call (`CBLDCRPC('CALL    ')`) can also be used in the main program.

MHP main program

Service programs created as applications for message processing are grouped into one executable file. An executable file comprising one main program and multiple service programs corresponds to a service group. The service group name must be unique in the domain (in the entire network).

The following OpenTP1 COBOL-UAP creation programs are always called from the MHP main program:

1. `CBLDCRPC('OPEN    ')`

   Start an application program
2. `CBLDCMCF('OPEN    ')`

   Open the MCF environment
3. `CBLDCMCF('MAINLOOP')`

   Start an MHP service
4. `CBLDCMCF('CLOSE   ')`

   Close the MCF environment (called after job termination)
5. `CBLDCRPC('CLOSE   ')`

   Terminate an application program (called after job termination)

The MHP having the service program corresponding to the application name is started. After initialization processing, the main program stops when `CBLDCMCF('MAINLOOP')` is called. Meanwhile, the main program performs processing requested by service programs. In addition to the above COBOL-UAP creation programs, those for requesting the initialization processing of UAP processes required for jobs, termination processing, and remote procedure call

(`CBLDCRPC('CALL   ')`) can also be used in the MHP main program.

Note

When creating a main program with COBOL, use `STOP RUN` to terminate processing when the creation terminates normally. Do not use `EXIT PROGRAM`.

Create a service program (SPP)

Format

```
PROGRAM-ID.    program-name.
LINKAGE SECTION.
01  unique-name-1.
    02  data-name-A    PIC X(n).
01  unique-name-2.
    02  data-name-B    PIC S9(9) COMP.
01  unique-name-3.
    02  data-name-C    PIC X(n).
01  unique-name-4.
    02  data-name-D    PIC S9(9) COMP.
PROCEDURE DIVISION USING unique-name-1  unique-name-2
                         unique-name-3  unique-name-4
                  :
                  :
Service processing
                  :
                  :
EXIT PROGRAM.
```

Description

The SPP service program executes a service and returns the results. `CBLDCRPC('CALL    ')` called from the client UAP requests a service.

Create the service program in the above format as required. The service name corresponds to the program ID of the service program. Specify the correspondence when setting the UAP execution environment. The methods of setting the UAP execution environment are as follows:

- For TP1/Server Base, specify it in the user service definition.
- For TP1/LiNK, specify it interactively by executing a command for setting the UAP execution environment.

Data area specification

The values listed below are passed to the data areas of the service program. These values are specified by the client UAP for `CBLDCRPC('CALL    ')`.

- Input parameter (*data-name-A*)
- Input parameter length (*data-name-B*)
- Response length (*data-name-D*)

  The values specified for the input parameter and input parameter length in the client UAP are passed to the service program as they are. (The expression formats of character codes and numbers are not converted.) The length specified in the client UAP is passed as the response length.

Specify the following values for the data areas of the service program:

- Service program response (*data-name-C*)
- Length of the service program response (*data-name-D*)

Set a response for *data-name-C*, set the response length for *data-name-D* then return the service program.

A response is sent to the service client UAP regardless of whether the service program was executed as a transaction or whether commitment or rollback processing was executed. Create a response with which the service program informs the client UAP of the occurrence of an error if necessary.

If the service program terminates abnormally because the program malfunctions, this abnormal termination is not reported to the client UAP. The client UAP gets to know the abnormal termination of the service program when the maximum response wait time specified in the user service definition is exceeded.

Data areas to which values are passed from the client UAP

- *data-name-A*

  The input parameter set in the client UAP is passed.

- *data-name-B*

  The input parameter length set in the client UAP is passed.

- *data-name-D*

  The response length set in the client UAP is passed.

Data areas whose values are set in the UAP

- *data-name-C*

  The response from the service program is set here. Before the service program can return, it must set the processing result in the data-named area.

- *data-name-D*

  Specify the length of the actual response from the service program. Set a numeric value which is equal to or smaller than the *data-name-D* value passed from the client UAP.

Notes on service program processing

1. The service program called by `CBLDCRPC('CALL    ')` (with `1` set in *data-name-C*) of a non-response-type RPC cannot reference *data-name-C* or *data-name-D*.

2. If the service program is a COBOL initialization program (`INITIAL` clause specified in the program header), the data item values are initialized each time a service request is used. If the `INITIAL` clause is not specified in the program header, the values upon the previous service request remain in the data items. Thus, initialize the values before using them if necessary.
3. The following COBOL-UAP creation programs cannot be called from the SPP service program:
   - `CBLDCRPC('OPEN    ')` - Start an application program
   - `CBLDCADM('COMPLETE')` - Report the completion of user server start processing
   - `CBLDCRSV('MAINLOOP')` - Start an SPP service
   - `CBLDCRPC('CLOSE   ')` - Terminate an application program

   Do not use a statement in the service program to exit the program. If used, UAP operation is not guaranteed. Once a system call is used to create a child process, COBOL-UAP creation programs can no longer be called from the child process.
4. In order to call an MCF facility (`CBLDCMCF`) from the SPP service program, the main program should call the following COBOL-UAP creation programs:
   - `CBLDCMCF('OPEN  ')` - Open the MCF environment
   - `CBLDCMCF('CLOSE ')` - Close the MCF environment
5. An SPP service program cannot call `CBLDCMCF('RECEIVE ')`.
6. Do not execute an operation or reference that extends beyond the area of the input parameter length passed to *data-name-B*, for the input parameter passed to *data-name-A*. If you execute such an operation or reference, operation cannot be guaranteed. The process may terminate abnormally.
7. Before an MCF function (`CBLDCMCF`) can be called from an SPP service program, data areas such as *unique-name-1* must begin at an even-numbered address. If a data area such as *unique-name-1* begins at an odd-numbered address, a bus error will occur.

   For example, assume that the *unique-name-3* data area to be used for `CBLDCMCF` is defined as an array, the send data contains an odd number of bytes, and SYNC is not specified in the structure (no boundary adjustment used). In this situation, using the second element of the array as a `CBLDCMCF` parameter will cause a bus error during function execution.

Relationship between transactions and the service program

When the transaction attribute has been specified in the UAP execution environment, and the client UAP has been executed as a transaction, the service program is also executed as a transaction. In this case, do not call `CBLDCTRN('BEGIN   ')` from the service program.

Commitment or rollback processing is ensured for all global transaction services. When the service program operating as a transaction branch is terminated by `EXIT PROGRAM`, the service program is assumed to request normal termination of the transaction branch.

When the transaction attribute has been specified in the UAP execution environment, but the client UAP has not been executed as a transaction, the service program is not executed as a transaction. To execute the service program as a transaction, call `CBLDCTRN('BEGIN  ')` and the program which acquires a synchronization point at any time.

When no transaction attribute is specified in the UAP execution environment, the service program cannot be executed as a transaction by `CBLDCTRN('BEGIN  ')` from the service program.

Status code

There is no status code. OpenTP1 does not reference status codes. The return of `-1` from the service program would not mean a request for rollback.

Create a service program (MHP)

Format

```
PROGRAM-ID.     program-name.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
         :
DATA DIVISION.
WORKING-STORAGE SECTION.
         :
         :
PROCEDURE DIVISION.
         :
         :
Service processing
         :
         :
EXIT PROGRAM.
```

Description

The MHP service program executes a service and returns the execution results. When the MCF receives a message, the MHP having the service program that corresponds to the application name is started.

Create the MHP service program in the above format as required. The service name corresponds to the program ID of the service program. Specify this correspondence in the user service definition of the process that executes the service program.

Specify the correspondence of the service name and the application name in the MCF application definition.

Data area specification

None

Notes on service program processing

1. If the service program is a COBOL initialization program (the `INITIAL` clause is specified in the program header), the values of data items and others are initialized each time the service program is executed. If the `INITIAL` clause is not specified in the program header, data items still assume the values which were given during the previous service program processing. Initialize them if necessary.
2. The following COBOL-UAP creation programs cannot be called from the MHP service program:

   `CBLDCRPC('OPEN     ')` - Start an application program

   `CBLDCRPC('CLOSE    ')` - Terminate an application program

`CBLDCADM('COMPLETE')` - Report the completion of user server start processing

`CBLDCRSV('MAINLOOP')` - Start an SPP service

`CBLDCMCF('MAINLOOP')` - Start an MHP service

`CBLDCMCF('OPEN    ')` - Open the MCF environment

`CBLDCMCF('CLOSE   ')` - Close the MCF environment

Do not use the statement in the service program to exit the program. If used, UAP operation is not guaranteed. Once a system call is used to create a child process, COBOL-UAP creation programs can no longer be called from the child process.

3. Another UAP cannot request a service to the MHP service program by `CBLDCRPC('CALL ')`.

4. Data areas such as `CBLDCMCF` *unique-name-1* must begin at an even-numbered address. If a data area such as *unique-name-1* begins at an odd-numbered address, a bus error will occur.

   For example, assume that the *unique-name-3* data area to be used for `CBLDCMCF` is defined as an array, the send data contains an odd number of bytes, and SYNC is not specified in the structure (no boundary adjustment used). In this situation, using the second element of the array as a `CBLDCMCF` parameter will cause a bus error during function execution.

Status code

There is no status code. The return of -1 from the service program would not mean a request for rollback.

System operation management (CBLDCADM)

This section gives the syntax and other information of the following COBOL-UAP creation programs which allow the UAP to use various OpenTP1 system facilities:

- `CBLDCADM('COMMAND ')` - Execute an operation command
- `CBLDCADM('COMPLETE')` - Report the completion of user server start processing
- `CBLDCADM('STATUS  ')` - Report the status of a user server

The COBOL-UAP creation programs for system operation management can be used in UAPs of both TP1/Server Base and TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language template for system operation management (CBLDCADM) is stored in `DCADM.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCADM ('COMMAND ') - Execute an operation command

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCADM'  USING  unique-name-1  unique-name-2
                         unique-name-3  unique-name-4
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'COMMAND '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
    02  data-name-C    PIC S9(9) COMP.
01  unique-name-2.
    02  data-name-E    PIC 9(9) COMP.
    02  data-name-G    PIC X(n).
01  unique-name-3.
    02  data-name-H    PIC 9(9) COMP.
    02  data-name-J    PIC X(n).
01  unique-name-4.
    02  data-name-K    PIC 9(9) COMP.
    02  data-name-M    PIC X(n).
```

Description

`CBLDCADM('COMMAND ')` passes data name G from the UAP to `sh(1)` as in the case of command entry in online mode. The process waits until the shell completes its processing, and returns the exit status of the shell. When the command processing ends, the standard output information and standard error information are returned to the respective areas.

If you want to use UAPs which execute a command, add the directory containing the commands to the search path. Use any of the following methods for addition to the search path.

- Specify the path name of the command in the `prcsvpath` operand of the process service definition.
- Add the search path with the `prcpath` command.
- Specify environment variable `putenv PATH` in the user service definition.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'COMMAND∆'` for the request code of the command to be executed.

- *data-name-Z*

  Specify `0`.

- *data-name-E*

  Specify the length of the command set for *data-name-G*.

- *data-name-G*

  Specify the character string of the command.

- *data-name-H*

  The execution results of the command are output to the standard output file. Specify the size of the contents (value returned to *data-name-J*) in bytes. Pre-allocate the area in size of the number of bytes that is to be specified for *data-name-H*. The area must begin from the address pointed to by *data-name-J*. The number of bytes to be specified for *data-name-H* must be decided according to the command executed by the UAP.

  After processing terminates, the length of the character string stored in *data-name-J* is returned from OpenTP1. The length of *data-name-H* itself is not included.

- *data-name-K*

  The execution results of the command are output to the standard error output file. Specify the size of the contents (value returned to *data-name-M*) in bytes. Pre-allocate the area in size of the number of bytes that is to be specified for *data-name-K*. The area must begin from the address pointed to by *data-name-M*. The number of bytes to be specified for *data-name-K* must be decided according to the command executed by the UAP.

  After processing terminates, the length of the character string stored in *data-name-M* is returned from OpenTP1. The length of *data-name-K* itself is not included.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-C*

  A shell termination code# is returned indicating whether the command terminated normally or abnormally.

  #: Denotes an `sh(1)` termination status in the format specified by `waitpid(2)`.

- *data-name-H*

  The length of the character string stored in *data-name-J* is returned after processing terminates. The length of *data-name-H* itself is not included.

- *data-name-J*

  This area stores characters output to the standard output file by the operation command. The maximum length of a character string which can be stored is as specified for *data-name-H*. If the length of a character string exceeds the value specified for *data-name-H*, the excess characters are truncated. If the character string exceeds the capacity of the pipe, the excess characters are also truncated.

- *data-name-K*

  The length of the character string stored in *data-name-M* is returned after processing terminates. The length of *data-name-K* itself is not included.

- *data-name-M*

  This area stores characters output to the standard error output file by the operation command. The maximum length of a character string which can be stored is as specified for *data-name-K*. If the length of a character string exceeds the value specified for *data-name-K*, the excess characters are truncated. If the character string exceeds the capacity of the pipe, the excess characters are also truncated.

Status codes

Status code	Explanation
`00000`	The shell termination code is 0 (normal termination of the command execution). The character string was stored in the standard output area and the standard error output area.
`01801`	The shell termination code is not 0 (abnormal termination of the command execution). Standard output data and standard error output data were stored in the areas.
`01802`	The value specified for the data name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
`01803`	All the standard output data could not be stored in the area.
`01804`	All the standard error output data could not be stored in the area.
`01805`	Both the standard output data and the standard error output data could not be stored in the areas.
`01806`	A system call (`close`, `pipe`, `dup`, or `read`) could not be executed.
`01807`	`CBLDCRPC('OPEN ')` was not called.
`01808`	The memory became insufficient.

Note

Be careful not to duplicate the command name between directories that are specified as search paths. The correct command will not execute if the command name is duplicated. In addition, be careful not to duplicate the command name with that of the command group provided by OpenTP1 (under `$DCDIR/bin`).

CBLDCADM ('COMPLETE') - Report the completion of user server start processing

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCADM'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8)  VALUE 'COMPLETE'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-Z    PIC S9(9)  COMP VALUE ZERO.
```

Description

`CBLDCADM('COMPLETE')` notifies OpenTP1 that SUP activation has been completed. SUP activation is completed when `CBLDCADM('COMPLETE')` returns normally.

For SPPs and MHPs, the normal termination of `CBLDCRSV('MAINLOOP')` and `CBLDCMCF('MAINLOOP')` is regarded as the completion of activation. Therefore, there is no need for calling `CBLDCADM('COMPLETE')`.

`CBLDCADM('COMPLETE')` cannot be called from a UAP that handles offline work.

Data areas whose values are set in the UAP

- *data-name-A*

  `VALUE 'COMPLETE'` is assigned to the request code indicating the report that user server activation has been completed.

- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination

Status code	Explanation
`01802`	The request code (*data-name-A*) is invalid.
`01830`	An error occurred during communication between processes.
`01831`	The value specified for the data name is invalid.
`01832`	A status information input/output error occurred.
`01833`	The user server is not being started/restarted normally.

CBLDCADM('STATUS ') - Report the status of a user server

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCADM'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8)  VALUE 'STATUS  '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(5).
    02  data-name-Z    PIC S9(9)  COMP VALUE ZERO.
```

Description

`CBLDCADM ('STATUS     ')` reports the status of the user server that called the program. The user server status is reported with the status code.

Data areas whose values are set in the UAP

- *data-name-A*

  `VALUE 'STATUS△△'` is assigned to the request code for the report of user server status.

- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-C*

  The user server status is returned when `00000` is returned to *data-name-B*.

Status codes

Status codes returned to *data-name-B* (indicating whether an error occurred):

Status code	Explanation
`00000`	Normal termination

Status code	Explanation
`01802`	The request code (*data-name-A*) is invalid.
`01830`	An error occurred during communication between processes.
`01831`	The value specified for the data name is invalid.
`01832`	A status information input/output error occurred.
`01833`	This program was called from a UAP that handles offline work. This program cannot be called from a UAP that handles offline work.
	`CBLDCRPC('OPEN')` was not called.

Status codes returned to *data-name-C* (indicating the user server status) when `00000` is returned to *data-name-B* (indicates the status of the user server):

Status code	Explanation
`00001`	The user server is being started normally.
`00002`	The user server is being restarted normally.
`00003`	The user server is in online mode.
`00004`	The user server is being terminate.

Audit log output (CBLDCADT)

This section gives the syntax and other information of the following COBOL-UAP creation program which is used to output audit log data from a UAP:

- `CBLDCADT('PRINT   ')` - Output audit log data

CBLDCADT('PRINT   ') - Output audit log data

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCADT' USING unique-name-1  unique-name-2  unique-name-3
```

- DATA DIVISION specification

```
01 unique-name-1.
   02 data-name-A  PIC X(8) VALUE 'PRINT   '.
   02 data-name-B  PIC X(5).
   02 FILLER    PIC X(3).
   02 data-name-Z  PIC S9(9) COMP VALUE ZERO.
01 unique-name-2.
   02 data-name-C  PIC X(12).
   02 data-name-D  PIC X(3).
   02 FILLER    PIC X(1).
   02 data-name-E  PIC S9(9) COMP.
   02 data-name-F  PIC S9(9) COMP.
   02 data-name-G  PIC S9(9) COMP.
01 unique-name-3.
   02 data-name-H  PIC S9(9) COMP.
   02 data-name-I  PIC X(n).
```

Description

`CBLDCADT('PRINT   ')` outputs to the audit log file the following information items, in addition to the information specified as arguments: header information, serial number, date and time, relevant program name, relevant process ID, location, subject identification information, object information, object location information, request sender host, and location identification information. The relevant program is OpenTP1, which generates the audit log data. If an error occurs during output of audit log data, an error message is sent to the standard error output and syslog.

In OpenTP1, numbers from 34000 to 34999 are assigned for message IDs used by `CBLDCADT('PRINT   ')`. If you create a UAP, make sure that the message IDs output by the UAP are in the range from 34000 to 34999.

For details on the items output as audit log data, see the manual *OpenTP1 Programming Guide*.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'PRINT`ΔΔΔ`'` as the request code that indicates a request for outputting audit log data.

- *data-name-Z*

  Specify the value 0.

- *data-name-C*

  Specify the identifier of an audit log entry (message ID).

  Specify the identifier in the format `KFCA`*nnnnn*`-`*x* (11 characters) and follow the identifier with a null character. For *nnnnn*, specify a five-digit serial number in the range from 34000 to 34999. For *x*, specify `E`, `W`, or `I` as the message type according to the type of information provided by the audit log entry to be output.

- *data-name-D*

  Specify any value that identifies the UAP that called the function `CBLDCADT('PRINT ')` (calling program ID). The value you set must be two numeric characters, alphabetic characters, or symbols followed by a null character. In the audit log, the format is `*`*AA*, with an asterisk (`*`) prefixed (*AA*: character string specified in *data-name-D*).

- *data-name-E*

  Specify one of the following numeric values as the audit event type to be included in the audit log data.

Audit event type	Value	Meaning
StartStop	`1000`	Audit event related to a start or stop operation
Authentication	`1001`	Audit event related to identification or authentication
AccessControl	`1002`	Audit event related to access control
ConfigurationAccess	`1003`	Audit event related to the configuration definition
Failure	`1004`	Audit event related to failures
LinkStatus	`1005`	Audit event related to the linkage status
ExternalService	`1006`	Audit event related to external services
ContentAccess	`1007`	Audit event related to access to important information
Maintenance	`1008`	Audit event related to maintenance
AnomalyEvent	`1009`	Audit event related to anomalies
ManagementAction	`1010`	Audit event related to management operation

  For details on audit event types, see the manual *OpenTP1 Operation*.

- *data-name-F*

  Specify one of the following values as the audit event result to be included in the audit log data:

Audit event result	Value	Meaning
Success	`2000`	Successful event
Failure	`2001`	Failed event
Occurrence	`2002`	Event that cannot be categorized as success or failure

- *data-name-G*

  Specify the value to be included as operation information in the audit log data. Make sure that you specify one of the following reserved words according to the audit event type specified by *data-name-E*. If you specify the value 0, this item will not be included in the audit log data.

Table 2-1: Correspondence between audit event types and reserved words

Audit event type	Reserved word	Value	Meaning
`StartStop` (start or stop operation)	`Start`	`3000`	Start or activation
	`Stop`	`3001`	Termination or stop
`Authentication` (identification or authentication)	`Login`	`3002`	Login
	`Logout`	`3003`	Logout
	`Logon`	`3004`	Logon
	`Logoff`	`3005`	Logoff
	`Disable`	`3006`	Account disabled
`AccessControl` (access control)	`Enforce`	`3007`	Enforcement
`ConfigurationAccess` (configuration definition)	`Refer`	`3008`	Reference
	`Add`	`3009`	Addition
	`Update`	`3010`	Updating
	`Delete`	`3011`	Deletion
`Failure` (failures)	`Occur`	`3012`	Occurrence
`LinkStatus` (linkage status)	`Up`	`3013`	Linkage active
	`Down`	`3014`	Linkage inactive

Audit event type	Reserved word	Value	Meaning
`ExternalService` (external services)	`Request`	`3015`	Request
	`Response`	`3016`	Response
	`Send`	`3017`	Sending
	`Receive`	`3018`	Receiving
`ContentAccess` (access to important information)	`Refer`	`3008`	Reference
	`Add`	`3009`	Addition
	`Update`	`3010`	Updating
	`Delete`	`3011`	Deletion
`Maintenance` (maintenance)	`Install`	`3019`	Installation
	`Uninstall`	`3020`	Uninstallation
	`Update`	`3010`	Updating
	`Backup`	`3021`	Backup
	`Maintain`	`3022`	Maintenance work
`AnomalyEvent` (anomalies)	`Occur`	`3012`	Occurrence
`ManagementAction` (management operation)	`Invoke`	`3023`	Invocation (the administrator)
	`Notify`	`3024`	Notification (the administrator)

- *data-name-H*

  Specify the length of the character string to be included as the freely specified description in the audit log data. If you specify the value 0, this item will not be included in the audit log data.

- *data-name-I*

  Set the freely specified description to be included in the audit log data.

  You can use numeric characters, alphabetic characters, symbols, spaces, double quotation marks (`"`), and commas (`,`). The description can have a maximum of 1024 characters.

  The description specified in *data-name-I* is enclosed in double quotation marks (`"`). If a double quotation mark (`"`) is included in the description, the double quotation mark is prefixed by another double quotation mark.

Data areas whose values are set in OpenTP1

- *data-name-B*

  A five-digit number is returned as the status code.

Status code

Status code	Meaning
00001	Output of audit log data has been disabled. Possible causes are as follows: • The `log_audit_out` operand in the log service definition has been set to `N` or has not been specified. • The `log_audit_suppress` operand has been set to `Y` in the log service definition.
	The message ID specified in *data-name-C* has not been specified in the `log_audit_message` operand in the log service definition.
	An invalid message has been specified.
00000	The function terminated normally.
01900	The value specified in a data area is incorrect.
01904	Definition analysis failed.
01999	The `dc_rpc_open` function was not issued.
01997	An error other than the above occurred.

DAM file service (online facility: CBLDCDAM, offline facility: CBLDCDMB)

This section gives the syntax and other information of the following COBOL-UAP creation programs which are used for DAM file service:

Functions that can only be used in an online environment

- `CBLDCDAM('CLOS')` - Close a logical file
- `CBLDCDAM('END ')` - Terminate using an unrecoverable DAM file
- `CBLDCDAM('HOLD')` - Shut down a logical file
- `CBLDCDAM('OPEN')` - Open a logical file
- `CBLDCDAM('READ')` - Input a logical file block
- `CBLDCDAM('REWT')` - Update a logical file block
- `CBLDCDAM('RLES')` - Release a logical file from the shutdown state
- `CBLDCDAM('STAT')` - Reference the status of a logical file
- `CBLDCDAM('STRT')` - Start using an unrecoverable DAM file
- `CBLDCDAM('WRIT')` - Output a logical file block

Functions that can only be used in an offline environment

- `CBLDCDMB('BSEK')` - Seek a physical file block
- `CBLDCDMB('CLOS')` - Close a physical file
- `CBLDCDMB('CRAT')` - Allocate a physical file
- `CBLDCDMB('DGET')` - Input directly a physical file block
- `CBLDCDMB('DPUT')` - Output directly a physical file block
- `CBLDCDMB('GET ')` - Input a physical file block
- `CBLDCDMB('OPEN')` - Open a physical file
- `CBLDCDMB('PUT ')` - Output a physical file block

The COBOL-UAP creation programs for DAM file service can be used only in UAPs of TP1/Server Base. They cannot be used in UAPs of TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language templates for DAM file service (CBLDCDAM, CBLDCDMB) are stored in `DCDAM.cbl` and `DCDMB.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCDAM('CLOS') - Close a logical file

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDAM'  USING  unique-name-1   unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(8) VALUE 'DCDAMSVC'.
    02  data-name-B     PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C     PIC X(8).
    02  FILLER         PIC S9(9) COMP.
    02  FILLER         PIC S9(9) COMP.
    02  data-name-H     PIC S9(9) COMP.
    02  FILLER         PIC X(28).
01  unique-name-2.
    02  data-name-E     PIC X(4) VALUE 'CLOS'.
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z     PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCDAM ('CLOS')` closes logical files.

- When accessing a recoverable DAM file:

  If a logical file is not closed before transaction processing terminates, the DAM service closes the file if it was opened in the transaction range when synchronization point processing was executed. However, the DAM service does not close a logical file which was opened outside the transaction range (before `CBLDCTRN('BEGIN  ')` was called) or which was an unrecoverable DAM file.

  If you open a logical file before starting the transaction, close the file when terminating the UAP processing.

- When accessing an unrecoverable DAM file:

  Since this type of file does not synchronize with transaction processing, `CBLDCDAM('CLOS')` can be called as required when a logical file is closed. Note that the open logical file should be closed by `CBLDCDAM('CLOS')` before calling `CBLDCDAM('END ')`.

  When closing a logical file, specify the file descriptor returned from

`CBLDCDAM('OPEN').`

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMSVC'` for the interface code used with the DAM file.

- *data-name-C*

  Specify a logical file name with up to 8 characters. If the specified logical file name comprises less than 8 characters, pad the remaining portion with space.

- *data-name-E*

  Specify `VALUE 'CLOS'` for the request code indicating that the logical file is closed.

- *data-name-H*

  Specify the file descriptor returned when the logical file was opened.

- *data-name-Z*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	The logical file was closed normally.
`01600`	`CBLDCRPC('OPEN    ')` was not called.
	A DAM file opened outside the transaction range is closed within the transaction range. (This error is returned only when accessing a recoverable DAM file.)
	`'N'` is specified for `atomic_update` in the user service definition. (This error is returned only when accessing a recoverable DAM file.)
	`CBLDCDAM('STRT')` was not called. (This error is returned only when accessing an unrecoverable DAM file.)
	The UAP is incorrectly linked as follows: • The library (`-ltdam`) to be used for access to a TAM file using the DAM service API is linked incorrectly. • The definition of the resource manager for transaction control object files is incorrect.

Status code	Explanation
01603	The specified file descriptor is not the one which was acquired by opening the file normally.
	The DAM file is not open.
01690	The interface code (*data-name-A*) is invalid.
01691	The request code (*data-name-E*) is invalid.

CBLDCDAM('END ') - Terminate using an unrecoverable DAM file

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDAM'  USING  unique-name-1   unique-name-2
                         unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMSVC'.
    02  data-name-B    PIC X(5).
    02  FILLER        PIC X(3).
    02  FILLER        PIC X(8).
    02  FILLER        PIC S9(9) COMP.
    02  FILLER        PIC S9(9) COMP.
    02  FILLER        PIC S9(9) COMP.
    02  FILLER        PIC X(28).
01  unique-name-2.
    02  data-name-C    PIC X(4) VALUE 'END '.
    02  FILLER        PIC X(1).
    02  FILLER        PIC X(1).
    02  FILLER        PIC X(1).
    02  FILLER        PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCDAM('END ')` declares that use of an unrecoverable DAM file is terminated.

Call `CBLDCDAM('END ')` whenever calling `CBLDCDAM('STRT')`. Otherwise, the resource used to access an unrecoverable DAM file remains unreleased until the UAP terminates.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMSVC'` for the interface code used with the DAM file.

- *data-name-C*

  Specify `VALUE 'END∆'` for the request code indicating that use of an unrecoverable DAM file is terminated.

- *data-name-Z*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  The status code of 5 digit is returned.

Status codes

Status code	Explanation
`00000`	Normal termination. Use of an unrecoverable DAM file is terminated.
`01600`	`CBLDCRPC('OPEN    ')` was not called.
`01605`	`CBLDCDAM('STRT')` was not called.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-C*) is invalid.

CBLDCDAM('HOLD') - Shut down a logical file

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDAM'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMSVC'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(8).
    02  FILLER         PIC S9(9) COMP.
    02  FILLER         PIC S9(9) COMP.
    02  FILLER         PIC X(32).
01  unique-name-2.
    02  data-name-E    PIC X(4) VALUE 'HOLD'.
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCDAM ('HOLD')` shuts down a logical file. After a logical file is logically shut down, a logical shutdown error is always returned if another UAP issues an access request for the logical file specified here.

- When accessing a recoverable DAM file:

  If the logical file specified here is under commitment processing in another transaction when `CBLDCDAM('HOLD')` is called, the logical file is shut down after the commitment processing terminates. Even if the commitment processing is not completed, control returns to the UAP that called `CBLDCDAM('HOLD')`.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMSVC'` for the interface code used with the DAM file.

- *data-name-C*

  Specify a logical file name with up to 8 characters. If the specified logical file name comprises less than 8 characters, pad the remaining portion with space.

- *data-name-E*

  Specify `VALUE 'HOLD'` for the request code indicating that the logical file is in shutdown state.

- *data-name-Z*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	The specified logical file was shut down normally.
`01600`	`CBLDCRPC('OPEN    ')` was not called.
	`'N'` is specified for `atomic_update` in the user service definition. (This error is returned only when accessing a recoverable DAM file.)
	`CBLDCDAM('STRT')` was not called. (This error is returned only when accessing an unrecoverable DAM file.)
	The UAP is incorrectly linked as follows: • The library (`-ltdam`) to be used for access to a TAM file using the DAM service API is linked incorrectly. • The definition of the resource manager for transaction control object files is incorrect.
`01601`	The specified logical file name has not been defined.
`01607`	The memory became insufficient.
`01610`	The value specified as the logical file name is invalid.
`01618`	The version of the DAM library linked to the UAP does not allow the UAP to operate with the current DAM service.
`01625`	The logical file name specified for *data-name-C* is in logical shutdown state.
`01626`	The logical file name specified for *data-name-C* is in shutdown state due to an error.
`01628`	The DAM file to be accessed is protected by the security facility. The UAP attempting to shut down the logical file has no access permission.
`01646`	The DAM file to be shut down is protected by the security facility. No ACL exists for the file.
`01690`	The interface code (*data-name-A*) is invalid.

Status code	Explanation
`01691`	The request code (*data-name-E*) is invalid.

CBLDCDAM('OPEN') - Open a logical file

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDAM'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMSVC'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(8).
    02  FILLER         PIC S9(9) COMP.
    02  FILLER         PIC S9(9) COMP.
    02  data-name-H    PIC S9(9) COMP.
    02  FILLER         PIC X(28).
01  unique-name-2.
    02  data-name-E    PIC X(4) VALUE 'OPEN'.
    02  data-name-F    PIC X(1).
    02  data-name-G    PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCDAM ('OPEN')` opens a logical file.

- When accessing a recoverable DAM file:

  Whether to apply file lock or block lock to the logical file is specified. File lock can be specified if:

  - The logical file is opened within the transaction range under the condition that lock control for individual transaction branches is specified.

  In the following condition, file lock cannot be specified. Specify block lock:

  - The logical file is opened outside the transaction range.
  - Lock control for individual global transaction units is specified.

  If a logical file is closed and is again opened in the same transaction branch, the status before the logical file is closed is inherited.

- When accessing an unrecoverable DAM file:

  Since this type of file does not synchronize with transaction processing, there is no restriction for lock.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMSVC'` for the interface code used with the DAM file.

- *data-name-C*

  Specify a logical file name with up to 8 characters. If the specified logical file name comprises less than 8 characters, pad the remaining portion with space.

- *data-name-E*

  Specify `VALUE 'OPEN'` for the request code indicating that the logical file is open.

- *data-name-F*

  Specify files or in blocks-based lock.

  `VALUE 'B'`: Blocks-based lock

  `VALUE 'F'`: Files-based lock

  File lock can be specified if:

  - The recoverable DAM file is opened within the transaction range.
  - The unrecoverable DAM file is opened.

- *data-name-G*

  Specify whether the program is to wait for the resource to be released from lock if a lock error occurs. When `VALUE 'F'` is specified for *data-name-F*, the action depends on whether the DAM file to be used is recoverable or not.

  `VALUE 'W'`: The program waits for the resource to be released from lock.

  `VALUE 'N'`: The program does not wait for the resource to be released from lock, and returns with an error.

  If both values are omitted, `VALUE 'N'` is assumed.

  The values which can be specified for *data-name-F* and *data-name-G* depend on whether the DAM file to be used is recoverable or not. The following describes the values specified for recoverable and unrecoverable DAM files:

  - When accessing a recoverable DAM file:

    The value specified for *data-name-G* can be invalid if a lock error occurs not in `CBLDCDAM('OPEN')`, but in `CBLDCDAM('READ')` and `CBLDCDAM('WRIT')` when accessing a logical file which has already been opened. If a lock error occurs in `CBLDCDAM('OPEN')`, an error is unconditionally returned with the status code `01602`.

The table below shows the correspondence between the values specified for *data-name-F* and *data-name-G* and the specified type of lock when accessing a recoverable DAM file.

data-name-F	data-name-G#	Specified lock
`'F'`	Δ	Files-based lock
`'B'`	`'W'`	Blocks-based lock, and waiting for release from lock if a lock error occurs
	`'N'`	Blocks-based lock, and error return if a lock error occurs

Legend:

Δ: Specify a space (`' '`).

#: If `'B'` is specified for *data-name-F* and the value for *data-name-G* is omitted, `'N'` is assumed.

- When accessing an unrecoverable DAM file:

The value specified for *data-name-G* from lock can be invalid if a lock error occurs. If a lock error occurs in `CBLDCDAM('OPEN')`, `CBLDCDAM('READ')` and `CBLDCDAM('WRIT')`, whether to wait for release from lock or not is determined according to the value of *data-name-G*. If a lock error occurs when *data-name-G* is set to `'N'` or omitted, an error is returned with the status code `01602`.

The table below shows the correspondence between the values specified for *data-name-F* and *data-name-G* and the specified type of lock when accessing an unrecoverable DAM file.

data-name-F	data-name-G#	Specified lock
`'F'`	`'W'`	Files-based lock, and waiting for release from lock if a lock error occurs
	`'N'`	Files-based lock, and error return if a lock error occurs
`'B'`	`'W'`	Blocks-based lock, and waiting for release from lock if a lock error occurs
	`'N'`	Blocks-based lock, and error return if a lock error occurs

#: The default is `'N'`.

Regardless of whether the file is recoverable or not, no lock error occurs in `CBLDCDAM('READ')` and `CBLDCDAM('WRIT')` when files-based lock is specified for *data-name-F*. This is because the file is entirely locked. Therefore, whether to wait for release from lock cannot be specified. The value specified for *data-name-G* in `CBLDCDAM('READ')` and

`CBLDCDAM('WRIT')` is ignored.

- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-H*

  The file descriptor is returned.

Status codes

Status code	Explanation
`00000`	The file descriptor was specified for *data-name-H* normally.
`01600`	`CBLDCRPC('OPEN    ')` was not called.
	`'N'` is specified for `atomic_update` in the user service definition. (This error is returned only when accessing a recoverable DAM file.)
	The `CBLDCDAM('STRT')` function is not called when `N` is specified for the `atomic_update` operand in the user service definition. (This value is returned only when an unrecoverable DAM file is accessed.)
	The UAP is incorrectly linked as follows: • The library (`-ltdam`) to be used for access to a TAM file using the DAM service API is linked incorrectly. • The definition of the resource manager for transaction control object files is incorrect.
	DAM file lock is specified from outside the transaction range. (This error is returned only when accessing a recoverable DAM file.)
	Files-based lock is specified for the DAM file under lock control for individual global transaction units. (This error is returned only when accessing a recoverable DAM file.)
`01601`	The logical file name specified for *data-name-C* has not been defined.
`01602`	A lock error occurred.
`01605`	The `CBLDCDAM('STRT')` function is not called when `Y` is specified for the `atomic_update` operand in the user service definition. (This value is returned only when an unrecoverable DAM file is accessed.)
`01607`	The memory became insufficient.
`01608`	The logical file specified for *data-name-C* is open.
`01610`	The value specified as the logical file name is invalid.

Status code	Explanation
`01611`	The value specified for *data-name-F* or *data-name-G* is invalid.
`01621`	The file specified for *data-name-C* is in logical shutdown state.
`01622`	The file specified for *data-name-C* is in shutdown state due to an error.
`01627`	The number of open character special files exceeds the specified limit.
`01628`	The access permission for character special files has not been granted.
`01629`	A transaction service error occurred. (This error is returned only when accessing an unrecoverable DAM file.)
`01642`	A deadlock occurred. (This error is returned only when accessing an unrecoverable DAM file.)
`01643`	The resource could not be acquired because a timeout occurred since the wait time specified in the lock service definition was exceeded. (This error is returned only when accessing an unrecoverable DAM file.)
`01645`	The number of lock requests exceeds the specified maximum number of concurrent lock requests.
`01646`	The DAM file to be opened is protected by the security facility. No ACL exists for the file.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-E*) is invalid.

CBLDCDAM('READ') - Input a logical file block

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCDAM'  USING  unique-name-1  unique-name-2
                         unique-name-n
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMSVC'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(8).
    02  data-name-D    PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  data-name-H    PIC S9(9) COMP.
    02  FILLER         PIC X(28).
01  unique-name-2.
    02  data-name-F    PIC X(4) VALUE 'READ'.
    02  data-name-G    PIC X(1).
    02  data-name-L    PIC X(1).
    02  data-name-M    PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
    02  unique-name-3.
        03  data-name-I    PIC S9(9) COMP.
        03  data-name-J    PIC S9(9) COMP.
    02  unique-name-4.
        03  data-name-I    PIC S9(9) COMP.
        03  data-name-J    PIC S9(9) COMP.
          :
          :
    02  unique-name-m.
        03  data-name-I    PIC S9(9) COMP.
        03  data-name-J    PIC S9(9) COMP.
01  unique-name-n.
    02  data-name-K    PIC X(n).
```

Description

`CBLDCDAM('READ')` inputs a block (which is in the specified range) for reference or update processing from the specified logical file.

- When accessing a recoverable DAM file:

  Blocks-based lock is enabled as specified when the logical file was opened. The block of a logical file can be input from a process out of the transaction range. In this case, however, the statement can be used only for reference and lock cannot be specified. When an request is made to input multiple blocks by specifying the block numbers, an error returned if even one of the blocks causes an error. In this case, the blocks are not input to the input buffer. All the blocks for which an input request was made are released from lock at this time.

Lock which is enabled for a block input for reference processing is released in the following case:

After the block is input for reference processing, an input request for update processing is made for the same block. Then, an input error occurs during the update processing.

Even if block update during a transaction is specified (`dam_update_block_over=flush` in the DAM service definition), an error is returned with the status code `01613` in the following case:

- DAM file blocks are not updated (REWRITE) in one transaction branch. The block input for update processing (READ) is called. Eventually, the number of blocks exceeds the maximum number of blocks collectively updated (the value specified for `dam_update_block` of the DAM service definition).

Call `CBLDCDAM('READ')` within the transaction range to input a recoverable DAM file block.

- When accessing an unrecoverable DAM file:

  When an unrecoverable DAM file block is input, there is no restriction for conditions to call `CBLDCDAM('READ')`.

  If unrecoverable DAM file blocks exceeding the value specified for `dam_update_block` in the DAM service definition are input for update processing, an error is returned with the status code `01648`.

When inputting a logical file block, specify the file descriptor returned from `CBLDCDAM('OPEN')`.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'DCDAMSVC'` for the interface code used with the DAM file.

■ *data-name-C*

Specify a logical file name with up to 8 characters. If the specified logical file name comprises less than 8 characters, pad the remaining portion with space.

■ *data-name-D*

Specify the number of blocks from *unique-name-3* to *unique-name-m* (number of sets of *data-name-I* and *data-name-J*).

- *data-name-E*

  Specify the length of the input buffer. The buffer length must be equal to or greater than (number of blocks to be input `x` block length to be input).

  'Number of blocks to be input' is the sum of blocks specified with the number of blocks from *unique-name-3* to *unique-name-m* (number of sets of *data-name-I* and *data-name-J*).

- *data-name-H*

  Specify the file descriptor returned when the logical file was opened.

- *data-name-F*

  Specify `VALUE 'READ'` for the request code indicating the input of a logical file block.

- *data-name-G*

  Specify an input request type with `VALUE 'R'` or `VALUE 'M'`.

  `VALUE 'R'`: Input request for reference

  `VALUE 'M'`: Input request for update

- *data-name-L*

  Specify whether to apply lock if the input request is for reference. If `'E'` is specified, lock will remain until processing reaches the synchronization point.

  If a block is input from an unlocked logical file, the block could be updated by another transaction during the input processing. In this case, the details input to the block depend on the update processing status of the other transaction. Therefore, to reference the latest block contents, be sure to specify `'E'`.

  If the input request is for update, set a blank here; any other value will be ignored.

  `VALUE 'E'`: Lock is enabled.

  `VALUE 'N'`: Lock is not enabled.

  Lock cannot be applied when accessing a recoverable DAM file outside the transaction for reference processing.

  If `VALUE 'M'` is specified for *data-name-G*, specify space (`' '`) for *data-name-L*.

- *data-name-M*

  Specify whether the program is to wait for the resource to be released from lock if a lock error occurs. If `VALUE 'N'` is specified for *data-name-L*, specify space (`' '`) for *data-name-M*.

  `VALUE 'W'`: The program waits for the resource to be released from lock.

  `VALUE 'N'`: The program does not wait for the resource to be released from lock, and returns with an error.

`VALUE SPACE` : Processing is done according to the value specified for *data-name-G* of `CBLDCDAM('OPEN')`.

If `VALUE SPACE` is specified or no value is specified, the subsequent processing is as follows:

- If `VALUE 'W'` is specified for *data-name-G* of `CBLDCDAM('OPEN')`, the program waits for the resource to be released from lock.
- If `VALUE 'N'` is specified for *data-name-G* of `CBLDCDAM('OPEN')` or if no value is specified, the program does not wait for the resource to be released from lock, and returns with an error.

However, if files-based lock is specified as the lock type in `CBLDCDAM('OPEN')` with the file descriptor specified for *data-name-H*, the value specified for *data-name-M* is meaningless.

The table below shows the correspondence between the values specified for *data names G*, *L*, and *M* and the specified type of lock.

data-name-G	data-name-L[#1]	data-name-M[#2]	Specified lock type
`'R'`	`'E'`	`'W'`	Input for reference, lock used, and waiting for release from lock if a lock error occurs
		`'N'`	Input for reference, lock used, and error return if a lock error occurs
	`'N'`	Δ	Input for reference, lock not used[#3]
`'M'`	Δ	`'W'`	Input for update, and waiting for release from lock if a lock error occurs
		`'N'`	Input for update, and error return if a lock error occurs

Legend:

Δ: Specify space (`' '`).

#1: If `'R'` is specified for *data-name-G* and the value for the *data-name-L* is omitted, `'N'` is assumed.

#2: If space (`' '`) is specified for *data-name-M* in the following situation, the value specified for opening the logical file (waiting for release from lock or error return) is assumed:

If `'R'` is specified for *data-name-G* and `'E'` is specified for *data-name-L*

If `'M'` is specified for *data-name-G*

#3: When accessing a recoverable DAM file, the statement for input a logical file block can be called from a process out of the transaction range only if `'R'` is specified for *data-name-G* and `'N'` is specified for *data-name-L*. If the statement for input a logical file block is called with the other values specified, it returns with an error, giving the status code `01600`.

- *data-name-I*

  Specify the first relative block number of the block to be accessed.

- *data-name-J*

  Specify the last relative block number of the block to be accessed. If 0 is specified, only the relative block number specified for *data-name-I* is input.

- *data-name-K*

  Specify an input data area.

- *data-name-Z*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	All blocks were input normally.
`01600`	`CBLDCRPC('OPEN    ')` was not called.
	A block is input for update processing or a locked block is input for reference processing outside the transaction range. (This error is returned only when accessing a recoverable DAM file.)
	`'N'` is specified for `atomic_update` in the user service definition. (This error is returned only when accessing a recoverable DAM file.)
	`CBLDCDAM('STRT')` was not called. (This error is returned only when accessing an unrecoverable DAM file.)
	The UAP is incorrectly linked as follows: • The library (`-1tdam`) to be used for access to a TAM file using the DAM service API is linked incorrectly. • The definition of the resource manager for transaction control object files is incorrect.
`01602`	A lock error occurred.

Status code	Explanation
01603	The file descriptor specified for *data-name-H* is not the one which was acquired by opening the file normally.
	The DAM file is not open.
01604	The specified input buffer is too small to contain all blocks.
01606	The relative block number is invalid.
01607	The memory became insufficient.
01609	The value specified for *data-name-D* is smaller than 1.
01611	The value specified for *data-name-G* or *L* is invalid.
01613	The number of block updates exceeded the maximum number of blocks that can be updated during one transaction according to the DAM service definition.
01618	The version of the DAM library linked to the UAP does not allow the UAP to operate with the current DAM service.
01620	An input error occurred.
01621	The specified file is in logical shutdown state.
01622	The specified file is in shutdown state due to an error.
01628	The DAM file to be accessed is protected by the security facility. The UAP attempting to input a logical file block has no access permission.
01629	A transaction service error occurred. (This error is returned only when accessing a recoverable DAM file.)
01642	A deadlock occurred.
01643	The resource could not be acquired because a timeout occurred (the wait time specified in the lock service definition was exceeded).
01645	The number of lock requests exceeds the specified maximum number of concurrent lock requests.
01648	The number of blocks exceeded the number of blocks accessible for an unrecoverable DAM file. (This error is returned only when accessing an unrecoverable DAM file.)
01690	The interface code (*data-name-A*) is invalid.
01691	The request code (*data-name-F*) is invalid.

CBLDCDAM('REWT') - Update a logical file block

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCDAM'  USING  unique-name-1  unique-name-2
                         unique-name-n
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMSVC'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(8).
    02  data-name-D    PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  data-name-H    PIC S9(9) COMP.
    02  FILLER         PIC X(28).
01  unique-name-2.
    02  data-name-F    PIC X(4) VALUE 'REWT'.
    02  data-name-G    PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
    02  unique-name-3.
        03  data-name-I    PIC S9(9) COMP.
        03  data-name-J    PIC S9(9) COMP.
    02  unique-name-4.
        03  data-name-I    PIC S9(9) COMP.
        03  data-name-J    PIC S9(9) COMP.
            :
            :
    02  unique-name-m.
        03  data-name-I    PIC S9(9) COMP.
        03  data-name-J    PIC S9(9) COMP.
01  unique-name-n.
    02  data-name-K    PIC X(n).
```

Description

`CBLDCDAM('REWT')` outputs a block, input from the logical file for update processing. `CBLDCDAM('REWT')` also cancels an update request.

The timing of updating blocks is shown below:

- When accessing a recoverable DAM file:

  The updated data is stored in the part of shared memory that is allocated for DAM service, and the actual file is updated when the transaction is committed. If deferred update is specified for the DAM file, the file contents are updated asynchronously with the commitment of the transaction.

- When accessing an unrecoverable DAM file:

  The contents of the DAM file are updated when `CBLDCDAM('REWT')` returns.

When multiple blocks are specified at a time and if even one of the specified blocks causes an error, processing is stopped and an error is returned. Update processing is not done in this case.

When updating a logical file block, specify the logical file name and the file descriptor returned from `CBLDCDAM('OPEN')`.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'DCDAMSVC'` for the interface code used with the DAM file.

■ *data-name-C*

Specify a logical file name with up to 8 characters. If the specified logical file name comprises less than 8 characters, pad the remaining portion with space.

■ *data-name-D*

Specify the number of blocks from *unique-name-3* to *unique-name-m* (number of sets of *data-name-I* and *data-name-J*).

■ *data-name-E*

Specify the length of the update data. The update data length must be equal to or greater than (block length to be updated x number of blocks to be updated).

'Number of blocks to be updated' is the sum of blocks specified with the number of blocks from *unique-name-3* to *unique-name-m* (number of sets of *data-name-I* and *data-name-J*).

■ *data-name-H*

Specify the file descriptor returned when the logical file was opened.

■ *data-name-F*

Specify `VALUE 'REWT'` for the request code indicating the update of a logical file block.

■ *data-name-G*

Specify an update request type with `VALUE 'U'` or `VALUE 'C'`.

`VALUE 'U'`: Update request

`VALUE 'C'`: Update request cancel

- *data-name-I*

  Specify the first relative block number of the block to be accessed.

- *data-name-J*

  Specify the last relative block number of the block to be accessed. If 0 is specified, only the relative block number specified for *data-name-I* is input.

- *data-name-K*

  Specify an update data area.

- *data-name-Z*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	All blocks were updated normally.
`01600`	`CBLDCRPC('OPEN    ')` was not called.
	`CBLDCDAM('REWT')` was called outside the transaction range. (This error is returned only when accessing a recoverable DAM file.)
	`'N'` is specified for `atomic_update` in the user service definition. (This error is returned only when accessing a recoverable DAM file.)
	`CBLDCDAM('STRT')` was not called. (This error is returned only when accessing an unrecoverable DAM file.)
	The UAP is incorrectly linked as follows: • The library (`-ltdam`) to be used for access to a TAM file using the DAM service API is linked incorrectly. • The definition of the resource manager for transaction control object files is incorrect.
`01603`	The file descriptor specified for *data-name-H* is not the one which was acquired by opening the file normally.
	The DAM file is not open.
`01604`	The update data length (block length to be updated `x` number of blocks to be updated) is too short.

Status code	Explanation
`01605`	The logical file block was not input for update processing.
`01606`	The relative block number is invalid.
`01607`	The memory became insufficient.
`01609`	The value specified for *data-name-D* is smaller than 1.
`01611`	The value specified for *data-name-G* is invalid.
`01613`	The number of block updates exceeded the maximum number of blocks that can be updated during one transaction according to the DAM service definition. (This error is returned only when accessing a recoverable DAM file.)
`01620`	An output error occurred. (This error is returned only when accessing an unrecoverable DAM file.)
`01621`	The specified file is in logical shutdown state.
`01622`	The specified file is in shutdown state due to an error.
`01629`	The transaction service encountered an error. (This error is returned only when accessing a recoverable DAM file.)
`01641`	The update data length (block length to be updated `x` number of blocks to be updated) is too long.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-F*) is invalid.

CBLDCDAM('RLES') - Release a logical file from the shutdown state

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDAM'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(8) VALUE 'DCDAMSVC'.
    02  data-name-B     PIC X(5).
    02  FILLER          PIC X(3).
    02  data-name-C     PIC X(8).
    02  FILLER          PIC S9(9) COMP.
    02  FILLER          PIC S9(9) COMP.
    02  FILLER          PIC X(32).
01  unique-name-2.
    02  data-name-E     PIC X(4) VALUE 'RLES'.
    02  data-name-F     PIC X(1).
    02  FILLER          PIC X(1).
    02  FILLER          PIC X(1).
    02  FILLER          PIC X(1).
    02  data-name-Z     PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCDAM('RLES')` releases a logical file which has been held by `CBLDCDAM('HOLD')`. It also releases a logical file which has been shut down due to an error.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMSVC'` for the interface code used with the DAM file.

- *data-name-C*

  Specify a logical file name with up to 8 characters. If the specified logical file name comprises less than 8 characters, pad the remaining portion with space.

- *data-name-E*

  Specify `VALUE 'RLES'` for the request code indicating release of a logical file from the shutdown state.

- *data-name-F*

  Specify a shutdown release type with `VALUE 'L'` or `VALUE 'O'`.

`VALUE 'L'`: The file is released from the logical shutdown state.

`VALUE 'O'`: The file is released from the shutdown state due to an error.

- *data-name-Z*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	The specified logical file was released from the shutdown state normally.
`01600`	`CBLDCRPC('OPEN    ')` was not called.
	`'N'` is specified for `atomic_update` in the user service definition. (This error is returned only when accessing a recoverable DAM file.)
	`CBLDCRPC('STRT')` was not called. (This error is returned only when accessing an unrecoverable DAM file.)
	The UAP is incorrectly linked as follows: • The library (`-1tdam`) to be used for access to a TAM file using the DAM service API is linked incorrectly. • The definition of the resource manager for transaction control object files is incorrect.
`01601`	The logical file specified for *data-name-C* has not been defined.
`01607`	The memory became insufficient.
`01610`	The value specified as the logical file name is invalid.
`01611`	The value specified for *data-name-F* is invalid.
`01618`	The OpenTP1 file system version does not much the OpenTP1 system version.
`01619`	The physical file corresponding to the logical file specified for *data-name-C* does not exist.
`01620`	An input error occurred.
`01623`	The specified logical file is not in logical shutdown state.
`01624`	The specified logical file is not in shutdown state due to an error.
`01627`	The number of open character special files exceeds the specified maximum number.

Status code	Explanation
`01628`	The access permission for character special files has not been granted.
	The DAM file to be accessed is protected by the security facility. The UAP attempting to release the logical file from the shutdown state has no access permission.
`01632`	The physical file is not a character special file, or the device corresponding to the specified special file does not exist.
`01633`	The physical file corresponding to the logical file specified for *data-name-C* has not been initialized as an OpenTP1 file system.
`01638`	The access permission for the physical file that corresponds to the logical file specified for *data-name-C* has not been granted.
`01646`	The DAM file to be released from the shutdown state is protected by the security facility. No ACL exists for the file.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-E*) is invalid.

CBLDCDAM('STAT') - Reference the status of a logical file

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCDAM'  USING  unique-name-1   unique-name-2
                         unique-name-3
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(8) VALUE 'DCDAMSVC'.
    02  data-name-B     PIC X(5).
    02  FILLER          PIC X(3).
    02  data-name-C     PIC X(8) COMP.
    02  data-name-E     PIC S9(9) COMP VALUE ZERO.
    02  data-name-F     PIC S9(9) COMP.
    02  data-name-G     PIC S9(9) COMP.
    02  FILLER          PIC S9(9) COMP.
    02  data-name-H     PIC X(1).
    02  data-name-I     PIC X(1).
    02  data-name-J     PIC X(1).
    02  FILLER          PIC X(1).
    02  FILLER          PIC X(20).
01  unique-name-2.
    02  data-name-D     PIC X(4) VALUE 'STAT'.
    02  FILLER          PIC X(1).
    02  FILLER          PIC X(1).
    02  FILLER          PIC X(1).
    02  FILLER          PIC X(1).
    02  data-name-Z     PIC S9(9) COMP VALUE ZERO.
01  unique-name-3.
    02  data-name-K     PIC X(64).
```

Description

`CBLDCDAM('STAT')` returns the current status of a logical file. The contents to be returned are shown below:

- Number of blocks of the logical file
- Block length of the logical file
- Physical file name corresponding to the logical file
- Current status of the logical file (whether in the shutdown state or not)
- Attribute of the logical file specified in the DAM service definition
- Security attribute of the logical file specified in the DAM service definition

The status of a logical file can be referenced whether the logical file is opened or not.

When referencing the status of a logical file, specify the logical file name.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMSVC'` for the interface code used with the DAM file.

- *data-name-C*

  Specify a logical file name with up to 8 characters. If the specified logical file name comprises less than 8 characters, pad the remaining portion with space.

- *data-name-E*

  Specify `0`.

- *data-name-D*

  Specify `VALUE 'STAT'` for the request code indicating the reference of the status of a logical file.

- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  The status code of 5 digit is returned.

- *data-name-F*

  Block length of a logical file is returned.

- *data-name-G*

  The number of blocks of a logical file is returned.

- *data-name-H*

  The current status of a logical file is set to one of the following values:

  `VALUE 'N'` ... The logical file is accessible.

  `VALUE 'L'` ... The logical file is in logical shutdown state.

  `VALUE 'O'` ... The logical file is in shutdown state due to an error.

  `VALUE 'H'` ... The logical file is requested to be shut down.

- *data-name-I*

  The attribute of a logical file specified in the DAM service definition is set to one of the following values:

`VALUE 'Q'` ... The DAM file is not a target for deferred update processing.

`VALUE 'D'` ... The DAM file is a target for deferred update processing.

`VALUE 'N'` ... The DAM file is unrecoverable.

`VALUE 'C'` ... Unrecoverable DAM file specified by a cache-less access.

- *data-name-J*

  The security attribute of a logical file specified in the DAM service definition is set to one of the following values:

  `VALUE 'N'` ... Security is not required.

  `VALUE 'S'` ... Security is required.

- *data-name-K*

  A physical file name corresponding to the logical file is set here.

Status codes

Status code	Explanation
`00000`	The status of the logical name was set to *data-name-H* normally.
`01600`	`CBLDCRPC('OPEN    ')` was not called.
	`'N'` is specified for `atomic_update` in the user service definition. (This error is returned only when accessing a recoverable DAM file.)
	`CBLDCDAM('STRT')` was not called. (This error is returned only when accessing an unrecoverable DAM file.)
`01601`	The logical file name specified for *data-name-C* has not been defined.
`01607`	The memory became insufficient.
`01610`	The logical file name specified for *data-name-C* is invalid.
`01611`	The value specified for *data-name-E* is invalid.
`01612`	The value set to a data name to which a value is returned from OpenTP1 is invalid or not a space.
`01618`	The version of the DAM library linked to the UAP does not allow the UAP to operate with the current DAM service.
`01628`	The DAM file of which the status was attempted to be referenced is protected by the security facility. The UAP that called `CBLDCDAM('STAT')` has no access permission.
`01646`	The DAM file of which the status was attempted to be referenced is protected by the security facility. No ACL exists for the file.

Status code	Explanation
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-D*) is invalid.

Note

When referencing the status of a DAM file, the DAM service applies lock to get information. Therefore, frequent calls of `CBLDCDAM('STAT')` may decrease throughput due to the waiting time for release from lock. Hold the frequency of referencing the status of a DAM file in online mode to a minimum.

CBLDCDAM('STRT') - Start using an unrecoverable DAM file

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDAM'  USING  unique-name-1  unique-name-2
                         unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMSVC'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  FILLER         PIC X(8).
    02  FILLER         PIC S9(9) COMP.
    02  FILLER         PIC S9(9) COMP.
    02  FILLER         PIC S9(9) COMP.
    02  FILLER         PIC X(28).
01  unique-name-2.
    02  data-name-C    PIC X(4) VALUE 'STRT'.
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCDAM('STRT')` declares that unrecoverable DAM files are used.

Whenever an unrecoverable DAM file is used, call `CBLDCDAM('STRT')` for each UAP process before opening a logical file.

When `CBLDCDAM('STRT')` returns normally, the environment for accessing an unrecoverable DAM file is prepared.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMSVC'` for the interface code used with the DAM file.

- *data-name-C*

  Specify `VALUE 'STRT'` for the request code indicating that using the unrecoverable DAM file is started.

- *data-name-Z*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  The status code of 5 digit is returned.

Status codes

Status code	Explanation
`00000`	Normal termination. Unrecoverable DAM files can now be used.
`01600`	`CBLDCRPC('OPEN    ')` was not called.
`01607`	The memory became insufficient.
`01618`	The version of the DAM library linked to the UAP does not allow the UAP to operate with the current DAM service.
`01647`	`CBLDCDAM('STRT')` has already been called.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-C*) is invalid.

CBLDCDAM('WRIT') - Output a logical file block

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCDAM'  USING  unique-name-1   unique-name-2
                         unique-name-n
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMSVC'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(8).
    02  data-name-D    PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  data-name-H    PIC S9(9) COMP.
    02  FILLER         PIC X(28).
01  unique-name-2.
    02  data-name-F    PIC X(4) VALUE 'WRIT'.
    02  data-name-G    PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
    02  unique-name-3.
        03  data-name-I    PIC S9(9) COMP.
        03  data-name-J    PIC S9(9) COMP.
    02  unique-name-4.
        03  data-name-I    PIC S9(9) COMP.
        03  data-name-J    PIC S9(9) COMP.
          :
          :
    02  unique-name-m.
        03  data-name-I    PIC S9(9) COMP.
        03  data-name-J    PIC S9(9) COMP.
01  unique-name-n.
    02  data-name-K    PIC X(n).
```

Description

`CBLDCDAM('WRIT')` outputs a specified block. The timing of outputting blocks is shown below:

- When accessing a recoverable DAM file:

  The updated data is stored in the part of shared memory that is allocated for DAM service, and the actual file is updated when the transaction is committed. If deferred output is specified for the DAM file, the file contents are output asynchronously with the commitment of the transaction.

- When accessing an unrecoverable DAM file:

  The contents of the DAM file are output when `CBLDCDAM('WRIT')` returns.

When a request is made to output multiple blocks at a time and if even one of the specified blocks causes an error, processing is stopped and an error is returned. The blocks are not output in this case.

Lock which is enabled for a block input for reference processing is released in the following case:

After the block is input for reference processing, an input request for update processing is made for the same block. Then, an input error occurs during the update processing.

When outputting a logical file block, specify the logical file name and the file descriptor returned from `CBLDCDAM('OPEN')`.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'DCDAMSVC'` for the interface code used with the DAM file.

■ *data-name-C*

Specify a logical file name with up to 8 characters. If the specified logical file name comprises less than 8 characters, pad the remaining portion with space.

■ *data-name-D*

Specify the number of blocks from *unique-name-3* to *unique-name-m* (number of sets of *data-name-I* and *data-name-J*).

■ *data-name-E*

Specify the length of output data. The output data length must be (block length to be output `x` number of blocks to be output).

'Number of blocks to be output' is the sum of blocks specified with the number of blocks from *unique-name-3* to *unique-name-m* (number of sets of *data-name-I* and *data-name-J*).

■ *data-name-H*

Specify the file descriptor returned when the logical file was opened.

■ *data-name-F*

Specify `VALUE 'WRIT'` for the request code indicating the output of a logical file block.

- *data-name-G*

  Specify whether the program is to wait for the resource to be released from lock if a lock error occurs.

  `VALUE 'W'`: The program waits for the resource to be released from lock.

  `VALUE 'N'`: The program does not wait for the resource to be released from lock, and returns with an error.

  `VALUE SPACE` : Processing is done according to the value specified for *data-name-G* of `CBLDCDAM('OPEN')`.

  If `VALUE SPACE` is specified or no value is specified, the subsequent processing is as follows:

  - If `VALUE 'W'` is specified for *data-name-G* of `CBLDCDAM('OPEN')`, the program waits for the resource to be released from lock.
  - If `VALUE 'N'` is specified for *data-name-G* of `CBLDCDAM('OPEN')` or if no value is specified, the program does not wait for the resource to be released from lock, and returns with an error.

  However, if files-based lock is specified as the lock type in `CBLDCDAM('OPEN')` with the file descriptor specified for *data-name-H*, the value specified for *data-name-G* is meaningless.

- *data-name-Z*

  Specify `0`.

- *data-name-I*

  Specify the first relative block number of the block to be accessed.

- *data-name-J*

  Specify the last relative block number of the block to be accessed. If 0 is specified, only the relative block number specified for *data-name-I* is output.

- *data-name-K*

  Specify an output data area (buffer).

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	All blocks were output normally.

Status code	Explanation
`01600`	`CBLDCRPC('OPEN    ')` was not called.
	`CBLDCDAM('WRIT')` was called outside the transaction range. (This error is returned only when accessing a recoverable DAM file.)
	`'N'` is specified for `atomic_update` in the user service definition. (This error is returned only when accessing a recoverable DAM file.)
	`CBLDCDAM('STRT')` was not called. (This error is returned only when accessing an unrecoverable DAM file.)
	The UAP is incorrectly linked as follows: • The library (`-ltdam`) to be used for access to a TAM file using the DAM service API is linked incorrectly. • The definition of the resource manager for transaction control object files is incorrect.
`01602`	A lock error occurred.
`01603`	The file descriptor specified for *data-name-H* is not the one which was acquired by opening the file normally.
	The DAM file is not open.
`01604`	The output data length (block length to be output `x` number of blocks to be output) is too short.
`01605`	The sequence for accessing the DAM file is invalid.
`01606`	The relative block number is invalid.
`01607`	The memory became insufficient.
`01609`	The value specified for *data-name-D* is smaller than 1.
`01611`	The value specified for *data-name-G* is invalid.
`01613`	The number of block updates exceeded the maximum number of blocks that can be updated during one transaction according to the DAM service definition. (This error is returned only when accessing a recoverable DAM file.)
`01620`	An output error occurred. (This error is returned only when accessing an unrecoverable DAM file.)
`01621`	The specified file is in logical shutdown state.
`01622`	The specified file is in shutdown state due to an error.
`01628`	The DAM file to be accessed is protected by the security facility. The UAP attempting to input a logical file block has no access permission.
`01629`	A transaction service error occurred. (This error is returned only when accessing a recoverable DAM file.)

Status code	Explanation
`01641`	The output data length (block length to be output `x` number of blocks to be output) is too long.
`01642`	A deadlock occurred.
`01643`	The resource could not be acquired because a timeout occurred (the wait time specified in the lock service definition was exceeded).
`01645`	The number of lock requests exceeds the specified maximum number of concurrent lock requests.
`01648`	The number of blocks exceeded the maximum number of accessible blocks. (This error is returned only when accessing an unrecoverable DAM file.)
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-F*) is invalid.

Note

If the status code `01613` or `01648` is returned, take the following actions:

- Make the number of blocks to be output less than or equal to the maximum number of blocks to be updated.
- Update the blocks which have not been updated by `CBLDCDAM('REWT')` before calling `CBLDCDAM('WRIT')`, if any.

CBLDCDMB('BSEK') - Seek a physical file block

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDMB'  USING  unique-name-1  unique-name-2
                         unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMINT'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(1).
    02  data-name-C    PIC X(63).
    02  FILLER         PIC X(3).
    02  data-name-H    PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  FILLER         PIC S9(9) COMP.
    02  data-name-I    PIC S9(9) COMP.
01  unique-name-2.
    02  data-name-D    PIC X(4) VALUE 'BSEK'.
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCDMB('BSEK')` specifies a relative block number of the physical file to locate the corresponding block. Call `CBLDCDMB('BSEK')` after opening a physical file with a re-creation output request.

When the relative block number exists in the file, the relative block number is returned.

When seeking a physical file block, specify the file descriptor returned from `CBLDCDMB('OPEN')`.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMINT'` for the interface code used with the DAM file.

- *data-name-C*

  Specify the physical file name which comprises 63 characters (special file name + 14 characters) or less. If the specified physical file name has less than 63 characters, pad the remaining portion with space.

- *data-name-E*

  Specify the file descriptor.
- *data-name-I*

  Specify the relative block number of the block to be sought.
- *data-name-D*

  Specify `VALUE 'BSEK'` for the request code indicating that a physical file is sought.
- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  The status code of 5 digit is returned.
- *data-name-H*

  The relative block number of the sought block is returned.

Status codes

Status code	Explanation
`00000`	The relative block number was set to *data-name-H* normally.
`01603`	The file descriptor specified for *data-name-E* is not the one which was acquired by opening the DAM file normally.
	The DAM file is not open.
`01605`	The sequence of accessing the DAM file is invalid.
`01606`	The relative block number is invalid.
`01620`	An output error occurred.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-D*) is invalid.

CBLDCDMB('CLOS') - Close a physical file

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCDMB'  USING  unique-name-1  unique-name-2
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMINT'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(1).
    02  data-name-C    PIC X(63).
    02  FILLER         PIC X(3).
    02  FILLER         PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  FILLER         PIC S9(9) COMP.
    02  FILLER         PIC X(4).
01  unique-name-2.
    02  data-name-D    PIC X(4) VALUE 'CLOS'.
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCDMB('CLOS')` closes a physical file created in the OpenTP1 file system.

If a file is not filled with data, the remaining part up to the end of the file is padded with space while the remaining part is padded with blocks of null characters only in the following cases:

- A creation output request has been specified for *data-name-I* of `CBLDCDMB('OPEN')`.
- `CBLDCDMB('CRAT')` has been called.

When closing a physical file, specify the physical file name and the file descriptor returned from `CBLDCDMB('OPEN')` or `CBLDCDMB('CRAT')`.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'DCDAMINT'` for the interface code used with the DAM file.

- *data-name-C*

  Specify a physical file name with a path name which comprises 63 characters (special file name + 14 characters) or less. If the physical file name comprises less than 63 characters, pad the remaining portion with space.

- *data-name-E*

  Specify the file descriptor.

- *data-name-D*

  Specify `VALUE 'CLOS'` for the request code indicating that the physical file is closed.

- *data-name-Z*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	The file was closed normally.
`01603`	The file descriptor specified for *data-name-E* is not the one which was acquired by opening the file normally.
	The DAM file is not open.
`01620`	An output error occurred.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-D*) is invalid.

CBLDCDMB('CRAT') - Allocate a physical file

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDMB'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMINT'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(1).
    02  data-name-C    PIC X(63).
    02  FILLER         PIC X(3).
    02  data-name-D    PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  data-name-G    PIC S9(9) COMP.
    02  data-name-H    PIC S9(9) COMP.
01  unique-name-2.
    02  data-name-F    PIC X(4) VALUE 'CRAT'.
    02  data-name-I    PIC X(1).
    02  data-name-J    PIC X(1).
    02  data-name-K    PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCDMB('CRAT')` allocates a physical file to the OpenTP1 file system.

The size of a physical file is (block length + 8) x (number of blocks + 1).

There is no need to open the physical file after the physical file is allocated.

The following COBOL-UAP creation programs cannot be called after the physical file is allocated by `CBLDCDMB('CRAT')`:

- `CBLDCDMB ('GET ')`
- `CBLDCDMB ('BSEK')`
- `CBLDCDMB ('DGET')`
- `CBLDCDMB ('DPUT')`

The size of an output buffer is (block length + 8) x (number of blocks collectively processed).

The default is assumed if `'N'` is specified for access permissions (no access permission).

When allocating a physical file, specify the physical file name.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMINT'` for the interface code used with the DAM file.

- *data-name-C*

  Specify the name of the physical file to be created, with a pathname which comprises 63 characters (special file name + 14 characters) or less. If the physical file name comprises less than 63 characters, pad the remaining portion with space.

- *data-name-D*

  Specify the length of a block to be allocated in the OpenTP1 file system.

- *data-name-G*

  Specify the number of blocks to be allocated in the OpenTP1 file system.

- *data-name-H*

  Specify the number of blocks collectively processed which is used as an input/output unit.

- *data-name-F*

  Specify `VALUE 'CRAT'` for the request code indicating physical file allocation.

- *data-name-I*

  Specify the access permission of the file owner.

  `VALUE 'N'`: No access permission

  `VALUE 'R'`: Read permission only

  `VALUE 'W'`: Write permission only

  `VALUE 'B'`: Both read permission and write permission

- *data-name-J*

  Specify the access permission of the file owner group.

  `VALUE 'N'`: No access permission

  `VALUE 'R'`: Read permission only

  `VALUE 'W'`: Write permission only

  `VALUE 'B'`: Both read permission and write permission

- *data-name-K*

  Specify the access permission of those who are not the file owner.

  `VALUE 'N'`: No access permission

  `VALUE 'R'`: Read permission only

  `VALUE 'W'`: Write permission only

  `VALUE 'B'`: Both read permission and write permission

- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-E*

  The file descriptor is returned.

Status codes

Status code	Explanation
`00000`	The file descriptor was specified for *data-name-E* normally.
`01607`	The memory became insufficient.
`01608`	The physical file specified for *data-name-C* is open.
`01611`	The value specified for *data-name-I*, *J*, or *K* is invalid.
`01614`	The physical file name is invalid.
`01615`	The value specified for the number of blocks collectively processed is invalid.
`01617`	The same physical file name already exists.
`01618`	The OpenTP1 file system versions used for creation and allocation do not match each other.
`01620`	An input/output error occurred.
`01628`	The access permission for special files has not been granted.
	The DAM file to be allocated is protected by the security facility. The UAP that called `CBLDCDMB('CRAT')` has no access permission.
`01630`	The value specified for the block length is not suitable.

Status code	Explanation
`01631`	The value specified for the number of blocks is not suitable.
`01632`	The physical file is not a character special file, or the device corresponding to the special file does not exist.
`01633`	The specified OpenTP1 file has not been initialized as an OpenTP1 file system.
`01634`	When the OpenTP1 file was initialized as an OpenTP1 file system, an attempt was made to allocate more OpenTP1 files (physical files) than specified.
`01635`	The specified value exceeds the maximum number of files which can be opened in the process being executed.
`01636`	The physical file specified for *data-name-C* is being used in online mode, or it is being used by another process.
`01640`	The OpenTP1 file system does not have a free area large enough to allocate physical files.
`01646`	The DAM file to be allocated is protected by the security facility. No ACL exists for the file.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-F*) is invalid.

CBLDCDMB('DGET') - Input directly a physical file block

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDMB'  USING  unique-name-1  unique-name-2
                         unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMINT'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(1).
    02  data-name-C    PIC X(63).
    02  FILLER         PIC X(3).
    02  data-name-H    PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  data-name-G    PIC S9(9) COMP.
    02  data-name-I    PIC S9(9) COMP.
01  unique-name-2.
    02  data-name-D    PIC X(4) VALUE 'DGET'.
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
01  unique-name-3.
    02  data-name-F    PIC X(n).
```

Description

`CBLDCDMB('DGET')` inputs a block corresponding to the specified relative block number from a physical file. Call `CBLDCDMB('DGET')` after opening a physical file with a re-creation output request.

If the block length is less than the buffer length, the length of the input block is returned to *data-name-H*. If the block length is greater than the buffer length, `CBLDCDMB('DGET')` returns with an error.

When directly inputting a physical file block, specify the file descriptor returned from `CBLDCDMB('OPEN')`.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMINT'` for the interface code used with the DAM file.

- *data-name-C*

  Specify the physical file name which comprises 63 characters (special file name + 14 characters) or less. If the specified physical file name has less than 63 characters, pad the remaining portion with space.

- *data-name-E*

  Specify the file descriptor.

- *data-name-G*

  Specify the length of the input buffer.

- *data-name-I*

  Specify the relative block number of the block to be input.

- *data-name-D*

  Specify `VALUE 'DGET'` for the request code indicating the direct input of a physical file block.

- *data-name-F*

  Specify the input buffer.

- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  The status code of 5 digit is returned.

- *data-name-H*

  The length of the sought block is returned.

Status codes

Status code	Explanation
`00000`	The input block length was set to *data-name-H* normally.
`01603`	The file descriptor specified for *data-name-E* is not the one which was acquired by opening the DAM file normally.
	The DAM file is not open.
`01604`	The value specified for *data-name-G* is less than the block length.
`01605`	The sequence of accessing the DAM file is invalid.

Status code	Explanation
`01606`	The relative block number is invalid.
`01620`	An input error occurred.
`01628`	The DAM file to be accessed is protected by the security facility. The UAP attempting to input a physical file block has no access permission.
`01646`	The DAM file to be accessed is protected by the security facility. No ACL exists for the file.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-D*) is invalid.

CBLDCDMB('DPUT') - Output directly a physical file block

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDMB'  USING  unique-name-1   unique-name-2
                         unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMINT'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(1).
    02  data-name-C    PIC X(63).
    02  FILLER         PIC X(3).
    02  data-name-H    PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  data-name-G    PIC S9(9) COMP.
    02  data-name-I    PIC S9(9) COMP.
01  unique-name-2.
    02  data-name-D    PIC X(4) VALUE 'DPUT'.
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
01  unique-name-3.
    02  data-name-F    PIC X(n).
```

Description

`CBLDCDMB('DPUT')` outputs a block corresponding to the specified relative block number to a physical file. Call `CBLDCDMB('DPUT')` after opening a physical file with a re-creation output request.

If the length of the data to be output is less than the block length, some blocks are filled with the output data, and the remaining blocks are padded with null characters. Then, the block length is returned to *data-name-H*. If the length of the data to be output is greater than the block length, `CBLDCDMB('DPUT')` returns with an error.

When directly outputting a physical file block, specify the file descriptor returned from `CBLDCDMB('OPEN')`.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMINT'` for the interface code used with the DAM file.

- *data-name-C*

  Specify the physical file name which comprises 63 characters (special file name + 14 characters) or less. If the specified physical file name has less than 63 characters, pad the remaining portion with space.

- *data-name-E*

  Specify the file descriptor.

- *data-name-G*

  Specify the length of the data to be output.

- *data-name-I*

  Specify the relative block number of the block to be output.

- *data-name-D*

  Specify `VALUE 'DPUT'` for the request code indicating the direct output of a physical file block.

- *data-name-F*

  Specify the data to be output.

- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  The status code of 5 digit is returned.

- *data-name-H*

  The length of the output block is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`01603`	The file descriptor specified for *data-name-E* is not the one which was acquired by opening the DAM file normally.
	The DAM file is not open.
`01604`	The value specified for the length of the data to be output is greater than the block length.
`01605`	The sequence of accessing the DAM file is invalid.

Status code	Explanation
`01606`	The relative block number is invalid.
`01620`	An output error occurred.
`01628`	The DAM file to be accessed is protected by the security facility. The UAP attempting to input a physical file block has no access permission.
`01646`	The DAM file to be accessed is protected by the security facility. No ACL exists for the file.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-D*) is invalid.

CBLDCDMB('GET ') - Input a physical file block

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDMB'  USING  unique-name-1   unique-name-2
                         unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMINT'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(1).
    02  data-name-C    PIC X(63).
    02  FILLER         PIC X(3).
    02  data-name-H    PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  data-name-G    PIC S9(9) COMP.
    02  FILLER         PIC X(4).
01  unique-name-2.
    02  data-name-D    PIC X(4) VALUE 'GET '.
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
01  unique-name-3.
    02  data-name-F    PIC X(n).
```

Description

`CBLDCDMB('GET ')` sequentially inputs data in blocks from a physical file of the OpenTP1 file system. Input the physical file block after opening the physical file.

If the value specified for the block length is smaller than the value specified for the buffer length, the length of the input block is returned to *data-name-H*. If the value specified for the block length is greater than the value specified for buffer length, an error is returned.

When inputting a physical file block, specify the physical file name and the file descriptor returned from `CBLDCDMB('OPEN')`.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMINT'` for the interface code used with the DAM file.

- *data-name-C*

  Specify the physical file name, with a path name which comprises 63 characters (special file name + 14 characters) or less. If the physical file name comprises less than 63 characters, pad the remaining portion with space.

- *data-name-E*

  Specify the file descriptor.

- *data-name-G*

  Specify the length of the input buffer.

- *data-name-D*

  Specify `VALUE 'GET△'` for the request code indicating the input of a block from the physical file.

- *data-name-F*

  Specify the input buffer.

- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-H*

  The length of the input block is returned.

Status codes

Status code	Explanation
`00000`	The input block length was specified for *data-name-H* normally.
`01603`	The file descriptor specified for *data-name-E* is not the one which was acquired by opening the file normally.
	The DAM file is not open.
`01604`	The value specified for the block length is greater than the value specified for the buffer length.
`01605`	The sequence of accessing the DAM file is invalid.
`01620`	An input error occurred.

Status code	Explanation
`01628`	The DAM file to be accessed is protected by the security facility. The UAP attempting to input a physical file block has no access permission.
`01637`	The file end was reached.
`01646`	The DAM file to be accessed is protected by the security facility. No ACL exists for the file.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-D*) is invalid.

CBLDCDMB('OPEN') - Open a physical file

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCDMB'  USING  unique-name-1   unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMINT'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(1).
    02  data-name-C    PIC X(63).
    02  FILLER         PIC X(3).
    02  FILLER         PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  FILLER         PIC S9(9) COMP.
    02  data-name-H    PIC S9(9) COMP.
01  unique-name-2.
    02  data-name-F    PIC X(4) VALUE 'OPEN'.
    02  data-name-I    PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z     PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCDMB('OPEN')` opens a physical file created in the OpenTP1 file system. However, it cannot open a physical file being used in online mode.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCDAMINT'` for the interface code used with the DAM file.

- *data-name-C*

  Specify the physical file name, with a path name which comprises 63 characters (special file name + 14 characters) or less. If the physical file name comprises less than 63 characters, pad the remaining portion with space.

- *data-name-H*

  Specify the number of blocks collectively processed which is used as an input/output unit.

- *data-name-F*

  Specify `VALUE 'OPEN'` for the request code indicating that the physical file is open.

- *data-name-I*

  Specify the type of request (creation output request or re-creation (overwrite) output request). The value specified here determines whether to pad the remaining area with space when the file is closed. The value set here will come into effect when `CBLDCDMB('CLOS')` call subsequent to `CBLDCDMB('PUT ')` brings about normal termination. Even though `CBLDCDMB('PUT ')` is called, the remaining area will not be padded with blocks of null characters provided that UAP processing is terminated without calling `CBLDCDMB('CLOS')`.

  `VALUE 'I'`: Creation output request

  `VALUE 'O'`: Re-creation output request

- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-E*

  The file descriptor in return for open processing is returned.

Status codes

Status code	Explanation
`00000`	The file descriptor was specified for *data-name-E* normally.
`01607`	The memory became insufficient.
`01608`	The physical file specified for *data-name-C* is open.
`01611`	The value specified for *data-name-I* is invalid.
`01614`	The physical file name specified is invalid.
`01615`	The value specified for the number of blocks collectively processed is invalid.
`01616`	The physical file specified for *data-name-C* is not a DAM file.
`01618`	The OpenTP1 file system versions used for creation and allocation do not match each other.
`01619`	The physical file specified for *data-name-C* does not exist.

Status code	Explanation
`01620`	An input/output error occurred.
`01628`	The access permission for special files has not been granted.
`01632`	The physical file is not a character special file, or the device corresponding to the special file does not exist.
`01633`	The physical file specified for *data-name-C* has not been initialized as an OpenTP1 file system.
`01635`	The specified value exceeds the maximum number of files which can be opened in the process.
`01636`	The physical file specified for *data-name-C* is being used in online mode, or it is being used by another process.
`01638`	The access permission for physical files has not been granted.
`01639`	Physical file damage was detected.
`01690`	The interface code (*data-name-A)* is invalid.
`01691`	The request code (*data-name-F*) is invalid.

CBLDCDMB('PUT ') - Output a physical file block

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCDMB'  USING  unique-name-1   unique-name-2
                         unique-name-3
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DCDAMINT'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(1).
    02  data-name-C    PIC X(63).
    02  FILLER         PIC X(3).
    02  FILLER         PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  data-name-G    PIC S9(9) COMP.
    02  FILLER         PIC X(4).
01  unique-name-2.
    02  data-name-D    PIC X(4) VALUE 'PUT '.
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  FILLER         PIC X(1).
    02  data-name-Z     PIC S9(9) COMP VALUE ZERO.
01  unique-name-3.
    02  data-name-F    PIC X(n).
```

Description

`CBLDCDMB('PUT ')` sequentially outputs data in blocks to a physical file created in the OpenTP1 file system. If the value specified for the data length is smaller than the value specified for the block length, the remaining part following the data is padded with space. If the value specified for the data length is greater than the value specified for the block length, an error is returned.

When outputting a physical file block, specify the physical file name and the file descriptor returned from `CBLDCDMB('OPEN')` or `CBLDCDMB('CRAT')`.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'DCDAMINT'` for the interface code used with the DAM file.

- *data-name-C*

  Specify the physical file name, with a path name which comprises 63 characters (special file name + 14 characters) or less. If the physical file name comprises less than 63 characters, pad the remaining portion with space.

- *data-name-E*

  Specify the file descriptor.

- *data-name-G*

  Specify the length of the data to be output.

- *data-name-D*

  Specify `VALUE 'PUT∆'` for the request code indicating block output to the physical file.

- *data-name-F*

  Specify the data to be output.

- *data-name-Z*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	The data specified for *data-name-F* was output normally.
`01603`	The specified file descriptor is not the one which was acquired by opening the DAM file normally.
	The DAM file is not open.
`01604`	The value specified for the data length is greater than the value specified for the block length.
`01605`	The sequence for accessing the DAM file is invalid.
`01620`	An output error occurred.
`01628`	The DAM file to be accessed is protected by the security facility. The UAP attempting to output a physical file block has no access permission.
`01637`	The end of the file was reached.

Status code	Explanation
`01646`	The DAM file to be accessed is protected by the security facility. No ACL exists for the file.
`01690`	The interface code (*data-name-A*) is invalid.
`01691`	The request code (*data-name-D*) is invalid.

IST service (CBLDCIST)

This section gives the syntax and other information of the following COBOL-UAP creation programs which access an internode shared table:

- `CBLDCIST('CLOS')` - Close an internode shared table
- `CBLDCIST('OPEN')` - Open an internode shared table
- `CBLDCIST('READ')` - Input an internode shared table record
- `CBLDCIST('WRIT')` - Output an internode shared table record

The COBOL-UAP creation programs for IST service (`CBLDCIST`) can be used only in UAPs of TP1/Server Base. They cannot be used in UAPs of TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language template for IST service (CBLDCIST) is stored in `DCIST.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCIST('CLOS') - Close an internode shared table

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCIST' USING unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02 data-name-A  PIC X(8)  VALUE'DCISTSVC'.
    02 data-name-B  PIC X(5).
    02 FILLER       PIC X(3).
    02 data-name-C  PIC X(8).
    02 FILLER       PIC S9(9) COMP.
    02 FILLER       PIC S9(9) COMP.
    02 data-name-F  PIC S9(9) COMP.
    02 FILLER       PIC X(12).
01  unique-name-2.
    02 data-name-G  PIC X(4)  VALUE 'CLOS'.
    02 FILLER       PIC X(1).
    02 FILLER       PIC X(1).
    02 FILLER       PIC X(1).
    02 FILLER       PIC X(1).
    02 data-name-H  PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCIST('CLOS')` closes the specified internode shared table.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCISTSVC'` for the interface code used with the internode shared table.

- *data-name-C*

  Specify the name of an internode shared table to be closed with up to 8 characters. If the specified name comprises less than 8 characters, pad the remaining portion with space.

- *data-name-F*

  Specify the table descriptor returned when the internode shared table is opened.

- *data-name-G*

  Specify `VALUE 'CLOS'` for the request code indicating that the internode shared table is closed.

- *data-name-H*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	The internode shared table was closed normally.
`03800`	The sequence of accessing the internode shared table is invalid.
`03803`	The table descriptor specified for *data-name-F* is not the one which was acquired by opening the internode shared table normally.
	The internode shared table is not open.
`03811`	The value specified for *data-name-H* is invalid.
`03890`	The interface code (*data-name-A*) is invalid.
`03891`	The request code (*data-name-G*) is invalid.

CBLDCIST('OPEN') - Open an internode shared table

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCIST' USING unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02 data-name-A   PIC X(8) VALUE 'DCISTSVC'.
    02 data-name-B   PIC X(5).
    02 FILLER        PIC X(3).
    02 data-name-C   PIC X(8).
    02 FILLER        PIC S9(9) COMP.
    02 FILLER        PIC S9(9) COMP.
    02 data-name-F   PIC S9(9) COMP.
    02 FILLER        PIC X(12).
01  unique-name-2.
    02 data-name-G   PIC X(4) VALUE 'OPEN'.
    02 FILLER        PIC X(1).
    02 FILLER        PIC X(1).
    02 FILLER        PIC X(1).
    02 FILLER        PIC X(1).
    02 data-name-H   PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCIST('OPEN')` opens the specified internode shared table. When the internode shared table is opened normally, the table descriptor is returned.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCISTSVC'` for the interface code used with the internode shared table.

- *data-name-C*

  Specify the name of an internode shared table to be opened with up to 8 characters. If the specified name comprises less than 8 characters, pad the remaining portion with space.

- *data-name-G*

  Specify `VALUE 'OPEN'` for the request code indicating the internode shared table is opened.

- *data-name-H*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-F*

  The table descriptor of the internode shared table is returned.

Status codes

Status code	Explanation
`00000`	The table descriptor was returned in *data-name-F* normally.
`03800`	The sequence of accessing the internode shared table is invalid.
`03801`	The internode shared table name specified for *data-name-C* has not been defined.
`03807`	The memory became insufficient.
`03808`	The internode shared table whose name is specified for *data-name-C* has already been opened.
`03810`	The length of the value specified for the internode shared table name is invalid.
`03811`	The value specified for *data-name-H* is invalid.
`03890`	The interface code (*data-name-A*) is invalid.
`03891`	The request code (*data-name-G*) is invalid.

CBLDCIST('READ') - Input an internode shared table record

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCIST'  USING unique-name-1  unique-name-2
                       unique-name-n
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A  PIC X(8)  VALUE 'DCISTSVC'.
    02  data-name-B  PIC X(5).
    02  FILLER       PIC X(3).
    02  data-name-C  PIC X(8).
    02  data-name-D  PIC S9(9) COMP.
    02  data-name-E  PIC S9(9) COMP.
    02  data-name-F  PIC S9(9) COMP.
    02  FILLER       PIC X(12).
01  unique-name-2.
    02  data-name-G  PIC X(4)  VALUE 'READ'.
    02  FILLER       PIC X(1).
    02  FILLER       PIC X(1).
    02  FILLER       PIC X(1).
    02  FILLER       PIC X(1).
    02  data-name-H  PIC S9(9) COMP VALUE ZERO.
    02  unique-name-3.
        03  data-name-I  PIC S9(9) COMP.
        03  data-name-J  PIC S9(9) COMP.
    02  unique-name-4.
        03  data-name-I  PIC S9(9) COMP.
        03  data-name-J  PIC S9(9) COMP.
          :
          :
    02  unique-name-m.
        03  data-name-I  PIC S9(9) COMP.
        03  data-name-J  PIC S9(9) COMP.
01  data-name-n.
    02  data-name-K  PIC X(n).
```

Description

`CBLDCIST('READ')` inputs record(s) in the specified range from the specified internode shared table for reference. When multiple records are specified at a time and the specified records contain an error, `CBLDCIST('READ')` does not input any record to the input buffer and returns with an error.

When inputting an internode shared table record, specify the internode shared table name and the table descriptor returned from `CBLDCIST('OPEN')`.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCISTSVC'` for the interface code used with the internode shared table.

- *data-name-C*

  Specify the name of an internode shared table to be accessed with up to 8 characters. If the specified name comprises less than 8 characters, pad the remaining portion with space.

- *data-name-D*

  Specify the number of records from *unique-name-2* to *unique-name-m* (the number of sets of *data-name-I* and *data-name-J*).

- *data-name-E*

  Specify the length of the input buffer with a value which is greater than or equal to (input record length `x` number of input records).

- *data-name-F*

  Specify the table descriptor returned when the internode shared table is opened.

- *data-name-G*

  Specify `VALUE 'READ'` for the request code indicating that an internode shared table record(s) is input.

- *data-name-H*

  Specify `0`.

- *data-name-I*

  Specify the relative record number of the first record to be accessed.

- *data-name-J*

  Specify the relative record number of the last record to be accessed. If 0 is specified, only the record whose relative record number is specified for *data-name-I* is input.

- *data-name-K*

  Specify the input data area (buffer).

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	The entire specified record(s) was input normally.
`03800`	The sequence of accessing the internode shared table is invalid.
`03803`	The table descriptor specified for *data-name-F* is not the one which was acquired by opening the internode shared table normally.
	The internode shared table is not open.
`03804`	The input buffer length specified for *data-name-E* is less than the total length of the record(s).
`03806`	The relative record number is invalid.
`03807`	The memory became insufficient.
`03809`	The value specified for *data-name-D* is less than 1.
`03811`	The value specified for *data-name-H* is invalid.
`03890`	The interface code (*data-name-A*) is invalid.
`03891`	The request code (*data-name-G*) is invalid.

CBLDCIST('WRIT') - Output an internode shared table record

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCIST'  USING unique-name-1  unique-name-2  unique-name-n
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A  PIC X(8)  VALUE 'DCISTSVC'.
    02  data-name-B  PIC X(5).
    02  FILLER       PIC X(3).
    02  data-name-C  PIC X(8).
    02  data-name-D  PIC S9(9) COMP.
    02  data-name-E  PIC S9(9) COMP.
    02  data-name-F  PIC S9(9) COMP.
    02  FILLER       PIC X(12).
01  unique-name-2.
    02  data-name-G  PIC X(4)  VALUE 'WRIT'.
    02  FILLER       PIC X(1).
    02  FILLER       PIC X(1).
    02  FILLER       PIC X(1).
    02  FILLER       PIC X(1).
    02  data-name-H  PIC S9(9) COMP VALUE ZERO.
    02  unique-name-3.
        03  data-name-I  PIC S9(9) COMP.
        03  data-name-J  PIC S9(9) COMP.
    02  unique-name-4.
        03  data-name-I  PIC S9(9) COMP.
        03  data-name-J  PIC S9(9) COMP.
          :
          :
    02  unique-name-m.
        03  data-name-I  PIC S9(9) COMP.
        03  data-name-J  PIC S9(9) COMP.
01  data-name-n.
    02  data-name-K  PIC X(n).
```

Description

`CBLDCIST('WRIT')` outputs record(s) in the specified range to the specified internode shared table. When multiple records are specified at a time and the specified records contain an error, `CBLDCIST('WRIT')` does not output any record to the output buffer and returns with an error.

When `CBLDCIST('WRIT')` terminates normally, the contents of record(s) at the local node are updated. The contents of record(s) at other nodes are updated presently after normal termination of `CBLDCIST('WRIT')`.

When outputting internode shared table record(s), specify the internode shared table name and the table descriptor returned from `CBLDCIST('OPEN')`.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DCISTSVC'` for the interface code used with the internode shared table.

- *data-name-C*

  Specify the name of an internode shared table to be accessed with up to 8 characters. If the specified name comprises less than 8 characters, pad the remaining portion with space.

- *data-name-D*

  Specify the number of records from *unique-name-2* to *unique-name-m* (the number of sets of *data-name-I* and *data-name-J*).

- *data-name-E*

  Specify the length of the output buffer with a value which is greater than or equal to (output record length `x` number of output records).

- *data-name-F*

  Specify the table descriptor returned when the internode shared table is opened.

- *data-name-G*

  Specify `VALUE 'WRIT'` for the request code indicating an internode shared table record(s) is output.

- *data-name-H*

  Specify `0`.

- *data-name-I*

  Specify the relative record number of the first record to be accessed.

- *data-name-J*

  Specify the relative record number of the last record to be accessed. If 0 is specified, only the record whose relative record number is specified for *data-name-I* is output.

- *data-name-K*

  Specify the output data area (buffer).

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	The entire specified record(s) was output normally.
`03800`	The sequence of accessing the internode shared table is invalid.
`03803`	The table descriptor specified for *data-name-F* is not the one which was acquired by opening the internode shared table normally.
	The internode shared table is not open.
`03804`	The output buffer length specified for *data-name-E* is less than the total length of the record(s).
`03806`	The relative record number is invalid.
`03807`	The memory became insufficient.
`03809`	The value specified for *data-name-D* is less than 1.
`03811`	The value specified for *data-name-H* is invalid.
`03841`	The output buffer length is too much longer than the total length of the record(s) to be output.
`03890`	The interface code (*data-name-A*) is invalid.
`03891`	The request code (*data-name-G*) is invalid.

User journal acquisition (CBLDCJNL)

This section gives the syntax and other information of the following COBOL-UAP creation program which is used for user journal acquisition:

- `CBLDCJNL('UJPUT   ')` - Acquire a user journal

The COBOL-UAP creation program for user journal acquisition (CBLDCJNL) can be used only in UAPs of TP1/Server Base. It cannot be used in UAPs of TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language template for user journal acquisition (CBLDCJNL) is stored in `DCJNL.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCJNL('UJPUT   ') - Acquire a user journal

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCJNL'  USING  unique-name-1   unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'UJPUT   '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-C    PIC 9(9) COMP.
    02  data-name-D    PIC 9(9) COMP.
    02  data-name-E    PIC X(n).
```

Description

`CBLDCJNL('UJPUT      ')` acquires a user journal (UJ), which is UAP historical information, into the system journal file (`system_jnl_file`). The unit of UJ acquired by `CBLDCJNL('UJPUT   ')` once is called a UJ record.

A user journal is not output to the system journal file immediately after `CBLDCJNL('UJPUT   ')` is called. The UJ record is output to the system journal file when the journal buffer becomes full or when the transaction processing is committed.

`CBLDCJNL('UJPUT   ')` can be called only after `CBLDCRPC('OPEN ')` has been called and before `CBLDCRPC('CLOSE ')` is called. Even if an error occurs in the transaction processing that called `CBLDCJNL('UJPUT ')`, the UJ record that has already been output cannot be invalidated through rollback processing (partial recovery). Even when rollback processing is executed for the transaction processing that called `CBLDCJNL('UJPUT   ')`, the UJ record is output to the system journal file.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'UJPUT`ΔΔΔ`'` for the request code indicating user journal acquisition.

- *data-name-Z*

  Using one of the following values, specify whether to output the UJ record to the system journal file at acquisition of the UJ record.

  `1`: Output the UJ record to the system journal file at acquisition of the UJ record. If the

UJ record is acquired inside the transaction, this setting is ignored.

`0`: Do not output the UJ record to the system journal file at acquisition of the UJ record.

- *data-name-C*

  Specify the length of the UJ to be acquired. The specified length must be in the range from 1 to (the value specified for the `jnl_max_datasize` operand of the system journal file service definition at the acquisition destination - 8).

- *data-name-D*

  Specify a value from 0 to 255 as a UJ code.

- *data-name-E*

  Specify the UJ data to be acquired. The length of valid data as UJ can be up to the length specified for *data-name-C*.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`01101`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
`01102`	The value specified for the length of user journal (*data-name-C*) is 0 or less.
`01103`	The value specified for the length of user journal (*data-name-C*) exceeds the limit.
`01105`	The `CBLDCRPC('OPEN    ')` function has not been called. Or, the `CBLDCJNL('UJPUT ')` function cannot be used because the execution environment of the applicable system is in the journal fileless mode.

Note

A UJ record that is outside the transaction is output to the system journal file when the journal buffer becomes full or when a transaction of another application terminates normally (when the transaction processing is committed). To acquire the UJ record using an application that does not generate transactions, call `CBLDCJNL('UJPUT ')` in which 1 is set for *data-name-Z* at the appropriate timing.

Journal data editing (CBLDCJUP)

This section gives the syntax and other information of the following `COBOL-UAP` creation programs handling journal data output with the `jnlrput` command. For the relationship between options of the `jnlrput` command and output data and the output format, see the description of syntax of the `jnlrput` command in the *OpenTP1 Operation* manual.

- `CBLDCJUP('CLOSERPT')` - Close the jnlrput output file
- `CBLDCJUP('OPENRPT ')` - Open the jnlrput output file
- `CBLDCJUP('RDGETRPT')` - Input journal data of the jnlrput output file

The COBOL-UAP creation programs for journal data editing (CBLDCJUP) can be used only in UAPs of TP1/Server Base. They cannot be used in UAPs of TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language template for journal data editing (CBLDCJUP) is stored in `DCJUP.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCJUP('CLOSERPT') - Close the jnlrput output file

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCJUP'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8) VALUE 'CLOSERPT'.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-Z   PIC S9(9) COMP VALUE ZERO.
01  unique-name-2.
    02  FILLER        PIC X(256).
    02  data-name-C   PIC S9(9) COMP.
    02  FILLER        PIC 9(9) COMP.
    02  FILLER        PIC 9(9) COMP.
```

Description

`CBLDJUP('CLOSERPT')` closes the execution result file of the `jnlrput` command. Specify the file descriptor returned from `CBLDCJUP('OPENRPT')` for `CBLDCJUP('CLOSERPT')`.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'CLOSERPT'` for the request code indicating that the `jnlrput` command output file is closed.

- *data-name-Z*

  Specify `0`.

- *data-name-C*

  Specify the file descriptor.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	The `jnlrput` command output file was closed normally.
`01101`	The request code (*data-name-A*) is invalid.
`01271`	The file descriptor specified for *data-name-C* is not the one which was acquired by opening the `jnlrput` command output file normally.
	The `jnlrput` command output file is not open.

CBLDCJUP('OPENRPT ') - Open the jnlrput output file

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCJUP'  USING  unique-name-1  unique-name-2
                         unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8) VALUE 'OPENRPT '.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-Z   PIC S9(9) COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-C   PIC X(256).
    02  data-name-D   PIC S9(9) COMP.
    02  FILLER        PIC 9(9) COMP.
    02  FILLER        PIC 9(9) COMP.
01  unique-name-3.
    02  data-name-E   PIC 9(9) COMP.
    02  data-name-F   PIC X(4).
    02  data-name-G   PIC X(8).
    02  data-name-H   PIC 9(9) COMP.
    02  data-name-I   PIC 9(9) COMP.
    02  data-name-J   PIC 9(9) COMP.
    02  data-name-K   PIC X(4).
    02  FILLER        PIC X(96).
```

Description

`CBLDJUP('OPENRPT ')` opens the execution file of the `jnlrput` command.

Specify the name created at execution of the `jnlrput` command as the name of the execution file to be opened.

If an input error or memory shortage occurs, `CBLDJUP('OPENRPT ')` closes the execution file of the `jnlrput` command and returns.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'OPENRPT`Δ`'` for the request code indicating that the `jnlrput` command output file is opened.

- *data-name-Z*

  Specify `0`.

- *data-name-C*

  Specify the name of the `jnlrput` command output file, with a pathname which comprises 256 characters or less. If the specified name comprises less than 256 characters, pad the remaining portion with space.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.
- *data-name-D*

  The file descriptor is returned.
- *data-name-E*

  The length of unique-name-3 is returned.
- *data-name-F*

  The identifier of the `jnlrput` command output file, JUP, is returned.
- *data-name-G*

  The node identifier is returned.
- *data-name-H*

  The length of record management information of the `jnlrput` command output file is returned.
- *data-name-I*

  The length of record data header of the `jnlrput` command output file is returned.
- *data-name-J*

  The maximum record length of the `jnlrput` command output file is returned.
- *data-name-K*

  The format version of the `jnlrput` command output file is returned.

Status codes

Status code	Explanation
`00000`	The file descriptor was set in *data-name-D* normally.
`01101`	The request code (*data-name-A*) is invalid.

Status code	Explanation
`01272`	The `jnlrput` command output file whose name is specified for *data-name-C* is not found.
	An input error (open error) occurred.
`01273`	The file whose name is specified for *data-name-C* is not the `jnlrput` command output file.
`01274`	An input error (read error) occurred.
`01278`	The `jnlrput` command output file whose name is specified for *data-name-C* has already been opened.
`01270`	The memory became insufficient.

CBLDCJUP('RDGETRPT') - Input journal data of the jnlrput output file

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCJUP'  USING unique-name-1  unique-name-2
                  unique-name-3  unique-name-4  unique-name-5
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8) VALUE 'RDGETRPT'.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-Z   PIC S9(9) COMP VALUE ZERO.
01  unique-name-2.
    02  FILLER        PIC X(256).
    02  data-name-C   PIC S9(9) COMP.
    02  data-name-D   PIC 9(9) COMP.
    02  data-name-E   PIC 9(9) COMP.
01  unique-name-3.
    02  data-name-F   PIC 9(9) COMP.
    02  data-name-G   PIC X(1).
    02  data-name-Y   PIC X(1).
    02  FILLER        PIC X(2).
    02  data-name-H   PIC X(4).
    02  data-name-I   PIC X(8).
    02  data-name-J   PIC X(9).
    02  FILLER        PIC X(3).
    02  data-name-K   PIC X(4).
    02  data-name-L   PIC X(8).
    02  FILLER        PIC X(12).
    02  data-name-M   PIC X(12).
    02  data-name-N   PIC X(12).
    02  FILLER        PIC X(1).
    02  FILLER        PIC X(3).
    02  FILLER        PIC 9(9) COMP.
    02  data-name-W   PIC X(4).
    02  FILLER        PIC X(36).
```

```
01  unique-name-4.
    02  data-name-O.
        03  data-name-O1  PIC 9(9) COMP.
        03  data-name-O2  PIC 9(9) COMP.
        03  FILLER        PIC X(120).
    02  data-name-P  REDEFINES  data-name-O.
        03  data-name-P1  PIC X(2).
        03  data-name-P2  PIC X(1).
        03  data-name-P3  PIC X(1).
        03  data-name-P4  PIC X(4).
        03  data-name-P5  PIC X(4).
        03  data-name-P6  PIC X(4).
        03  data-name-P7  PIC X(4).
        03  data-name-P8  PIC X(9).
        03  FILLER        PIC X(3).
        03  data-name-P9  PIC X(32).
        03  data-name-P10 PIC X(4).
        03  data-name-P11 PIC X(4).
        03  data-name-P12 PIC X(4).
        03  data-name-P13 PIC X(4).
        03  data-name-P14 PIC X(4).
        03  data-name-P15 PIC X(4).
        03  FILLER        PIC X(40).
    02  data-name-Q  REDEFINES  data-name-O.
        03  data-name-Q1  PIC X(16).
        03  data-name-Q2  PIC X(10).
        03  FILLER        PIC X(2).
        03  data-name-Q3  PIC X(12).
        03  data-name-Q4  PIC X(9).
        03  FILLER        PIC X(3).
        03  data-name-Q5  PIC X(1).
        03  data-name-Q6  PIC X(1).
        03  FILLER        PIC X(2).
        03  data-name-Q7  PIC 9(9) COMP.
        03  FILLER        PIC X(68).
    02  data-name-R  REDEFINES  data-name-O.
        03  data-name-R1  PIC X(16).
        03  data-name-R2  PIC X(10).
        03  data-name-R3  PIC X(1).
        03  data-name-R4  PIC X(1).
        03  FILLER       PIC X(2).
        03  data-name-R5  PIC 9(9) COMP.
        03  FILLER       PIC X(88).
    02  data-name-S  REDEFINES  data-name-O.
        03  data-name-S1  PIC X(16).
        03  data-name-S2  PIC X(10).
        03  FILLER       PIC X(2).
        03  data-name-S3  PIC X(1).
        03  data-name-S4  PIC X(1).
        03  FILLER       PIC X(2).
        03  data-name-S5  PIC 9(9) COMP.
        03  data-name-S6  PIC 9(9) COMP.
        03  FILLER       PIC X(88).
    02  data-name-T  REDEFINES  data-name-O.
        03  data-name-T1  PIC X(16).
        03  data-name-T2  PIC X(10).
        03  FILLER       PIC X(2).
        03  data-name-T3  PIC X(1).
        03  data-name-T4 PIC X(1).
        03  FILLER        PIC X(2).
        03  data-name-T5 PIC 9(9) COMP.
        03  FILLER        PIC X(92).
    02  data-name-U  REDEFINES  data-name-O.
        03  data-name-U1 PIC X(16).
        03  data-name-U2 PIC X(10).
        03  FILLER        PIC X(2).
        03  data-name-U3 PIC X(12).
        03  data-name-U4 PIC 9(9) COMP.
        03  FILLER       PIC X(84).
01  unique-name-5.
    02  data-name-V       PIC X(n).
```

Description

`CBLDCJUP('RDGETRPT')` inputs journal data in units of (record management information (unique-name-3) + record data header (unique-name-4) + record data (unique-name-5)) sequentially from the `jnlrput` command output file.

Open the `jnlrput` command output file with `CBLDCJUP('OPENRPT ')` before inputting journal data with `CBLDCJUP('RDGETRPT')`.

Specify the length of the journal data input buffer for *data-name-V*. Specify the same length as the specified journal data length. If the length specified for *data-name-V* is less than the journal data length, operation of `CBLDCJUP('RDGETRPT')` is undefined.

When the journal data length specified for *data-name-D* is greater than the actual journal data, the journal data is input and the length of the actual journal data is returned in *data-name-E*.

If the journal data length specified for *data-name-D* is less than the actual journal data, `CBLDCJUP('RDGETRPT')` returns with an error.

If an input error occurs, `CBLDCJUP('RDGETRPT')` closes the `jnlrput` command output file and returns.

When end of file is detected, no journal data is returned.

Specify the file descriptor returned from `CBLDCJUP('OPENRPT ')` for `CBLDCJUP('RDGETRPT')`.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'RDGETRPT'` for the request code indicating that journal data of the `jnlrput` command output file is input.

- *data-name-Z*

  Specify `0`.

- *data-name-C*

  Specify the file descriptor.

- *data-name-D*

  Specify the length of journal data to be input. The length can be 1 to the value of the `jnl_max_datasize` operand in the target system journal service definition.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-E*

  The length of input journal data is returned.

- *data-name-F*

  The total length of *unique-name-3*, *unique-name-4*, and *unique-name-5* is returned.

- *data-name-G*

  The record type of input journal data is returned:

  `'U'`: UJ record is returned.

  `'S'`: SJ record is returned.

  `'I'`: IJ record is returned.

  `'M'`: MJ record is returned.

  `'O'`: OJ record is returned.

  `'A'`: AJ record is returned.

  `'G'` : GJ record is returned.

- *data-name-Y*

  A value indicating whether transaction ID setting information exists in input journal data is returned.

  `0x00`: Transaction ID setting information does not exist.

  `0x80`: Transaction ID setting information exists. If this value is returned, the record transaction global identifier is returned to *data-name-M* and the record transaction branch identifier is returned to *data-name-N*.

- *data-name-H*

  The record acquisition time in input journal data is returned.

- *data-name-I*

  The record acquisition requesting node identifier in input journal data is returned.

- *data-name-J*

  The record acquisition requesting server name in input journal data is returned.

- *data-name-K*

  The record acquisition requesting server time stamp in input journal data is returned.

- *data-name-L*

  The record user information in input journal data is returned.

- *data-name-M*

  The record transaction global identifier in input journal data is returned. If the UJ is outside the transaction, `0` is returned.

- *data-name-N*

  The record transaction branch identifier in input journal data is returned. If the UJ is outside the transaction, `0` is returned.

- *data-name-W*

  The record acquisition time (in microseconds) of the input journal data is returned.

  However, if you input a file to which you previously edited and output journal data of a version earlier than TP1/Server Base 06-01 using the `jnlrput` command, `0` is returned.

- *data-name-O*

  The data header information of input UJ data is returned. The value returned in *data-name-O* is valid only when `U` is returned in *data-name-G*.

  - *data-name-O1*

The data length in input UJ data is returned.

- *data-name-O2*

  The UJ code in input UJ data is returned.

■ *data-name-P*

The data header information of input SJ data is returned. The value returned in *data-name-P* is valid only when `S` is returned in *data-name-G*.

- *data-name-P1*

  The resolution type of transaction and child transaction branch in input SJ data is returned:

  `'C'`: Commit decision

  `'R'`: Rollback decision

  `'HC'`: Commit decision with a command

  `'HR'`: Rollback decision with a command

  `'HM'`: Mix decision with a command

  `'HH'`: Hazard decision with a command

- *data-name-P2*

  The process type in input SJ data is returned:

  `'U'`: Decided in a user server process

  `'R'`: Decided in a recovery process

- *data-name-P3*

  The resolution type of transaction branch in input SJ data is returned:

  `'C'`: Commit decision

  `'R'`: Rollback decision

- *data-name-P4*

  Branch execution time second data in input SJ data is returned.

- *data-name-P5*

  Branch execution time fraction-of-a-second data in input SJ data is returned.

- *data-name-P6*

  Branch synchronization point processing execution time second data in input SJ data is returned.

- *data-name-P7*

Branch synchronization point processing execution time fraction-of-a-second data in input SJ data is returned.

- *data-name-P8*

  The user server name in input SJ data is returned.

- *data-name-P9*

  The service name in input SJ data is returned.

- *data-name-P10*

  Transaction total CPU time (in microseconds) in input SJ data is returned.

- *data-name-P11*

  CPU time for OpenTP1 (in microseconds) in input SJ data is returned.

- *data-name-P12*

  CPU time for UAPs (in microseconds) in input SJ data is returned.

- *data-name-P13*

  CPU time for TP1/FS/Direct Access (in microseconds) in input SJ data is returned.

- *data-name-P14*

  CPU time for TP1/FS/Table Access (in microseconds) in input SJ data is returned.

- *data-name-P15*

  CPU time for ISAM/B (in microseconds) in input SJ data is returned.

■ *data-name-Q*

The data header information of input IJ data is returned. The value returned in *data-name-Q* is valid only when `I` is returned in *data-name-G*.

- *data-name-Q1*

  The input destination logical terminal name in input IJ data is returned.

- *data-name-Q2*

  The application name in input IJ data is returned.

- *data-name-Q3*

  The input message sequence number in input IJ data is returned.

- *data-name-Q4*

  The map name in input IJ data is returned.

- *data-name-Q5*

The input message type in input IJ data is returned.

- *data-name-Q6*

  The sequence identifier in input IJ data is returned.

- *data-name-Q7*

  The input message length in input IJ data is returned.

■ *data-name-R*

The data header information of input MJ data is returned. The value returned in *data-name-R* is valid only when `M` is returned in *data-name-G*.

- *data-name-R1*

  The logical terminal name in input MJ data is returned.

- *data-name-R2*

  The connection name in input MJ data is returned.

- *data-name-R3*

  The MJ type in input MJ data is returned.

- *data-name-R4*

  The sequence identifier in input MJ data is returned.

- *data-name-R5*

  The message length in input MJ data is returned.

■ *data-name-S*

The data header information of input OJ data is returned. The value returned in *data-name-S* is valid only when `O` is returned in *data-name-G*.

- *data-name-S1*

  The output destination logical terminal name in input OJ data is returned.

- *data-name-S2*

  The application name in input OJ data is returned.

- *data-name-S3*

  The output message type in input OJ data is returned.

- *data-name-S4*

  The existence of output sequence number in input OJ data is returned.

- *data-name-S5*

The output message sequence number in input OJ data is returned.

- *data-name-S6*

  The output message length in input OJ data is returned.

■ *data-name-T*

The data header information of input AJ data is returned. The value returned in *data-name-T* is valid only when `A` is returned in *data-name-G*.

- *data-name-T1*

  The output destination logical terminal name in input AJ data is returned.

- *data-name-T2*

  The application name in input AJ data is returned.

- *data-name-T3*

  The output message type in input AJ data is returned.

- *data-name-T4*

  The existence of output sequence number in input AJ data is returned.

- *data-name-T5*

  The output message sequence number in input AJ data is returned.

■ *data-name-U*

The data header information of input GJ data is returned. The value returned in *data-name-U* is valid only when `G` is returned in *data-name-G*.

- *data-name-U1*

  The input destination logical terminal name in input GJ data is returned.

- *data-name-U2*

  The application name in input GJ data is returned.

- *data-name-U3*

  The input message sequence number in input GJ data is returned.

- *data-name-U4*

  The input message length in input UJ data is returned.

■ *data-name-V*

The input journal data is returned. The data invalid as journal data is in the length returned in *data-name-E*.

The value returned in *data-name-V* is valid only when `U`, `I`, `M`, `O`, or `G` is returned in *data-name-G*.

Status codes

Status code	Explanation
`00000`	The file descriptor was set in *data-name-D* normally.
`01275`	End of file was detected.
`01101`	The request code (*data-name-A*) is invalid.
`01271`	The file descriptor specified for *data-name-C* is not the one which was acquired by opening the `jnlrput` command output file normally.
	The `jnlrput` command output file is not open.
`01276`	The length specified for *data-name-D* is less than the data length.
`01274`	An input error (read error) occurred.

Resource lock control (CBLDCLCK)

This section gives the syntax and other information of the following COBOL-UAP creation programs which are used to lock arbitrary user files:

- `CBLDCLCK('GET     ')` - Enable locking of a resource
- `CBLDCLCK('RELALL  ')` - Release all resources from lock
- `CBLDCLCK('RELNAME ')` - Release resource from lock specified by name

The COBOL-UAP creation programs for resource lock control can be used only in UAPs of TP1/Server Base. They cannot be used in UAPs of TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language template for resource lock control (CBLDCLCK) is stored in `DCLCK.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCLCK('GET   ') - Enable locking of a resource

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCLCK'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(8) VALUE 'GET     '.
    02  data-name-B     PIC X(5).
    02  FILLER          PIC X(3).
    02  data-name-Z     PIC S9(9) COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-C     PIC X(16).
    02  data-name-D     PIC X(2).
    02  FILLER          PIC X(6).
    02  data-name-E     PIC X(6).
    02  FILLER          PIC X(2).
    02  data-name-F     PIC X(6).
    02  FILLER          PIC X(2).
    02  data-name-G     PIC X(7) VALUE 'MIGRATE'.
    02  FILLER          PIC X(1).
```

Description

`CBLDCLCK('GET     ')` specifies lock for resources to be used by UAPs. The resource indicated by *data-name-C* is handled under lock. Lock is managed in global transactions which are managed by the OpenTP1 transaction manager.

The lock specified here is released by COBOL-UAP creation program (releasing all resources from lock or specifying a resource name and releasing the resource from lock). The lock is also released when the synchronization point of the global transaction that called `CBLDCLCK('GET     ')` is acquired.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'GET`ΔΔΔΔΔ`'` for the request code indicating lock for resources.

- *data-name-Z*

  Specify `0`.

- *data-name-C*

  Specify the name of the resource for which lock is to be specified. The name can be specified with up to 16-byte alphanumeric characters. The OpenTP1 lock service manages the lock on the basis of the specified resource name.

The lock service does not check the contents of the character string. Specify a logically correct name. If a value other than alphanumeric characters is used to specify the resource name, the deadlock information, the timeout information, and the lckls command might not be displayed normally.

- *data-name-D*

  Specify a lock mode with `VALUE 'PR'` or `VALUE 'EX'`.

  `VALUE 'PR'`: The resource is referenced. Other UAPs are permitted to reference the resource but are not permitted to update it.

  `VALUE 'EX'`: The resource is updated. Other UAPs are not permitted to reference or update the resource.

- *data-name-E*

  Specify processing if the program competes for the resource with another UAP. (The program returns with an error or the program waits until the resource is released.) The following values are available:

  `VALUE 'WAIT△△'`: If the program competes for the resource with another UAP, the program waits until the resource is released.

  `VALUE 'NOWAIT'`: If the program competes for the resource with another UAP, the program returns with an error.

- *data-name-F*

  Specify whether the lock specified here is test lock or not. The following values are available:

  `VALUE 'TEST△△'`: Test lock

  `VALUE 'NOTEST'`: Not test lock

  When test lock is specified, note that the resource specified for *data-name-C* is not under lock even if `CBLDCLCK('GET     ')` terminates normally.

- *data-name-G*

  Specify `VALUE 'MIGRATE'`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00401`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
`00450`	Another UAP is using the specified resource.
`00452`	A deadlock occurred.
`00453`	The resource could not be acquired because a timeout occurred (the wait time specified in the OpenTP1 lock service definition was exceeded).
`00454`	The table for lock is insufficient.
`00455`	`CBLDCLCK('GET    ')` was called from a UAP which was not operating as a transaction.
`00457`	The OpenTP1 library version does not match the lock service version.

CBLDCLCK('RELALL ') - Release all resources from lock

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCLCK'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'RELALL  '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-C    PIC X(7) VALUE 'MIGRATE'.
    02  FILLER         PIC X(1).
```

Description

`BLDCLCK('RELALL  ')` releases all the resources from lock which was specified in `CBLDCLCK('GET     ')`. Use this program when releasing the resources from lock before the synchronization point is acquired.

When the global transaction with lock specified terminates, the OpenTP1 lock service automatically releases the resources from lock. In this case, there is no need to specify release from lock in the UAP.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'RELALL`ΔΔ`'` for the request code indicating that all resources are released from lock.

- *data-name-Z*

  Specify `0`.

- *data-name-C*

  Specify `VALUE 'MIGRATE'`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00401`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
`00455`	`CBLDCLCK('RELALL  ')` was called from a UAP which was not operating as a transaction.
`00456`	The resource could not be acquired in the transaction that called this program.
`00457`	The OpenTP1 library version does not match the lock service version.

CBLDCLCK('RELNAME ') - Release resource from lock specified by name

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCLCK'  USING  unique-name-1   unique-name-2
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'RELNAME '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-C    PIC X(16).
    02  data-name-D    PIC X(7) VALUE 'MIGRATE'.
    02  FILLER         PIC X(1).
```

Description

`CBLDCLCK('RELNAME ')` specifies the name of a resource for which `CBLDCLCK('GET ')` specified lock, and releases the resource from the lock. Use this program when releasing the resource from lock before the synchronization point is acquired.

When the global transaction with lock specified terminates, the OpenTP1 lock service automatically releases the resource from lock. In this case, there is no need to specify release from lock in the UAP.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'RELNAME∆'` for the request code indicating release from lock by specifying the resource name.

■ *data-name-Z*

Specify `0`.

■ *data-name-C*

Specify the name of the resource to be released from lock. The resource name must be the same name as when lock was specified for the resource.

- *data-name-D*

  Specify `VALUE 'MIGRATE'`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination
`00401`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
`00456`	The resource that corresponds to the resource name specified for release from lock does not exist.
`00455`	`CBLDCLCK('RELNAME ')` was called from a UAP which was not operating as a transaction.
`00457`	The OpenTP1 library version does not match the lock service version.

Message log output (CBLDCLOG)

This section gives the syntax and other information of the following COBOL-UAP creation program which is used for message log output from a UAP:

- `CBLDCLOG('PRINT   ')` - Output message log

The COBOL-UAP creation programs for message log output can be used in UAPs of both TP1/Server Base and TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language template for message log output (CBLDCLOG) is stored in `DCLOG.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCLOG('PRINT   ') - Output message log

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCLOG'  USING  unique-name-1  unique-name-2
                         unique-name-3
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'PRINT   '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
    02  data-name-C    PIC X(12).
    02  data-name-D    PIC X(3).
    02  FILLER         PIC X(1).
    02  data-name-E    PIC S9(9) COMP.
01  unique-name-2.
    02  data-name-F    PIC S9(9) COMP.
    02  data-name-G    PIC X(n).
01  unique-name-3.
    02  data-name-H    PIC S9(9) COMP.
```

Description

`CBLDCLOG('PRINT   ')` outputs the character string specified for *data-name-G* to the message log file. Before the output, it adds the following information to the character string through OpenTP1:

- Line header
- OpenTP1 ID
- Date and time
- Request source node name
- Request source program ID
- Message ID

The OpenTP1 assigns ID numbers 05000 to 06999 to the messages used for `CBLDCLOG('PRINT   ')`. To messages output from your UAP, assign ID numbers in the range from `05000` to `06999`.

Even if an error occurs, the status code `00000` might be returned. Consequently, a message log might be missing. The missing message log can be identified by checking the message log serial numbers in the message log file.

If the message log is output more than once from one process, the sequence of output to the message log file is ensured. However, if the message log is output from each of multiple processes, the message logs might not be output to the message log file in the call sequence.

If a communication error (`01901`) or a log service inactive error (`01905`) occurs, the message output from the UAP is edited in the UAP process and is output to the standard error output file. Either of the following codes which indicate the causes of errors is added to the end of the message:

- `E1`: Indicates that the message log could not be output to the message log file because the log service was not activated.
- `E2`: Indicates that the message log could not be output to the message log file due to a communication error.

  Examples:

  `KFCA05201-I SPP1`: A service request was received (`E1`)

  `KFCA05410-I SPP1`: Update processing is starting (`E2`)

When an error other than `E1` or `E2` is detected, the OpenTP1 prints a log message which indicates the cause of the error to the standard error output. This message is assigned the message ID number specified by `CBLDCLOG('PRINT   ')`.

Data area whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'PRINT`ΔΔΔ`'` for the request code indicating a request of message log output.

■ *data-name-Z*

Specify `0`.

■ *data-name-C*

Specify the message ID to be assigned to each message log. The message ID must be in the `KFCA`$n_1n_2n_3n_4n_5$`-x` format (11 characters) and end with a null character. Specify a value from `05000` to `06999` for the serial number ($n_1n_2n_3n_4n_5$) output from the UAP.

■ *data-name-D*

Specify a user-selected value (request source program ID) for identifying the UAP that output the message log. The value must comprise two alphanumeric characters and end with a null character.

- *data-name-E*

  Specify the display color of the message log specified in `CBLDCLOG('PRINT     ')` when the message log is output to the NETM operation support terminal. The following colors are available:

  `VALUE '1'`: Green

  `VALUE '2'`: Red

  `VALUE '3'`: White

  `VALUE '4'`: Blue

  `VALUE '5'`: Purple

  `VALUE '6'`: Sky blue

  `VALUE '7'`: Yellow

  If a value other than the above or space is specified, green is assumed to be specified.

- *data-name-F*

  Specify the length of the character string that is to be output as a message log to the message log file. The length can be specified with up to 222 bytes.

- *data-name-G*

  Specify the character string that is to be output as a message log to the message log file.

- *data-name-H*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`01900`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
`01901`	A communication error occurred or `CBLDCRPC('OPEN ')` was not issued.
`01902`	The memory became insufficient.
`01904`	The system definition is invalid.

Status code	Explanation
`01905`	The message log service is not active.
`01906`	An error occurred when the message log service acquired the information to be added to the message log.

Note

When a large log is output, return of the `CBLDCLOG('PRINT ')` may be delayed. For example, when the volume of output messages greatly increases upon occurrence of an error, the transaction processing time increases. Note that this may cause a slowdown.

Message exchange (CBLDCMCF)

This section gives the syntax and other information of the following COBOL-UAP creation programs which are used for message exchange communication:

- `CBLDCMCF('ADLTAP   ')`: Delete an application timer start request
- `CBLDCMCF('APINFO   ')`: Report the application information
- `CBLDCMCF('CLOSE    ')`: Close the MCF environment
- `CBLDCMCF('COMMIT   ')`: Commit an MHP
- `CBLDCMCF('CONTEND  ')`: Terminate continuous-inquiry-response processing
- `CBLDCMCF('EXECAP   ')`: Activate an application program
- `CBLDCMCF('MAINLOOP')`: Start an MHP service
- `CBLDCMCF('OPEN     ')`: Open the MCF environment
- `CBLDCMCF('RECEIVE ')`: Receive a message#
- `CBLDCMCF('RECVSYNC')`: Receive a synchronous message#
- `CBLDCMCF('REPLY    ')`: Send a response message#
- `CBLDCMCF('RESEND   ')`: Resend a message#
- `CBLDCMCF('ROLLBACK')`: Enable MHP rollback
- `CBLDCMCF('SEND     ')`: Send a message#
- `CBLDCMCF('SENDRECV')`: Exchange a synchronous message#
- `CBLDCMCF('SENDSYNC')`: Send a synchronous message#
- `CBLDCMCF('TACTCN   ')`: Establish connection#
- `CBLDCMCF('TACTLE   ')`: Release a logical terminal from shutdown status#
- `CBLDCMCF('TDCTCN   ')`: Release connection#
- `CBLDCMCF('TDCTLE   ')`: Shut down a logical terminal#
- `CBLDCMCF('TDLQLE   ')`: Delete a logical terminal's output queue
- `CBLDCMCF('TEMPGET ')`: Accept temporary-stored data
- `CBLDCMCF('TEMPPUT ')`: Update temporary-stored data
- `CBLDCMCF('TIMERCAN')`: Cancel user timer monitoring

- `CBLDCMCF('TIMERSET')`: Set user timer monitoring
- `CBLDCMCF('TLSCN    ')`: Acquire a connection status#
- `CBLDCMCF('TLSCOM   ')`: Acquire status of MCF communication services
- `CBLDCMCF('TLSLE    ')`: Acquire a logical terminal status#
- `CBLDCMCF('TLSLN    ')`: Acquire the acceptance status for a server-type connection establishment request#
- `CBLDCMCF('TOFLN   ')`: Stop accepting server-type connection establishment requests#
- `CBLDCMCF('TONLN   ')`: Start accepting server-type connection establishment requests#

#: For details on the syntax of COBOL-UAP creation programs, see the applicable *OpenTP1 Protocol* manual.

The COBOL-UAP creation programs for message exchange can be used only in UAPs of TP1/Server Base. They cannot be used in UAPs of TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language template for message exchange (CBLDCMCF) is stored in `DCMCF.cbl` under the `/BeTRAN/examples/COBOL/` directory. For the usage of the API stored in `DCMCF.cbl`, see the syntax description in the applicable *OpenTP1 Protocol* manual.

Note

In the DATA DIVISION specification, the first character of the unique name must be placed at the boundary of the word length.

CBLDCMCF('ADLTAP ') - Delete an application timer start request

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8)  VALUE 'ADLTAP  '.
    02  data-name-B    PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-C    PIC X(4) VALUE SPACE.
    02  data-name-D    PIC X(28) VALUE SPACE.
    02  data-name-E    PIC 9(9)  COMP.
    02  data-name-F1   PIC X(8).
    02  data-name-F2   PIC X(56) VALUE SPACE.
    02  data-name-G    PIC X(8)  VALUE SPACE.
    02  data-name-H    PIC X(8)  VALUE SPACE.
    02  data-name-I    PIC X(144) VALUE SPACE.
    02  data-name-J    PIC X(184) VALUE SPACE.
    02  data-name-K    PIC 9(9)  COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-L    PIC 9(9)  COMP VALUE ZERO.
```

Description

`CBLDCMCF('ADLTAP  ')` deletes a specified application timer start request and cancels startup of the application. Note that this function cannot delete application timer start requests of the `ans` and `cont` types.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'ADLTAP`ΔΔ`'` for the request code indicating deletion of an application timer start request.

- *data-name-C*, *data-name-D*

  Specify a space.

- *data-name-E*

  Specify the application start process identifier of the application start service that has the target application that is to be processed. The permitted value range is from 1 to

239.

- *data-name-F1*

  Specify the name of the application whose start is to be canceled. Express the application name as a maximum of 8 bytes. If the specified name is shorter than 8 bytes, pad the name with trailing spaces.

- *data-name-F2, data-name-G, data-name-H, data-name-I, data-name-J*

  Specify a space.

- *data-name-K, data-name-L*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71001`	`CBLDCMCF('ADLTAP  ')` cannot be accepted because MCF is under start processing.
`71002`	`CBLDCMCF('ADLTAP  ')` cannot be accepted because MCF is under termination processing.
`71004`	A memory shortage occurred during `CBLDCMCF('ADLTAP  ')` processing.
`71005`	A communication error occurred. For the cause, see the message log file.
`71006`	An internal error occurred. For the cause, see the message log file.
`71007`	The specified application name has not been registered.
	No timer start request has been issued for the specified application name.
	The specified application name belongs to an application whose type is inquiry-response or continuous-inquiry-response.
`71009`	`CBLDCMCF('ADLTAP  ')` is not supported by the applicable application start process.
`71010`	Although the request to delete the specified application timer start request was issued, the request was not accepted. For the cause, see the message log file.
`72028`	The value specified for *data-name-A* is invalid.
`72052`	A nonzero value is specified for *data-name-K*.

Status code	Explanation
`72053`	A nonzero value is specified for *data-name-L*.
`72058`	The value specified for *data-name-C* is not a space.
`72059`	The value specified for *data-name-D* is not a space.
`72061`	A value of `0` or smaller or greater than `239` is specified for *data-name-E*.
`72063`	*data-name-F1* begins with a space.
`72065`	The value specified for *data-name-F2* is not a space.
`72066`	The value specified for *data-name-G* is not a space.
`72068`	The value specified for *data-name-H* is not a space.
`72070`	The value specified for *data-name-I* is not a space.
`72072`	The value specified for *data-name-J* is not a space.
`72074`	The character string specified for *data-name-F1* contains an invalid character.

CBLDCMCF('APINFO ') - Report the application information

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCMCF' USING unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'APINFO  '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(4).
    02  data-name-D    PIC X(4).
    02  data-name-E    PIC X(2).
    02  data-name-F    PIC X(2).
    02  data-name-G    PIC X(8).
    02  data-name-H    PIC X(28).
01  unique-name-2.
    02  data-name-I    PIC X(4).
    02  data-name-J    PIC X(8).
    02  data-name-K    PIC X(2).
    02  data-name-L    PIC X(2).
    02  data-name-M    PIC X(4).
    02  data-name-M1   PIC X(1).
    02  data-name-M2   PIC X(3).
    02  data-name-N    PIC X(31).
    02  data-name-N1   PIC X(4).
    02  data-name-N2   PIC X(4).
    02  data-name-N3   PIC X(1).
    02  data-name-O    PIC X(31).
    02  data-name-P    PIC X(4).
    02  data-name-Q    PIC X(5).
    02  data-name-R    PIC 9(9)  COMP.
    02  data-name-S    PIC 9(9)  COMP.
    02  data-name-T    PIC 9(9)  COMP.
    02  data-name-U    PIC X(4).
    02  data-name-V    PIC X(1).
    02  data-name-W    PIC X(79).
```

Description

`CBLDCMCF('APINFO  ')` acquires various types of information on the application from an MHP.

The application information on the MHP that called `CBLDCMCF('APINFO  ')` or the other MHP is reported.

The application information becomes effective only when `CBLDCMCF('APINFO  ')` is normally terminated.

Data area whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'APINFO△△'` for the request code indicating an application information report.

- *data-name-C*

  Specify one of the following values according to the application to be referenced:

  `'SELF'`: Specify this value to acquire the application information on MHP that called `CBLDCMCF('APINFO  ')`.

  `'OTHE'`: Specify this value to acquire the information on a specific application according to the process identifier for MCF communication service in which the application definition is included, and application name.

- *data-name-D*

  Specify space.

- *data-name-E*

  Specify the process identifier.

  - When specifying `'SELF'` for *data-name-C*

    Specify space.

  - When specifying `'OTHE'` for *data-name-C*

    Specify the MCF communication process identifier or application startup process identifier in which the definition of the application to be referenced is included.

- *data-name-F*

  Specify space.

- *data-name-G*

  Specify the application name.

  - When specifying `'SELF'` for *data-name-C*

    Specify space.

  - When specifying `'OTHE'` for *data-name-C*

    Specify the name of the application to be referenced.

    When specifying an error event name (`ERREVT1`, `ERREVT2`, `ERREVT3`, or `ERREVT4`), the default value of the application definition, the NO-response type

('N') is set for *data-name-M1*.

- *data-name-H*

  Specify `LOW_VALUE`.

- *data-name-I, data-name-L, data-name-M2, data-name-N3,* and *data-name-Q*

  Specify space.

- *data-name-W*

  Specify `LOW_VALUE`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-J*

  The name of the application whose information is to be reported is returned.

- *data-name-K*

  The process identifier for MCF communication service that includes the definition of the application whose information is to be reported is returned.

- *data-name-M*

  The shutdown or release shutdown status of the application is returned as follows:

  `'INDA'`: Input shutdown status

  `'SCDA'`: Schedule shutdown status

  `'DACT'`: Input and schedule shutdown status

  `'ACT△'`: Release shutdown status

- *data-name-M1*

  The type of the application is returned as follows:

  (The type specified in the type operand of the -n option in the MCF application definition mcfaalcap is set here).

  `'A'`: Response type

  `'N'`: NO-response type

  `'C'`: Continuous-inquiry-response type

  When specifying `'OTHE'` for *data-name-C* and specifying an error event name (`ERREVT1`, `ERREVT2`, `ERREVT3`, or `ERREVT4`) for *data-name-G*, the actual type is not reported. In this case, the default value of the application definition, NO-response type

(`'N'`) is set here.

- *data-name-N*

  The name of the service group corresponding to the application is returned.

- *data-name-N1*

  The shutdown or release shutdown status of the service group is returned as follows:

  `'INDA'`: Input shutdown status

  `'SCDA'`: Schedule shutdown status

  `'DACT'`: Input and schedule shutdown status

  `'ACT△'`: Release shutdown status

- *data-name-N2*

  The holding or release holding status of the service group is returned as follows:

  `'INHO'`: Input holding status

  `'SCHO'`: Schedule holding status

  `'HOLD'`: Input and schedule holding status

  `'RLSS'`: Release holding status

- *data-name-O*

  The name of the service corresponding to the application is returned.

- *data-name-P*

  The shutdown or release shutdown status of the service is returned as follows:

  `'INDA'`: Input shutdown status

  `'SCDA'`: Schedule shutdown status

  `'DACT'`: Input and schedule shutdown status

  `'ACT△'`: Release shutdown status

- *data-name-R*

  The limit elapsed time for the non-transaction attribute MHP in seconds is returned.

  When `'TRN△'` is set in *data-name-U*, 0 is set here.

  (The value specified in the `ntmetim` operand of the `-v` option in the MCF application definition `mcfaalcap` is set here. If the MCF application definition is omitted, the value specified in the `ntmetim` operand of the `-u` option in the MCF manager definition `mcfmuap` is used.)

- *data-name-S*

  The size of the temporary-stored data storage area for the continuous-inquiry response is returned.

  When the value set in *data-name-M1* is not `'C'`, 0 is set here.

  (The value specified in the `tempsize` operand of the `-n` option in the MCF application definition `mcfaalcap` is set here.)

- *data-name-T*

  The maximum number of input messages that can be stored is returned.

  (The value specified in the `msgcnt` operand of the `-n` option in the MCF application definition `mcfaalcap` is set here.)

- *data-name-U*

  The transaction attribute of the application is returned as follows:

  (The value specified in the `trnmode` operand of the `-n` option in the MCF application definition `mcfaalcap` is set here.)

  `'TRN∆'`: Managed as a transaction

  `'NTRN'`: Not managed as a transaction

- *data-name-V*

  The queue to which the received message is assigned is returned as follows:

  (The value specified in the `quekind` operand of the `-g` option in the MCF application definition `mcfaalcap` is set here.)

  `'D'`: When the message is assigned to the disk queue

  `'M'`: When the message is assigned to the memory queue

Status codes

Status code	Explanation
`00000`	Normal termination.
`72000`	`CBLDCMCF('APINFO  ')` was called from a service other than the MHP service.
`72001`	The value specified for *data-name-G* is invalid. Combination of the values specified for *data-name-G* and *data-name-E* is invalid.
`72016`	The value specified for *data-name-C* is invalid. The value specified for *data-name-E* is invalid. The value specified for *data-name-H* is invalid. The value specified for *data-name-W* is invalid.

Status code	Explanation
`72028`	The value specified for *data-name-A* is invalid.
Other than the above	Unprecedented error (e.g., program damage) occurred.

Note

When two or more MHPs for ERREVT1, ERREVT2, ERREVT3, and ERREVT4 are started at the same time, the MHPs for the same error event name may have a different application type. For the MHPs other than the MHP that called `CBLDCMCF('APINFO ')`, the application type for the error event (ERREVT1, ERREVT2, ERREVT3, or ERREVT4) is not reported. In this case, the default value of the MCF application definition, NO-response type is reported.

CBLDCMCF('CLOSE   ') - Close the MCF environment

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'CLOSE   '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9).
    02  data-name-D    PIC X(12).
```

Description

`CBLDCMCF('CLOSE   ')` closes the environment in which MCF facilities are used. Call `CBLDCMCF('CLOSE   ')` only once in the process before the UAP that called `CBLDCMCF('OPEN   ')` calls `CBLDCRPC('CLOSE   ')` in the main program.

Data area whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'CLOSE`△△△`'` for the request code indicating MHP termination.

- *data-name-C*

  Specify `0`.

- *data-name-D*

  Specify `LOW-VALUE`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`72016`	The value specified for *data-name-D* is invalid.

Status code	Explanation
`72028`	The value specified for *data-name-A* is invalid.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCMCF('COMMIT ') - Commit an MHP

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1
```

■ DATA DIVISION specification

```
01  unique-name-1.
   02  data-name-A    PIC X(8) VALUE 'COMMIT  '.
   02  data-name-B    PIC X(5).
   02  FILLER         PIC X(3).
   02  data-name-C    PIC X(16).
```

Description

`CBLDCMCF('COMMIT  ')` notifies the UAP at the transaction branch as a root transaction branch making up the transaction, the transaction service, and the resource manager that the global transaction initiated by the MHP has terminated processing normally (the global transaction has been committed).

When the `CBLDCMCF('COMMIT  ')` returns normally, a new global transaction is started.

If a global transaction consists of multiple transaction branches (it involves programs other than the MHP which called this `CBLDCMCF('COMMIT  ')`), the entire global transaction will not be committed until each transaction branch is committed. If the global transaction is composed of multiple resource managers, it will not be committed until the results of each resource manager's processing are committed. If the global transaction is not committed, all the transaction branches are rolled back and the program returns with an error, giving the status code of `70906`.

`CBLDCMCF('COMMIT  ')` can be used only by an MHP specified as nonresponse type (`type=noans`) in the MCF application definition. If it is used by an MHP of another type, it returns with an error, giving the status code of `72000`. If it is called by a UAP other than an MHP, it also returns with an error, giving the status code of `72000`.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'COMMIT△△'` as the request code indicating MHP commitment.

■ *data-name-C*

Specify `LOW-VALUE`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination. If this status code returns, the process which called `CBLDCMCF('COMMIT  ')` has started a new transaction.
`70906`	The transaction was not committed, but was rolled back. If this status code returns, the process which called `CBLDCMCF('COMMIT  ')` has started a new transaction.
`70907`	The global transaction which called `CBLDCMCF('COMMIT  ')` was subjected to a heuristic decision which brought about the following: Some transaction branches were committed, whereas other transaction branches were rolled back. If this status code returns, the process which called `CBLDCMCF('COMMIT  ')` has started a new transaction.
`70908`	The transaction branch of the global transaction was completed heuristically. However, the synchronization point of the heuristically completed transaction branch cannot be identified because of an error. If this status code returns, the process which called `CBLDCMCF('COMMIT  ')` has started a new transaction. The status code `00904` will also be returned if `00000001` is assigned to the `trn_extend_function` operand in the transaction service definition and if the resource manager returns `XAER_NOTA` after a one-phase commit.
`72000`	Return at MHP execution: The MHP called `CBLDCMCF('COMMIT  ')` before `CBLDCMCF('RECEIVE ')` for receiving the first segment. `CBLDCMCF('COMMIT  ')` was called by an MHP which is not nonresponse type (`type=noans`). `CBLDCMCF('COMMIT  ')` was called by an MHP with the nontransaction attribute.
	Return at SPP execution: `CBLDCMCF('COMMIT  ')` cannot be called by SPPs.
`72028`	The value specified for *data-name-A* is invalid.
Others than the above	An unprecedented error (e.g., program damage) occurred.

Notes

Even when `CBLDCMCF('COMMIT  ')` returns normally, the input message is not deleted from the input queue. This means that when message processing is restarted after the MHP is rescheduled, the already committed range (up to what point the results of processing have been committed) is unknown. The MHP is rescheduled when:

- An MCF event is reported to schedule an MHP for MCF event processing.

- Since the system is terminated abnormally, OpenTP1 reschedules the MHP for the process.

If message processing is to be continued by the rescheduled MHP, the user is responsible for learning the committed range of processing results.

CBLDCMCF('CONTEND ') - Terminate continuous-inquiry-response processing

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'CONTEND '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(16).
```

Description

`CBLDCMCF('CONTEND  ')` terminates continuous-inquiry-response processing. Before terminating continuous-inquiry-response processing, verify that *data-name-R* of `CBLDCMCF('REPLY  ')` called from the MHP indicates space and that `CBLDCMCF('EXECAP  ')` for activating a cont type MHP has not been called. If the MHP to be activated next is specified for *data-name-R* of `CBLDCMCF('REPLY  ')` or if `CBLDCMCF('EXECAP  ')` for activating a cont type MHP has been called, `CBLDCMCF('CONTEND ')` returns with an error.

After `CBLDCMCF('CONTEND ')` is called, temporary-stored data cannot be accessed.

Data area whose values are set in the UAP

- *data-name-A*

  Specify `'VALUE 'CONTEND△'` for the request code indicating termination of continuous-inquiry-response processing.

- *data-name-C*

  Specify `LOW-VALUE`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`72000`	Return at MHP execution: The MHP called `CBLDCMCF('CONTEND ')` before `CBLDCMCF('RECEIVE')` for receiving the first segment.
	Return at SPP execution: `CBLDCMCF('CONTEND ')` cannot be called by SPPs.
`72016`	The value specified for *data-name-C* is invalid.
`72028`	The value specified for *data-name-A* is invalid.
`72101`	`CBLDCMCF('CONTEND ')` was called by an MHP which is not continuous-inquiry-response type (`type=cont`).
`72107`	`CBLDCMCF('CONTEND ')` has already been called.
`72111`	The cont type application to be activated next was specified, `CBLDCMCF('REPLY  ')` was called, then `CBLDCMCF('CONTEND ')` was called.
	The cont type application to be activated next was specified, `CBLDCMCF('EXECAP ')` was called, then `CBLDCMCF('CONTEND ')` was called.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCMCF('EXECAP ') - Activate an application program

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1  unique-name-2
                         unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(8) VALUE 'EXECAP  '.
    02  data-name-B     PIC X(5).
    02  FILLER          PIC X(3).
    02  data-name-C     PIC X(4).
    02  data-name-D     PIC X(4).
    02  data-name-E     PIC 9(8).
    02  data-name-F     PIC 9(8).
    02  data-name-G     PIC 9(9) COMP.
    02  data-name-H     PIC X(4).
    02  data-name-I     PIC X(4).
    02  data-name-J     PIC X(4).
    02  data-name-K     PIC X(4).
    02  data-name-L     PIC X(8).
    02  data-name-M     PIC X(4).
    02  data-name-N     PIC X(8).
    02  data-name-O1    PIC X(4).
    02  data-name-O2    PIC 9(9) COMP.
    02  data-name-O3    PIC 9(9) COMP.
    02  data-name-O4    PIC X(1).
    02  data-name-O5    PIC X(1).
    02  data-name-P     PIC X(14).
01  unique-name-2.
    02  data-name-Q     PIC X(4).
    02  data-name-R     PIC X(8).
    02  data-name-S     PIC X(8).
    02  data-name-T     PIC X(6).
    02  data-name-U     PIC X(2).
    02  data-name-V     PIC X(28).
01  unique-name-3.
    02  data-name-W     PIC 9(x) COMP.
    02  data-name-X     PIC X(x).
    02  data-name-Y     PIC X(n).
```

Description

`CBLDCMCF('EXECAP     ')` starts the MHP or SPP of the application name specified for *data-name-N* from a UAP (SPP or MHP). After the transaction or service function has terminated, the MHP or SPP with the application name specified for *data-name-N* can be started immediately or after a preset length of time.

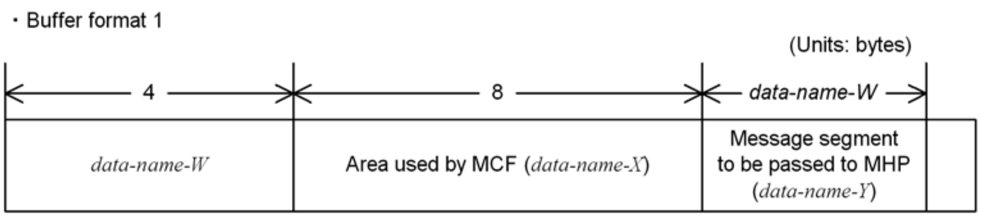
To start an application program from an SPP, process the SPP as a transaction and call `CBLDCMCF('OPEN    ')` in the SPP main function.

If an MHP is activated from another MHP, the name in the first-received message is used as the logical terminal name of the input source that receives messages through the activated MHP. If the application program is started from the MHP, the name in the first-received message is also used as the logical terminal name of the input source that receives messages.

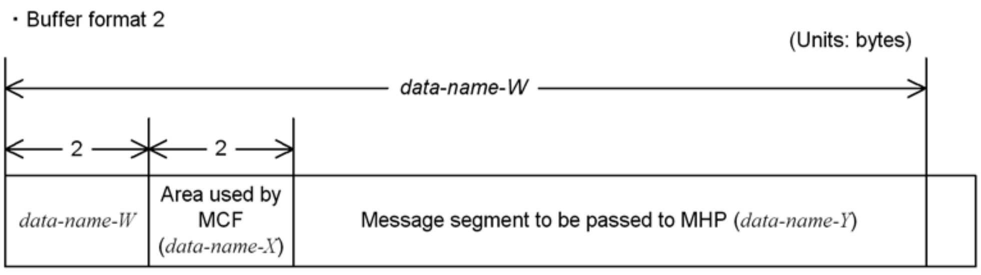
If an MHP is activated from an SPP, "*" is used as the logical terminal name of the input source that receives messages through the activated MHP. If the application program is started from the MHP, "*" is also used as the logical terminal name of the input source that receives messages.

The maximum length of a single message segment that can be sent is 32 kilobytes. Note that the actual value might be smaller depending on the protocol. For details, see the applicable *OpenTP1 Protocol* manual.

The figure below shows the format of the area (indicated by *unique-name-3*) of the segment which is passed to the MHP to be started.



・Buffer format 1
(Units: bytes)
4
8
data-name-W
data-name-W
Area used by MCF (data-name-X)
Message segment to be passed to MHP (data-name-Y)



・Buffer format 2
(Units: bytes)
data-name-W
2
2
data-name-W
Area used by MCF (data-name-X)
Message segment to be passed to MHP (data-name-Y)

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'EXECAP△△'` for the request code indicating activation of an application program.

- *data-name-C, data-name-D*

  Specify space.

- *data-name-E, data-name-F*

  These areas are used by the MCF.

- *data-name-G*

  Specify `0`.

- *data-name-H*

  Specify whether the segment to be passed to the MHP or SPP is the last segment of a logical message. The following values are available:

  `'ESI∆'`: Specify this value to pass the first segment or an intermediate segment. If `CBLDCMCF('EXECAP    ')` with `'ESI∆'` specified is used, `CBLDCMCF('EXECAP    ')` with `'EMI∆'` specified for *data-name-H* must be used.

  `'EMI∆'`: Specify this value to pass the last segment. If the logical message comprises only a single segment, also specify `'EMI∆'`.

  Also specify `'EMI∆'` if the sending of the first or an intermediate segment is to be followed by the notice of the completion of message sending.

- *data-name-I, data-name-J, data-name-K*

  Specify space.

- *data-name-L*

  - Interval timer start (specification of `'INTV'` for *data-name-O1*)

    Specify the period of time which will elapse from the calling `CBLDCMCF('EXECAP    ')` to the activation of the MHP or SPP. The value must be specified in the format of `HHMMSS00` (where `HH` indicates hours, `MM` indicates minutes, `SS` indicates seconds, and `00` is fixed) in the range from `'00000100'` (activation after 1 second) to `'99595900'` (activation after 99 hours 59 minutes 59 seconds).

  - Time-point timer start (specification of `'TIME'` for *data-name-O1*)

    Specify when to activate the MHP or SPP. The time must be specified in the format of `HHMMSS00` (where `HH` indicates hours, `MM` indicates minutes, `SS` indicates seconds, and `00` is fixed) in the range from `'00000000'` (activation at 00:00:00) to `'23595900'` (activation at 23:59:59) in local time.

    The value specified for *data-name-L* is valid only for timer-start. If immediate start is specified, the value specified for *data-name-L* is ignored.

    Since OpenTP1 checks whether the activation time has been reached at regular intervals, there is a difference between the time specified for *data-name-L* and the actual activation time. The accuracy of time monitoring depends on the value for the time monitoring interval specified for the `btim` operand in the `-t` option of the MCF communication configuration definition `mcfttim`.

- *data-name-M*

  Specify space.

- *data-name-N*

  Specify the application name of the MHP or SPP to be started after the MHP that called `CBLDCMCF('EXECAP    ')` terminates. The application name can be specified with up to 8 bytes. The application name must end with space. If the specified name comprises less than 8 bytes, pad the remaining portion with space.

- *data-name-O1*

  Specify when to activate the MHP. The following values are available:

  `'JUST'`: Specify this value for immediate start. When `VALUE 'JUST'` is specified, the value specified for *data-name-L* is ignored.

  `'INTV'`: Specify this value for interval timer start. The MHP will start the time specified for *data-name-L* after `CBLDCMCF('EXECAP    ')` is called.

  `'TIME'`: Specify this value for time point timer start. The MHP or SPP will start at the time specified for *data-name-L*.

  Space: The default value `'JUST'` (immediate start) is assumed.

- *data-name-O2, data-name-O3*

  Specify `0`.

- *data-name-O4*

  Specify space.

- *data-name-O5*

  Specify the buffer format to be used. The following values are available:

  `'1'`: Specify this value to use buffer format 1.

  `'2'`: Specify this value to use buffer format 2.

  Space: The specification for *data-name-O5* is assumed to be omitted. `'1'` (buffer format 1) is assumed.

- *data-name-P*

  Specify `LOW-VALUE`.

- *data-name-Q, data-name-R, data-name-S, data-name-T, data-name-U*

  Specify space.

- *data-name-V*

  Specify `LOW-VALUE`.

- *data-name-W*

  With buffer format 1: `PIC 9(9)`

  Specify the send segment length.

  With buffer format 2: `PIC 9(4)`

  Specify the send segment length + 4.

  Specify `0` as the send segment length if the sending of the first or an intermediate segment is to be followed by the notice of the completion of message sending.

- *data-name-X*

  With buffer format 1: `PIC X(8)`

  With buffer format 2: `PIC X(2)`

  This area is used by the MCF.

- *data-name-Y*

  Specify the contents of the message segment to be passed to the MHP or SPP to be activated. Specify *data-name-Y* also when the sending of the first or an intermediate segment is to be followed by the notice of the completion of message sending.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71002`	An error occurred during input/output processing for the message queue.
	The message queue is in descriptor state.
	No message queue was allocated.
	The value specified for the segment length exceeds 32,000 bytes.
	The MHP or SPP specified for *data-name-N* cannot be activated because the MCF is being terminated.
`71003`	The message queue is full.

Status code	Explanation
`71004`	The buffer for storing messages could not be acquired in the memory.
`71108`	An attempt was made to start the MHP or SPP of the application name specified for *data-name-N*, but the MHP or SPP management table could not be acquired.
	The local memory of the process is insufficient.
`72000`	Return at MHP execution: The MHP called `CBLDCMCF('EXECAP  ')` before `CBLDCMCF('RECEIVE ')` for receiving the first segment.
	Return at SPP execution: `CBLDCMCF('EXECAP  ')` was called from a nontransactional SPP process.
`72001`	The specified application name is not defined with the MCF.
	The application name is incorrect.
	The application startup process name is not specified in the communication service definition (`mcfmcname` definition command) included in the MCF manager definition.
	No application startup process identifier is specified in the environment definition (`-p` option to the `mcfaenv` definition command) included in the MCF application definition for the application startup process.
	The following two values do not match: • Application startup process identifier specified in the application environment definition (`-p` option to the `mcfaenv` definition command) • Application startup process identifier specified in the communication configuration definition (`mcftenv` definition command) for the application startup process
	If a non-response type MHP or SPP is to be started: • No logical terminal (the `lname` operand of the `-n` option to the `mcfaalcap` definition command) is specified in the application attribute definition for the application to be started. • The logical terminal specified in the application attribute definition for the application to be started is not defined in the communication configuration definition (`mcftalcle` definition command) for the application startup process. • The logical terminal specified in the application attribute definition of the application to be started is not for send-only communication (`mcftalcle -t=send`). • The logical terminal specified in the application attribute definition for the application to be started cannot use the application startup process.

Status code	Explanation
	If an MHP of the response type or continuous inquiry-response type is to be started: • No internal communication path (`cname` operand of the `-n` option to the `mcfaalcap` definition command) is specified in the application attribute definition for the application to be started. • The internal communication path specified in the application attribute definition for the application to be started is not defined in the communication configuration definition (`-c` option to the `mcftpsvr` definition command) for the application startup process. • The inquiry logical terminal (`mcftalcle -t=request`) is not specified in the communication configuration definition (`mcftalcle` definition command) of the application start process.
	If an application is to be started from an SPP: • The `mcf_psv_id` operand in the user service definition or user service default definition for the caller UAP is assigned no application startup process identifier. • The following two values do not match: Application startup process identifier assigned to the `mcf_psv_id` operand in the user service definition or user service default definition for the caller UAP Application startup process identifier specified in the communication configuration definition (`-s` option to the `mcftenv` definition command) for the application startup process or in the application environment definition (`-p` option to the `mcfaenv` definition command) • The following two values do not match: MCF manager identifier assigned to the `mcf_mgrid` operand in the user service definition or user service default definition for the caller UAP Identifier of the MCF manager to which the application startup process belongs
`72005`	A value less than 1 byte (with buffer format 1) or less than 5 bytes (with buffer format 2) was specified for the message segment length (*data-name-W*) in `CBLDCMCF('EXECAP ')` in which `'ESI Δ '` was specified for *data-name-H*.
`72007`	From a response type (`type=ans`) MHP that called `CBLDCMCF('REPLY    ')`, another response type MHP was started by calling `CBLDCMCF('EXECAP  ')`.
	From a continuous-inquiry-response type (`type=cont`) MHP that called `CBLDCMCF('REPLY    ')`, another continuous-inquiry-response type MHP was started by calling `CBLDCMCF('EXECAP  ')`.
`72009`	`CBLDCMCF('EXECAP  ')` was called more than once from a response type (`type=ans`) MHP.
	`CBLDCMCF('EXECAP  ')` was called more than once from a continuous-inquiry-response type (`type=cont`) MHP.
`72011`	From an MHP which is not response type (`type=ans`), a response type MHP was started by calling `CBLDCMCF('EXECAP  ')`.
	From an MHP which is not continuous-inquiry-response type (`type=cont`), a continuous-inquiry-response type MHP was started by calling `CBLDCMCF('EXECAP  ')`.

Status code	Explanation
`72016`	The value specified for *data-name-O1*, *data-name-O2*, *data-name-O3*, *data-name-P*, or *data-name-V* is invalid.
`72024`	The value specified for *data-name-Q* is invalid.
`72026`	The value specified as the segment type for *data-name-H* is invalid. `'EMI`Δ`'` must be specified for the last segment. `'ESI`Δ`'` must be specified for a segment other than the last segment.
`72028`	The value specified for *data-name-A* is invalid.
`72041`	When the logical message comprised a single segment, the `CBLDCMCF('EXECAP  ')` in which send segment length = 0 (with buffer format 1) or send segment length = 4 or less (with buffer format 2) was specified was called.
`72044`	From a continuous-inquiry-response type (`type=cont`) MHP that called `CBLDCMCF('CONTEND ')`, another continuous-inquiry-response type MHP was started by `CBLDCMCF('EXECAP  ')`.
`72108`	The value specified for *data-name-L* exceeds the limit.
`72109`	An attempt was made to activate an MHP, for which `type=cont` (continuous-inquiry-response type) was specified, by `CBLDCMCF('EXECAP  ')` with timer start specified.
`77001`	The logical terminal (LE) corresponding to the application to be activated is being started and cannot be used, or no logical terminals are available.
Other than the above	An unprecedented error (e.g., program damage) occurred.

Notes

1. The activation order of application programs varies depending on the `mcfmuap -c order` specification in the UAP common definition of the MCF manager definition.
2. If you use a single service function to update a TAM or DAM file and call the function `CBLDCMCF('EXECAP  ')` to start an application that will reference the updated file, make sure that the application will lock the file. If the application references the file without locking the file, the data existing before the file was updated might be referenced.

CBLDCMCF('MAINLOOP') - Start an MHP service

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(8) VALUE 'MAINLOOP'.
    02  data-name-B     PIC X(5).
    02  FILLER          PIC X(3).
    02  data-name-C     PIC X(16).
```

Description

`CBLDCMCF('MAINLOOP')` accepts scheduling to a service program corresponding to the application name. Call `CBLDCMCF('MAINLOOP')` only once in the process from the MHP main program.

`CBLDCMCF('MAINLOOP')` does not return until it receives a termination request for the MHP from OpenTP1.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'MAINLOOP'` for the request code indicating MHP service start.

- *data-name-C*

  Specify `LOW-VALUE`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	The program received a termination request for the MHP from OpenTP1. The MHP must immediately execute termination processing for its process. Then the MHP must call `CBLDCMCF('CLOSE   ')` and `CBLDCRPC('CLOSE   ')` to exit the main program with `STOP RUN`.
`70900`	The value specified for a data-name is invalid.

Status code	Explanation
`70901`	`CBLDCRPC('OPEN    ')` was not called before `CBLDCMCF('MAINLOOP')`.
`70902`	The service could not be started.
`70903`	The local memory became insufficient.
`72016`	The value specified for *data-name-C* is invalid.
`72028`	The value specified for *data-name-A* is invalid.

CBLDCMCF('OPEN   ') - Open the MCF environment

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'OPEN    '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP.
    02  data-name-D    PIC X(12).
```

Description

`CBLDCMCF('OPEN    ')` constructs the environment in which MCF facilities are used. Call `CBLDCMCF('OPEN    ')` for UAPs which use MCF facilities (CBLDCMCF).

After `CBLDCRPC('OPEN    ')` is called, `CBLDCMCF('OPEN    ')` must be called in the main program. Call `CBLDCMCF('OPEN    ')` only once in the process before `CBLDCMCF('MAINLOOP')` (`CBLDCRSV('MAINLOOP')` for an SPP). The following figure shows when to call `CBLDCMCF('OPEN    ')`:

```
CALL 'CBLDCRPC' (OPEN)
CALL 'CBLDCMCF' (OPEN)
CALL 'CBLDCMCF' (MAINLOOP)  (CALL 'CBLDCRSV' (MAINLOOP) for an SPP)
    :
    :
CALL 'CBLDCMCF' (CLOSE)
CALL 'CBLDCRPC' (CLOSE)
```

Data area whose value is set in the UAP

- *data-name-A*

  Specify `'VALUE 'OPEN△△△△'` for the request code indicating opening of the MCF environment.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-C*

  Specify `0`.

- *data-name-D*

  Specify `LOW-VALUE`.

Status codes

Status code	Explanation
`00000`	Normal termination.
`70900`	The value specified for *data-name-C* is invalid.
`70901`	`CBLDCRPC('OPEN    ')` was not called.
	`CBLDCMCF('OPEN    ')` was called.
`70902`	Initialization processing was unsuccessful.
`70903`	The memory became insufficient.
`72016`	The value specified for *data-name-D* is invalid.
`72028`	The value specified for *data-name-A* is invalid.

CBLDCMCF('RECEIVE ') - Receive a message

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING unique-name-1  unique-name-2
                        unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8) VALUE 'RECEIVE '.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-C   PIC X(4).
    02  data-name-D   PIC X(4).
    02  data-name-E   PIC 9(8).
    02  data-name-F   PIC 9(8).
    02  data-name-G    PIC 9(9)  COMP.
    02  data-name-H   PIC X(4).
    02  data-name-I    PIC X(4).
    02  data-name-J    PIC X(4).
    02  data-name-K   PIC X(4).
    02  data-name-L   PIC X(8).
    02  data-name-M1  PIC X(4).
    02  data-name-M2  PIC X(8).
    02  data-name-M3  PIC X(4).
    02  data-name-M4  PIC 9(9)  COMP.
    02  data-name-M5  PIC 9(9)  COMP.
    02  data-name-M6  PIC X(1).
    02  data-name-M7  PIC X(1).
    02  data-name-N   PIC X(14).
01  unique-name-2.
    02  data-name-O   PIC X(4).
    02  data-name-P   PIC X(8).
    02  data-name-Q   PIC X(8).
    02  data-name-R   PIC X(8).
    02  data-name-T   PIC X(28).
01  unique-name-3.
    02  data-name-U   PIC 9(x)  COMP.
    02  data-name-V   PIC X(x).
    02  data-name-W   PIC X(n).
```

Description

`CBLDCMCF('RECEIVE ')` receives a segment of a message. When a whole logical message is received, call `CBLDCMCF('RECEIVE ')` as many times as there are segments.

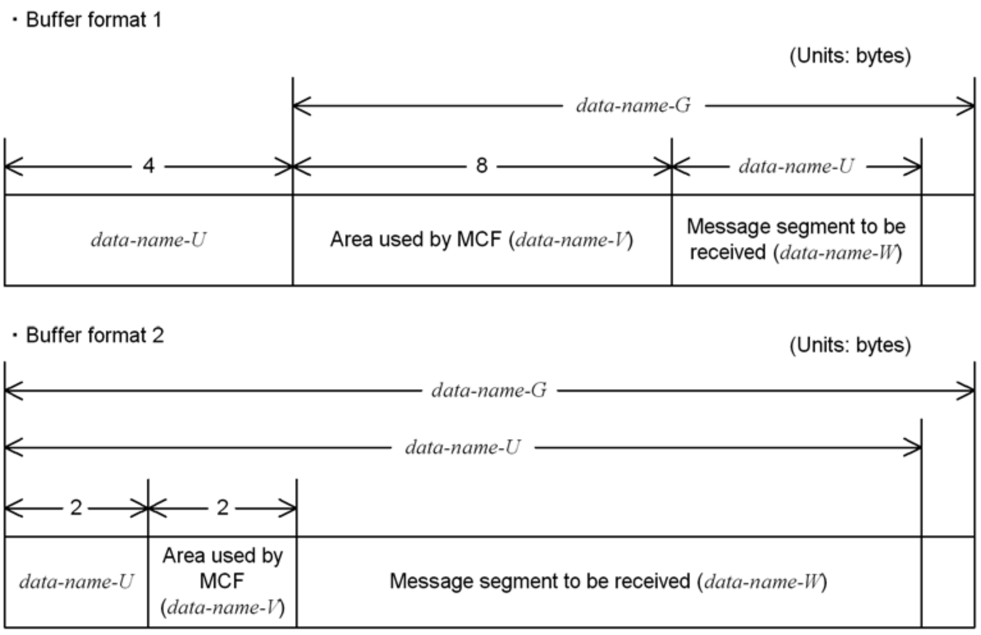
`CBLDCMCF('RECEIVE ')` can receive the following messages:

- Messages which are sent from the remote system via a protocol
- MCF events which are reported from the local system
- Messages which are sent by `CBLDCMCF('EXECAP  ')` (Activate an application program) from a UAP of the local system
- Messages which are sent by executing the `mcfuevt` command on the local system

When receiving a message which is sent from the remote system via a protocol, the syntax of `CBLDCMCF('RECEIVE ')` varies from one protocol to another. For the syntax of `CBLDCMCF('RECEIVE ')` which receives a message from the remote system, see the applicable *OpenTP1 Protocol* manual.

The maximum length of a single segment that can be received is 1 megabyte. Note that the actual value might be smaller depending on the protocol. For details, see the applicable *OpenTP1 Protocol* manual.

The figure below shows the format of the receive message area (area indicated with *unique-name-3*).



・Buffer format 1
(Units: bytes)
data-name-G
4
8
data-name-U
data-name-U
Area used by MCF (data-name-V)
Message segment to be received (data-name-W)
・Buffer format 2
(Units: bytes)
data-name-G
data-name-U
2
2
data-name-U
Area used by MCF (data-name-V)
Message segment to be received (data-name-W)

Data areas whose values are set in the UAP

- *data-name-A*

Specify `VALUE 'RECEIVE△'` for the request code indicating that the message was received.

- *data-name-C*

  Specify whether the first segment of the message is received with one of the following:

  `'FRST'`: Specify `'FRST'` to receive the first segment. If the message comprises only a single segment, also specify `'FRST'`.

  `'SEG∆'`: Specify `'SEG∆'` to receive an intermediate segment or the last segment.

- *data-name-D*

  Specify space.

- *data-name-G*

  Specify the length of the receive segment area.

- *data-name-H, data-name-I, data-name-J, data-name-K, data-name-L, data-name-M1, data-name-M2, data-name-M3*

  Specify space.

- *data-name-M4, data-name-M5*

  Specify `0`.

- *data-name-M6*

  Specify space.

- *data-name-M7*

  Specify the buffer format to be used:

  `'1'`: Specify `'1'` to use buffer format 1. In general, buffer format 1 is used.

  `'2'`: Specify `'2'` to use buffer format 2.

  Space: `'1'` (buffer format 1) is assumed as default.

- *data-name-N*

  Specify `LOW-VALUE`.

- *data-name-O*

  Specify space.

- *data-name-P* [when an intermediate segment or the last segment is received]

  Specify the input logical terminal name. Specify the logical terminal name returned when the first segment is received.

- *data-name-Q*

  This area is used by the MCF.

- *data-name-R*

  Specify space.

- *data-name-T*

  Specify `LOW-VALUE`.

- *data-name-V*

  With buffer format 1: `PIC X(8)`

  With buffer format 2: `PIC X(2)`

  This area is used by the MCF.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-E*

  The date when the message is received is returned in the format of `YYYYMMDD` (where `YYYY` is year, `MM` is month, and `DD` is day).

- *data-name-F*

  The time when the message is received is returned in the format of `HHMMSS00` (where `HH` indicates hours, `MM` indicates minutes, `SS` indicates seconds, and `00` is fixed).

- *data-name-P* [when the first segment is received]

  The input logical terminal name is returned.

  Specify the returned logical terminal name for *data-name-P* when an intermediate segment or the last segment is received.

- *data-name-U*

  With buffer format 1: `PIC 9(9)`

  The length of the receive segment is returned.

  With buffer format 2: `PIC 9(4)`

  (The length of the receive segment + 4) is returned.

- *data-name-W*

  The contents of the receive segment are returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71000`	`CBLDCMCF('RECEIVE ')` for receiving the first segment was called more than once. To receive an intermediate segment or the last segment, call `CBLDCMCF('RECEIVE ')` with `'SEG '` specified for *data-name-C*.
`71001`	`CBLDCMCF('RECEIVE ')` for receiving the next segment was called after the last segment of the message was received. `CBLDCMCF('RECEIVE ')` called immediately before receives a message completely. If `CBLDCMCF('RECEIVE ')` is called again after this status code is returned, the status code `72000` is returned.
`71002`	An error occurred during input processing for the message queue.
	The message queue is in shutdown state.
`71108`	The local memory for processes became insufficient.
`72000`	Return at MHP execution: `BLDCMCF('RECEIVE ')` for receiving an intermediate segment or the last segment was called before `CBLDCMCF('RECEIVE ')` for receiving the first segment was called. To receive the first segment, call `CBLDCMCF('RECEIVE ')` with `FRST` specified for *data-name-C*. `CBLDCMCF('RECEIVE ')` was called again after the status code `71001` was returned.
	Return at SPP execution: `CBLDCMCF('RECEIVE ')` cannot be called from an SPP.
`72001`	The logical terminal name specified for *data-name-P* is invalid.
`72013`	A segment exceeding the length of the receive area was received. The excess portion was truncated.
	A segment exceeding 32,767 bytes was received in the case of buffer format 2. The excess portion was truncated.
`72016`	The value specified for *data-name-D* is invalid.
	The value specified for *data-name-N* or *data-name-T* is invalid.
	The value specified for *data-name-M7* is invalid.
`72024`	The value specified for *data-name-O* is invalid.
`72025`	The value specified for *data-name-C* is invalid.
`72028`	The value specified for *data-name-A* is invalid.
`72036`	The segment receive area is insufficient. Allocate an area of 9 bytes or more for buffer format 1; 5 bytes or more for buffer format 2.

Status code	Explanation
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCMCF('RECVSYNC') - Receive a synchronous message

Format

For details on the format, see the applicable *OpenTP1 Protocol* manual.

Description

`CBLDCMCF('RECVSYNC')` makes an active UAP receive a logical message from other system.

The maximum length of a single segment that can be received is 1 megabyte. Note that the actual value might be smaller depending on the protocol. For details, see the applicable *OpenTP1 Protocol* manual.

The values to be set in the data areas and the status codes vary from one protocol to another. For details, see the applicable *OpenTP1 Protocol* manual.

CBLDCMCF('REPLY   ') - Send a response message

Format

For details on the format, see the applicable *OpenTP1 Protocol* manual.

Description

`CBLDCMCF('REPLY   ')` sends a logical message as a response to other system. The response message can be sent only by MHPs whose application type is ans or cont.

The maximum length of a single message segment that can be sent is 32 kilobytes. Note that the actual value might be smaller depending on the protocol. For details, see the applicable *OpenTP1 Protocol* manual.

The values to be set in the data areas and the status codes vary from one protocol to another. For details, see the applicable *OpenTP1 Protocol* manual.

CBLDCMCF('RESEND ') - Resend a message

Format

For details on the format, see the applicable *OpenTP1 Protocol* manual.

Description

`CBLDCMCF('RESEND  ')` resends an already sent logical message to other system. The resent message is treated as a new message separate from the already sent message.

The values to be set in the data areas and the status codes vary from one protocol to another. For details, see the applicable *OpenTP1 Protocol* manual.

Note

The message resend order varies depending on the `mcfmuap -c order` specification in the UAP common definition of the MCF manager definition.

CBLDCMCF('ROLLBACK') - Enable MHP rollback

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1
    02  data-name-A    PIC X(8) VALUE 'ROLLBACK'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(4).
    02  data-name-D    PIC X(12).
```

Description

`CBLDCMCF('ROLLBACK')` cancels processing between when the MHP service program that defines the transaction attribute is started and when the rollback is required. If `'RTRY'` is specified for *data-name-C*, processing between when the MHP is started and when the rollback is called is canceled, and the canceled MHP processing is rescheduled.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'ROLLBACK'` for the request code indicating partial recovery.

- *data-name-C*

  Specify the type of rollback. The following values are available:

  `'RTRY'`: Processing between when the MHP is started and when `CBLDCMCF('ROLLBACK')` is called is canceled, and the canceled MHP processing is rescheduled (any received messages are stored at the end of the relevant input queue and the MHP is rescheduled). Control does not return from the rollback statement, and the process is terminated.

  `'RTN∆'`: Processing between the MHP is started and when the `CBLDCMCF('ROLLBACK')` is called is canceled, and control returns. Processing after the normal return of the rollback with `'RTN∆'` specified is treated as another transaction.

  `'NRTN'`: Processing between the MHP is started and when the `CBLDCMCF('ROLLBACK')` is called is canceled. Control does not return from `CBLDCMCF('ROLLBACK')`, and the process is terminated.

- *data-name-D*

  Specify `LOW-VALUE`.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status Code	Explanation
`00000`	Normal termination.
`72000`	<Return at MHP execution> The MHP called `CBLDCMCF('ROLLBACK')` with `'RTN` Δ `'` specified for *data-name-C* before `CBLDCMCF('RECEIVE')` for receiving the first segment. `CBLDCMCF('ROLLBACK')` was called by an MHP with the nontransaction attribute.
	<Return at SPP execution> `CBLDCMCF('ROLLBACK')` cannot be called from an SPP.
`72016`	The value specified for *data-name-D* is invalid.
`72027`	The value specified for *data-name-C* (type of rollback) is invalid.
`72028`	The value specified for *data-name-A* is invalid.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCMCF('SEND   ') - Send a message

Format

For details on the format, see the applicable *OpenTP1 Protocol* manual.

Description

`CBLDCMCF('SEND    ')` sends a logical message to other system.

The maximum length of a single message segment that can be sent is 32 kilobytes. Note that the actual value might be smaller depending on the protocol. For details, see the applicable *OpenTP1 Protocol* manual.

The values to be set in the data areas and the status codes vary from one protocol to another. For details, see the applicable *OpenTP1 Protocol* manual.

Notes

`CBLDCMCF('SEND    ')` registers messages in the OTQ in the same order in which the messages were issued. The messages will be processed on a first-in, first-out basis at each logical terminal. If a message is at the start of the OTQ of a particular logical terminal and the corresponding transaction has been committed (for a non-transaction MHP, when the service function has returned a value), the message is fetched from the OTQ and processed.

If multiple UAPs issue `CBLDCMCF('SEND    ')` to the same logical terminal at the same time, the first message in the OTQ will be processed first, provided that it satisfies the above conditions. The subsequent messages in the OTQ registered by other UAPs will not be processed until the first message is processed. Therefore, if the service function does not return after `CBLDCMCF('SEND    ')` is issued, or if `CBLDCMCF('COMMIT ')` has not been issued, message transmission (or application activation) from other UAPs to the same logical terminal will be delayed. Be careful not to cause delays from `CBLDCMCF('SEND    ')` issuance to service function return or `CBLDCMCF('COMMIT ')` issuance.

Note

The message send order varies depending on the `mcfmuap -c order` specification in the UAP common definition of the MCF manager definition.

CBLDCMCF('SENDRECV') - Exchange a synchronous message

Format

For details on the format, see the applicable *OpenTP1 Protocol* manual.

Description

`CBLDCMCF('SENDRECV')` sends a logical message from a UAP to the remote system and receives a response. The UAP waits during the period from the time the logical message is sent to the time a response is received. Upon receiving a response, it begins subsequent processing.

The maximum length of a single segment that can be received is 1 megabyte. Note that the actual value might be smaller depending on the protocol. The maximum length of a single message segment that can be sent is 32 kilobytes. Note that the actual value might be smaller depending on the protocol. For details, see the applicable *OpenTP1 Protocol* manual.

The values to be set in the data areas and the status codes vary from one protocol to another. For details, see the applicable *OpenTP1 Protocol* manual.

CBLDCMCF('SENDSYNC') - Send a synchronous message

Format

For details on the format, see the applicable *OpenTP1 Protocol* manual.

Description

`CBLDCMCF('SENDSYNC')` sends a logical message from an active UAP to other system. The UAP waits until the MCF finishes sending the message. Upon completing the sending, it begins subsequent processing.

The maximum length of a single message segment that can be sent is 32 kilobytes. Note that the actual value might be smaller depending on the protocol. For details, see the applicable *OpenTP1 Protocol* manual.

The values to be set in the data areas and the status codes vary from one protocol to another. For details, see the applicable *OpenTP1 Protocol* manual.

CBLDCMCF('TACTCN ') - Establish connection

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8)  VALUE 'TACTCN  '.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-C   PIC X(4).
    02  data-name-D1  PIC X(1) VALUE SPACE.
    02  data-name-D2  PIC X(1).
    02  data-name-D3  PIC X(26) VALUE SPACE.
    02  data-name-E   PIC 9(9)  COMP.
    02  data-name-F1  PIC X(8).
    02  data-name-F2  PIC X(56) VALUE SPACE.
    02  data-name-G   PIC X(8)  VALUE SPACE.
    02  data-name-H   PIC X(8)  VALUE SPACE.
    02  data-name-I   PIC X(144) VALUE SPACE.
    02  data-name-J   PIC X(184) VALUE SPACE.
    02  data-name-K1  PIC 9(9)  COMP.
    02  data-name-K2  PIC X(n).
01  unique-name-2.
    02  data-name-L   PIC 9(9)  COMP VALUE ZERO.
```

Description

`CBLDCMCF('TACTCN  ')` establishes connection.

Normal termination of `CBLDCMCF('TACTCN  ')` indicates that the connection establishment request was accepted successfully by the protocol product. However, this does not necessarily mean that connection with the remote system has been established.

If you intend to perform any connection-related operation after calling `CBLDCMCF('TACTCN  ')`, first use `CBLDCMCF('TLSCN   ')` to check the connection status.

Data areas whose values are set in the UAP

- *data-name-A*

Specify `VALUE 'TACTCN△△'` for the request code indicating connection establishment.

- *data-name-C*

  Specify how to specify the connection that is to be established.

  `'LE△△'`

    Specifies that a logical terminal name is specified for the connection that is to be established. This argument is not supported by TP1/NET/NCSB or TP1/NET/X25-Extended.

  `'CN△△'`

    Specifies that a connection ID is specified for the connection that is to be established.

  Space

    `'LE△△'` (specification of a logical terminal name) is assumed.

- *data-name-D1*

  Specify a space.

- *data-name-D2*

  Specifies whether functionality that depends on the communication protocol is being used.

  `'1'`

    Specifies that functionality that depends on the communication protocol is being used.

  `'0'`

    Specifies that functionality that depends on the communication protocol is not being used.

  Space

    `'0'` (functionality that depends on the communication protocol is not being used) is assumed.

- *data-name-D3*

  Specify a space.

- *data-name-E*

  Specify the MCF communication process identifier of the MCF communication service for the connection that is to be processed. The permitted value range is from 0 to 239.

  This argument is ignored when a logical terminal name is used to request connection establishment.

If you specify `0`, the system searches for the MCF communication service to which the specified connection ID belongs. In a configuration where many MCF communication services are running or when you issue this function many times from a UAP, we recommend that you specify the MCF communication process identifier.

- *data-name-F1*

  Specify the logical terminal name or connection ID of the connection that is to be established. Express the logical terminal name or connection ID as a maximum of 8 bytes. If the specified name is shorter than 8 bytes, pad the name with trailing spaces.

- *data-name-F2*, *data-name-G*, *data-name-H*, *data-name-I*, *data-name-J*

  Specify a space.

- *data-name-K1*

  Specify the length of a protocol-specific area. You can specify a maximum of 1024 bytes.

  If you do not use functionality that depends on the communication protocol, specify `0`.

- *data-name-K2*

  Specify contents for the protocol-specific area.

  The permitted value depends on the communication protocol being used. For details, see the applicable *OpenTP1 Protocol* manual.

- *data-name-L*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71001`	`CBLDCMCF('TACTCN  ')` cannot be accepted because MCF is under start processing.
`71002`	`CBLDCMCF('TACTCN  ')` cannot be accepted because MCF is under termination processing.
`71004`	A memory shortage occurred during `CBLDCMCF('TACTCN  ')` processing.
`71005`	A communication error occurred. For the cause, see the message log file.

Status code	Explanation
71006	An internal error occurred. For the cause, see the message log file.
71007	The specified connection name has not been registered.
71008	The specified logical terminal name has not been registered.
71009	`CBLDCMCF('TACTCN  ')` is not supported by the applicable communication process.
71010	Although the request to establish connection was issued to the MCF communication process, the request was not accepted. For the cause, see the message log file.
71011	`CBLDCMCF('TACTCN  ')` cannot be accepted because the connection has been deleted.
71014	The specified logical terminal name belongs to TP1/NET/NCSB or TP1/NET/X25-Extended; or, the specified connection group name belongs to TP1/NET/OSI-TP or TP1/NET/TCP/IP.
72028	The value specified for *data-name-A* is invalid.
72052	<If `'0'` or a space is specified for *data-name-D2*> A nonzero value is specified for *data-name-K1*.
	<If `'1'` is specified for *data-name-D2*> A value smaller than `0` or greater than `1024` is specified for *data-name-K1*.
72053	A nonzero value is specified for *data-name-L*.
72058	The value specified for *data-name-C* is not `'LE`ΔΔ`'`, `CN`ΔΔ`'`, or a space.
72059	The value specified for *data-name-D1* or *data-name-D3* is not a space.
	The value specified for *data-name-D2* is not `1`, `0`, or a space.
72061	A value smaller than `0` or greater than `239` is specified for *data-name-E*.
72063	A space is specified for *data-name-F1*.
72065	The value specified for *data-name-F2* is not a space.
72066	The value specified for *data-name-G* is not a space.
72068	The value specified for *data-name-H* is not a space.
72070	The value specified for *data-name-I* is not a space.
72072	The value specified for *data-name-J* is not a space.
72074	The character string specified for *data-name-F1* contains an invalid character.

CBLDCMCF('TACTLE ') - Release a logical terminal from shutdown status

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1   unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8)  VALUE 'TACTLE  '.
    02  data-name-B    PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-C    PIC X(4) VALUE SPACE.
    02  data-name-D1   PIC X(1) VALUE SPACE.
    02  data-name-D2   PIC X(1).
    02  data-name-D3   PIC X(26) VALUE SPACE.
    02  data-name-E    PIC 9(9)  COMP.
    02  data-name-F1   PIC X(8).
    02  data-name-F2   PIC X(56) VALUE SPACE.
    02  data-name-G    PIC X(8)  VALUE SPACE.
    02  data-name-H    PIC X(8)  VALUE SPACE.
    02  data-name-I    PIC X(144) VALUE SPACE.
    02  data-name-J    PIC X(184) VALUE SPACE.
    02  data-name-K1   PIC 9(9)  COMP.
    02  data-name-K2   PIC X(n).
01  unique-name-2.
    02  data-name-L    PIC 9(9)  COMP VALUE ZERO.
```

Description

`CBLDCMCF('TACTLE  ')` releases a logical terminal from shutdown status.

Normal termination of `CBLDCMCF('TACTLE  ')` indicates that the logical terminal shutdown release request was accepted successfully by the protocol product. However, this does not necessarily mean that the logical terminal has been released from the shutdown status.

If you intend to perform any operation related to the logical terminal after calling `CBLDCMCF('TACTLE  ')`, first use `CBLDCMCF('TLSLE  ')` to check the logical terminal's status.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TACTLE△△'` for the request code indicating release of a logical terminal from shutdown status.

- *data-name-C, data-name-D1*

  Specify a space.

- *data-name-D2*

  Specifies whether functionality that depends on the communication protocol is being used.

  `'1'`

  Specifies that functionality that depends on the communication protocol is being used.

  `'0'`

  Specifies that functionality that depends on the communication protocol is not being used.

  Space

  `'0'` (functionality that depends on the communication protocol is not being used) is assumed.

- *data-name-D3*

  Specify a space.

- *data-name-E*

  Specify the MCF communication process identifier of the MCF communication service that has the logical terminal to be processed. The permitted value range is from 0 to 239.

  If you specify `0`, the system searches for the MCF communication service to which the specified connection ID belongs. In a configuration where many MCF communication services are running or when you issue this function many times from a UAP, we recommend that you specify the MCF communication process identifier.

- *data-name-F1*

  Specify the name of the logical terminal that is to be released from shutdown status. Express the logical terminal name as a maximum of 8 bytes. If the specified name is shorter than 8 bytes, pad the name with trailing spaces.

- *data-name-F2, data-name-G, data-name-H, data-name-I, data-name-J*

  Specify a space.

- *data-name-K1*

  Specify the length of a protocol-specific area. You can specify a maximum of 1024 bytes.

  If you do not use functionality that depends on the communication protocol, specify `0`.

- *data-name-K2*

  Specify contents for the protocol-specific area.

  The permitted value depends on the communication protocol being used. For details, see the applicable *OpenTP1 Protocol* manual.

- *data-name-L*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71001`	`CBLDCMCF('TACTLE  ')` cannot be accepted because MCF is under start processing.
`71002`	`CBLDCMCF('TACTLE  ')` cannot be accepted because MCF is under termination processing.
`71004`	A memory shortage occurred during `CBLDCMCF('TACTLE  ')` processing.
`71005`	A communication error occurred. For the cause, see the message log file.
`71006`	An internal error occurred. For the cause, see the message log file.
`71008`	The specified logical terminal name has not been registered.
`71009`	`CBLDCMCF('TACTLE  ')` is not supported by the applicable communication process.
`71010`	Although the request to release the logical terminal from shutdown status was issued to the MCF communication process, the request was not accepted. For the cause, see the message log file.
`71011`	`CBLDCMCF('TACTLE  ')` cannot be accepted because the logical terminal has been deleted.
`72028`	The value specified for *data-name-A* is invalid.

Status code	Explanation
`72052`	<If `'0'` or a space is specified for *data-name-D2*> A nonzero value is specified for *data-name-K1*.
	<If `'1'` is specified for *data-name-D2*> A value smaller than `0` or greater than `1024` is specified for *data-name-K1*.
`72053`	A nonzero value is specified for *data-name-L*.
`72058`	The value specified for *data-name-C* is not a space.
`72059`	The value specified for *data-name-D1* or *data-name-D3* is not a space.
	The value specified for *data-name-D2* is not `1`, `0`, or a space.
`72061`	A value smaller than `0` or greater than `239` is specified for *data-name-E*.
`72063`	*data-name-F1* begins with a space.
`72065`	The value specified for *data-name-F2* is not a space.
`72066`	The value specified for *data-name-G* is not a space.
`72068`	The value specified for *data-name-H* is not a space.
`72070`	The value specified for *data-name-I* is not a space.
`72072`	The value specified for *data-name-J* is not a space.
`72074`	The character string specified for *data-name-F1* contains an invalid character.

CBLDCMCF('TDCTCN ') - Release connection

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8)  VALUE 'TDCTCN  '.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-C   PIC X(4).
    02  data-name-D1  PIC X(1).
    02  data-name-D2  PIC X(1).
    02  data-name-D3  PIC X(26) VALUE SPACE.
    02  data-name-E   PIC 9(9)  COMP.
    02  data-name-F1  PIC X(8).
    02  data-name-F2  PIC X(56) VALUE SPACE.
    02  data-name-G   PIC X(8)  VALUE SPACE.
    02  data-name-H   PIC X(8)  VALUE SPACE.
    02  data-name-I   PIC X(144) VALUE SPACE.
    02  data-name-J   PIC X(184) VALUE SPACE.
    02  data-name-K1  PIC 9(9)  COMP.
    02  data-name-K2  PIC X(n).
01  unique-name-2.
    02  data-name-L   PIC 9(9)  COMP VALUE ZERO.
```

Description

`CBLDCMCF('TDCTCN  ')` releases connection.

Normal termination of `CBLDCMCF('TDCTCN  ')` indicates that the connection release request was accepted successfully by the protocol product. However, this does not necessarily mean that connection with the remote system has been released.

If you intend to perform any connection-related operation after calling `CBLDCMCF('TDCTCN  ')`, first use `CBLDCMCF('TLSCN   ')` to check the status of the connection.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TDCTCN`ΔΔ`'` for the request code indicating connection release.

- *data-name-C*

  Specify how to specify the connection that is to be released.

  `'LE△△'`

  Specifies that a logical terminal name is specified for the connection that is to be released. This argument is not supported by TP1/NET/NCSB or TP1/NET/X25-Extended.

  `'CN△△'`

  Specifies that a connection ID is specified for the connection that is to be released.

  Space

  `'LE△△'` (specification of a logical terminal name) is assumed.

- *data-name-D1*

  Specify whether the connection is to be released forcibly.

  `'1'`

  Release connection forcibly.

  `'0'`

  Release connection normally.

  Space

  `'0'` (normal release) is assumed.

- *data-name-D2*

  Specifies whether functionality that depends on the communication protocol is being used.

  `'1'`

  Specifies that functionality that depends on the communication protocol is being used.

  `'0'`

  Specifies that functionality that depends on the communication protocol is not being used.

  Space

  `'0'` (functionality that depends on the communication protocol is not being used) is assumed.

- *data-name-D3*

  Specify a space.

- *data-name-E*

  Specify the MCF communication process identifier of the MCF communication service for the connection that is to be processed. The permitted value range is from 0 to 239.

  This argument is ignored when a logical terminal name is used to request connection release.

  If you specify `0`, the system searches for the MCF communication service to which the specified connection ID belongs. In a configuration where many MCF communication services are running or when you issue this function many times from a UAP, we recommend that you specify the MCF communication process identifier.

- *data-name-F1*

  Specify the logical terminal name or connection ID of the connection that is to be released. Express the logical terminal name or connection ID as a maximum of 8 bytes. If the specified name is shorter than 8 bytes, pad the name with trailing spaces.

- *data-name-F2*, *data-name-G*, *data-name-H*, *data-name-I*, *data-name-J*

  Specify a space.

- *data-name-K1*

  Specify the length of a protocol-specific area. You can specify a maximum of 1024 bytes.

  If you do not use functionality that depends on the communication protocol, specify `0`.

- *data-name-K2*

  Specify contents for the protocol-specific area.

  The permitted value depends on the communication protocol being used. For details, see the applicable *OpenTP1 Protocol* manual.

- *data-name-L*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.

Status code	Explanation
`71001`	`CBLDCMCF('TDCTCN  ')` cannot be accepted because MCF is under start processing.
`71002`	`CBLDCMCF('TDCTCN  ')` cannot be accepted because MCF is under termination processing.
`71004`	A memory shortage occurred during `CBLDCMCF('TDCTCN  ')` processing.
`71005`	A communication error occurred. For the cause, see the message log file.
`71006`	An internal error occurred. For the cause, see the message log file.
`71007`	The specified connection name has not been registered.
`71008`	The specified logical terminal name has not been registered.
`71009`	`CBLDCMCF('TDCTCN  ')` is not supported by the applicable communication process.
`71010`	Although the request to release the connection was issued to the MCF communication process, the request was not accepted. For the cause, see the message log file.
`71011`	`CBLDCMCF('TDCTCN  ')` cannot be accepted because the connection has been deleted.
`71014`	The specified logical terminal name belongs to TP1/NET/NCSB or TP1/NET/X25-Extended; or, the specified connection group name belongs to TP1/NET/OSI-TP or TP1/NET/TCP/IP.
`72028`	The value specified for *data-name-A* is invalid.
`72052`	<If `'0'` or a space is specified for *data-name-D2*> A nonzero value is specified for *data-name-K1*.
	<If `'1'` is specified for *data-name-D2*> A value smaller than `0` or greater than `1024` is specified for *data-name-K1*.
`72053`	A nonzero value is specified for *data-name-L*.
`72058`	The value specified for *data-name-C* is not `'LE`ΔΔ`'`, ΔΔ`'CN`ΔΔ`'`, or a space.
`72059`	The value specified for *data-name-D2* is not `1`, `0`, or a space.
	The value specified for *data-name-D3* is not a space.
`72061`	A value smaller than `0` or greater than `239` is specified for *data-name-E*.
`72063`	*data-name-F1* begins with a space.
`72065`	The value specified for *data-name-F2* is not a space.
`72066`	The value specified for *data-name-G* is not a space.
`72068`	The value specified for *data-name-H* is not a space.

Status code	Explanation
`72070`	The value specified for *data-name-I* is not a space.
`72072`	The value specified for *data-name-J* is not a space.
`72074`	The character string specified for *data-name-F1* contains an invalid character.
`72075`	The value specified for *data-name-D1* is not `1`, `0`, or a space.

CBLDCMCF('TDCTLE ') - Shut down a logical terminal

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1  unique-name-2
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8)  VALUE 'TDCTLE  '.
    02  data-name-B   PIC X(5).
    02  FILLER       PIC X(3).
    02  data-name-C   PIC X(4) VALUE SPACE.
    02  data-name-D1  PIC X(1) VALUE SPACE.
    02  data-name-D2  PIC X(1).
    02  data-name-D3  PIC X(26) VALUE SPACE.
    02  data-name-E   PIC 9(9)  COMP.
    02  data-name-F1  PIC X(8).
    02  data-name-F2  PIC X(56) VALUE SPACE.
    02  data-name-G   PIC X(8)  VALUE SPACE.
    02  data-name-H   PIC X(8)  VALUE SPACE.
    02  data-name-I   PIC X(144) VALUE SPACE.
    02  data-name-J   PIC X(184) VALUE SPACE.
    02  data-name-K1  PIC 9(9)  COMP.
    02  data-name-K2  PIC X(n).
01  unique-name-2.
    02  data-name-L   PIC 9(9)  COMP VALUE ZERO.
```

Description

`CBLDCMCF('TDCTLE  ')` shuts down a logical terminal.

Normal termination of `CBLDCMCF('TDCTLE  ')` indicates that the logical terminal shutdown request was accepted successfully by the protocol product. However, this does not necessarily mean that the logical terminal has been shut down.

If you intend to perform any operation related to the logical terminal after calling `CBLDCMCF('TDCTLE  ')`, first use `CBLDCMCF('TLSLE   ')` to check the logical terminal's status.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'TDCTLE`ΔΔ`'` for the request code indicating logical terminal shutdown.

- *data-name-C*, *data-name-D1*

  Specify a space.

- *data-name-D2*

  Specifies whether functionality that depends on the communication protocol is being used.

  `'1'`

  Specifies that functionality that depends on the communication protocol is being used.

  `'0'`

  Specifies that functionality that depends on the communication protocol is not being used.

  Space

  `'0'` (functionality that depends on the communication protocol is not being used) is assumed.

- *data-name-D3*

  Specify a space.

- *data-name-E*

  Specify the MCF communication process identifier of the MCF communication service that has the logical terminal to be processed. The permitted value range is from 0 to 239.

  If you specify `0`, the system searches for the MCF communication service to which the specified connection ID belongs. In a configuration where many MCF communication services are running or when you issue this function many times from a UAP, we recommend that you specify the MCF communication process identifier.

- *data-name-F1*

  Specify the name of the logical terminal that is to be shut down. Express the logical terminal name as a maximum of 8 bytes. If the specified name is shorter than 8 bytes, pad the name with trailing spaces.

- *data-name-F2*, *data-name-G*, *data-name-H*, *data-name-I*, *data-name-J*

  Specify a space.

- *data-name-K1*

  Specify the length of a protocol-specific area. You can specify a maximum of 1024 bytes.

  If you do not use functionality that depends on the communication protocol, specify `0`.

- *data-name-K2*

  Specify contents for the protocol-specific area.

  The permitted value depends on the communication protocol being used. For details, see the applicable *OpenTP1 Protocol* manual.

- *data-name-L*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71001`	`CBLDCMCF('TDCTLE  ')` cannot be accepted because MCF is under start processing.
`71002`	`CBLDCMCF('TDCTLE  ')` cannot be accepted because MCF is under termination processing.
`71004`	A memory shortage occurred during `CBLDCMCF('TDCTLE  ')` processing.
`71005`	A communication error occurred. For the cause, see the message log file.
`71006`	An internal error occurred. For the cause, see the message log file.
`71008`	The specified logical terminal name has not been registered.
`71009`	`CBLDCMCF('TDCTLE  ')` is not supported by the applicable communication process.
`71010`	Although the request to shut down the logical terminal was issued to the MCF communication process, the request was not accepted. For the cause, see the message log file.
`71011`	`CBLDCMCF('TDCTLE  ')` cannot be accepted because the logical terminal has been deleted.
`72028`	The value specified for *data-name-A* is invalid.
`72052`	<If `'0'` or a space is specified for *data-name-D2*> A nonzero value is specified for *data-name-K1*.
	<If `'1'` is specified for *data-name-D2*> A value smaller than `0` or greater than `1024` is specified for *data-name-K1*.
`72053`	A nonzero value is specified for *data-name-L*.

Status code	Explanation
`72058`	The value specified for *data-name-C* is not a space.
`72059`	The value specified for *data-name-D2* is not `1`, `0`, or a space.
	The value specified for *data-name-D3* is not a space.
`72061`	A value smaller than `0` or greater than `239` is specified for *data-name-E*.
`72063`	*data-name-F1* begins with a space.
`72065`	The value specified for *data-name-F2* is not a space.
`72066`	The value specified for *data-name-G* is not a space.
`72068`	The value specified for *data-name-H* is not a space.
`72070`	The value specified for *data-name-I* is not a space.
`72072`	The value specified for *data-name-J* is not a space.
`72074`	The character string specified for *data-name-F1* contains an invalid character.
`72075`	The value specified in *data-name-D1* is not a space.

CBLDCMCF('TDLQLE ') - Delete a logical terminal's output queue

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8)  VALUE 'TDLQLE  '.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-C   PIC X(4) VALUE SPACE.
    02  data-name-D   PIC X(28) VALUE SPACE.
    02  data-name-E   PIC 9(9)  COMP.
    02  data-name-F1  PIC X(8).
    02  data-name-F2  PIC X(56) VALUE SPACE.
    02  data-name-G   PIC X(8)  VALUE SPACE.
    02  data-name-H   PIC X(8)  VALUE SPACE.
    02  data-name-I   PIC X(144) VALUE SPACE.
    02  data-name-J   PIC X(184) VALUE SPACE.
    02  data-name-K   PIC 9(9)  COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-L   PIC 9(9)  COMP VALUE ZERO.
```

Description

`CBLDCMCF('TDLQLE  ')` deletes a logical terminal's output queue.

When the function deletes the output queue successfully, it sends an MCF event that reports discarding of an unprocessed send mesage (`ERREVTA`).

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TDLQLE`ΔΔ`'` for the request code indicating deletion of a logical teminal's output queue.

- *data-name-C*, *data-name-D*

  Specify a space.

- *data-name-E*

  Specify the MCF communication process identifier of the MCF communication service that has the logical terminal to be processed. The permitted value range is from

0 to 239.

If you specify `0`, the system searches for the MCF communication service to which the specified logical terminal name belongs. In a configuration where many MCF communication services are running or when you issue this function many times from a UAP, we recommend that you specify the MCF communication process identifier.

- *data-name-F1*

  Specify the name of the logical terminal whose output queue is to be deleted. Express the logical terminal name as a maximum of 8 bytes. If the specified name is shorter than 8 bytes, pad the name with trailing spaces.

- *data-name-F2*, *data-name-G*, *data-name-H*, *data-name-I*, *data-name-J*

  Specify a space.

- *data-name-K*, *data-name-L*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71001`	`CBLDCMCF('TDLQLE  ')` cannot be accepted because MCF is under start processing.
`71002`	`CBLDCMCF('TDLQLE  ')` cannot be accepted because MCF is under termination processing.
`71004`	A memory shortage occurred during `CBLDCMCF('TDLQLE  ')` processing.
`71005`	A communication error occurred. For the cause, see the message log file.
`71006`	An internal error occurred. For the cause, see the message log file.
`71008`	The specified logical terminal name has not been registered.
`71009`	`CBLDCMCF('TDLQLE  ')` is not supported by the applicable communication process.
`71010`	Although the request to delete the logical terminal's output queue was issued to the MCF communication process, the request was not accepted. For the cause, see the message log file.
`71011`	`CBLDCMCF('TDLQLE  ')` cannot be accepted because the logical terminal has been deleted.

Status code	Explanation
71017	`CBLDCMCF('TDLQLE  ')` cannot be accepted because the logical terminal has not been shut down.
71018	`CBLDCMCF('TDLQLE  ')` cannot be accepted because the session has not been closed.
71019	`CBLDCMCF('TDLQLE  ')` cannot be accepted because an alternate send operation is under way.
72028	The value specified for *data-name-A* is invalid.
72052	A nonzero value is specified for *data-name-K*.
72053	A nonzero value is specified for *data-name-L*.
72058	The value specified for *data-name-C* is not a space.
72059	The value specified for *data-name-D* is not a space.
72061	A value smaller than `0` or greater than `239` is specified for *data-name-E*.
72063	*data-name-F1* begins with a space.
72065	The value specified for *data-name-F2* is not a space.
72066	The value specified for *data-name-G* is not a space.
72068	The value specified for *data-name-H* is not a space.
72070	The value specified for *data-name-I* is not a space.
72072	The value specified for *data-name-J* is not a space.
72074	The character string specified for *data-name-F1* contains an invalid character.

CBLDCMCF('TEMPGET ') - Accept temporarily-stored data

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1  unique-name-2
                         unique-name-3
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'TEMPGET '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(4).
    02  data-name-D    PIC X(4).
    02  data-name-E    PIC 9(8).
    02  data-name-F    PIC 9(8).
    02  data-name-G    PIC 9(9) COMP.
    02  data-name-H    PIC X(4).
    02  data-name-I    PIC X(4).
    02  data-name-J    PIC X(4).
    02  data-name-K    PIC X(4).
    02  data-name-L    PIC X(8).
    02  data-name-M1   PIC X(4).
    02  data-name-M2   PIC X(8).
    02  data-name-M3   PIC X(4).
    02  data-name-M4   PIC 9(9) COMP.
    02  data-name-M5   PIC 9(9) COMP.
    02  data-name-M6   PIC X(1).
    02  data-name-M7   PIC X(1).
    02  data-name-N    PIC X(14).
01  unique-name-2.
    02  data-name-O    PIC X(4).
    02  data-name-P    PIC X(8).
    02  data-name-Q    PIC X(8).
    02  data-name-R    PIC X(8).
    02  data-name-S    PIC X(28).
01  unique-name-3.
    02  data-name-T    PIC 9(x) COMP.
    02  data-name-U    PIC X(x).
    02  data-name-V    PIC X(n).
```

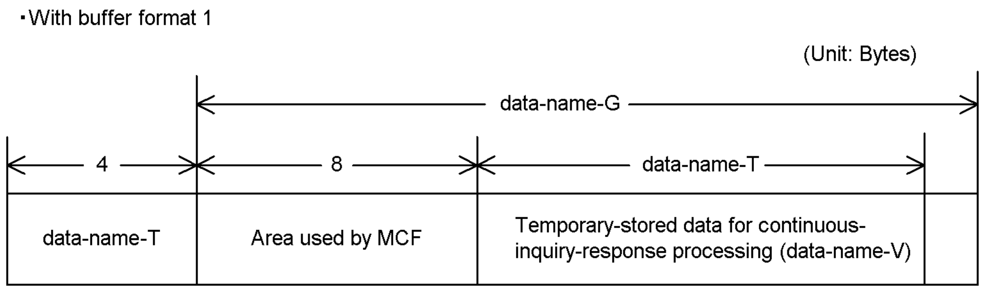
Description

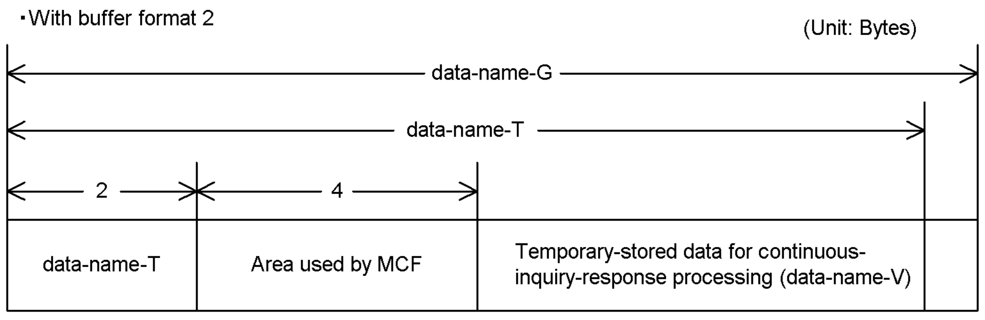
`CBLDCMCF('TEMPGET  ')` receives data stored in the temporary memory area.

For *data-name-G*, specify a value from 1 to 32,000 bytes. If the temporary-stored data exceeds the length specified for *data-name-G*, the excess portion is truncated. If the temporary-stored data is shorter than the length specified for *data-name-G*, no processing is done for the remaining receive area.

If there is no temporary-stored data, the temporary-stored data is received on the assumption that $(00)_{16}$ equivalent to the length specified for tempsize in the MCF application definition is specified.

The figure below shows the format of the receive segment area (indicated by *unique-name-3*).





Data area whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TEMPGET∆'` for the request code indicating to accept temporarily-stored data.

- *data-name-C, data-name-D*

  Specify space.

- *data-name-E, data-name-F*

  These areas are used by the MCF.

- *data-name-G*

  Specify the length of the area for receiving temporary-stored data. The number of bytes to be specified varies depending on the buffer format.

- *data-name-H, data-name-I, data-name-J, data-name-K, data-name-L, data-name-M1, data-name-M2, data-name-M3*

  Specify space.

- *data-name-M4, data-name-M5*

  Specify `0`.

- *data-name-M6*

  Specify space.

- *data-name-M7*

  Specify the buffer format to be used. The following values are available:

  `'1'`: Specify this value to use buffer format 1.

  `'2'`: Specify this value to use buffer format 2.

- *data-name-N*

  Specify `LOW-VALUE`.

- *data-name-O, data-name-P, data-name-Q, data-name-R*

  Specify space.

- *data-name-S*

  Specify `LOW-VALUE`.

- *data-name-U*

  With buffer format 1: `PIC X(8)`

  With buffer format 2: `PIC X(4)`

  This area is used by the MCF.

Data values whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-T*

  With buffer format 1: `PIC 9(9)`

  With buffer format 2: `PIC 9(4)`

  The length of the previously updated data is set.

- *data-name-V*

  The temporary-stored data is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`72000`	The temporary-stored data cannot be received from an SPP.
`72013`	Temporary-stored data exceeding the length of the receive area was received. The excess portion was truncated. With buffer format 2, the excess portion is truncated if the data exceeds 32,761 bytes.
`72016`	The value specified for *data-name-N* or *S* is invalid.
`72028`	The value specified for *data-name-A* is invalid.
`72036`	The receive area length specified for *data-name-G* is less than 9 bytes (with buffer format 1) or less than 7 bytes (with buffer format 2).
`72101`	`CBLDCMCF('TEMPGET ')` was called from an MHP for which `type=cont` (continuous-inquiry- response type) was not specified in the MCF application definition.
`72106`	`CBLDCMCF('TEMPGET ')` was called before the memory receive statement for receiving the first segment.
`72107`	`CBLDCMCF('TEMPGET ')` was called after a termination request for continuous-inquiry-response processing.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCMCF('TEMPPUT ') - Update temporarily-stored data

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1   unique-name-2
                         unique-name-3
```

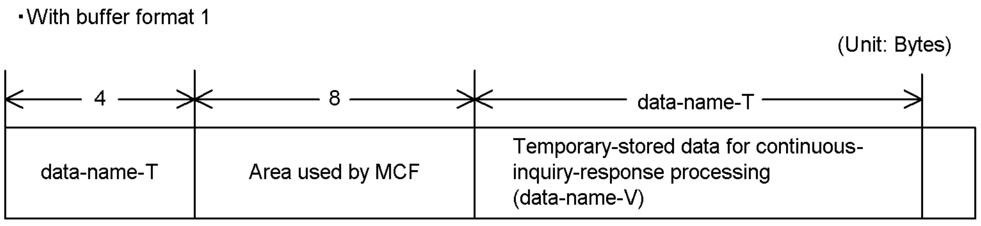
■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(8) VALUE 'TEMPPUT '.
    02  data-name-B     PIC X(5).
    02  FILLER          PIC X(3).
    02  data-name-C     PIC X(4).
    02  data-name-D     PIC X(4).
    02  data-name-E     PIC 9(8).
    02  data-name-F     PIC 9(8).
    02  data-name-G     PIC 9(9) COMP.
    02  data-name-H     PIC X(4).
    02  data-name-I     PIC X(4).
    02  data-name-J     PIC X(4).
    02  data-name-K     PIC X(4).
    02  data-name-L     PIC X(8).
    02  data-name-M1    PIC X(4).
    02  data-name-M2    PIC X(8).
    02  data-name-M3    PIC X(4).
    02  data-name-M4    PIC 9(9) COMP.
    02  data-name-M5    PIC 9(9) COMP.
    02  data-name-M6    PIC X(1).
    02  data-name-M7    PIC X(1).
    02  data-name-N     PIC X(14).
01  unique-name-2.
    02  data-name-O     PIC X(4).
    02  data-name-P     PIC X(8).
    02  data-name-Q     PIC X(8).
    02  data-name-R     PIC X(8).
    02  data-name-S     PIC X(28).
01  unique-name-3.
    02  data-name-T     PIC 9(x) COMP.
    02  data-name-U     PIC X(x).
    02  data-name-V     PIC X(n).
```

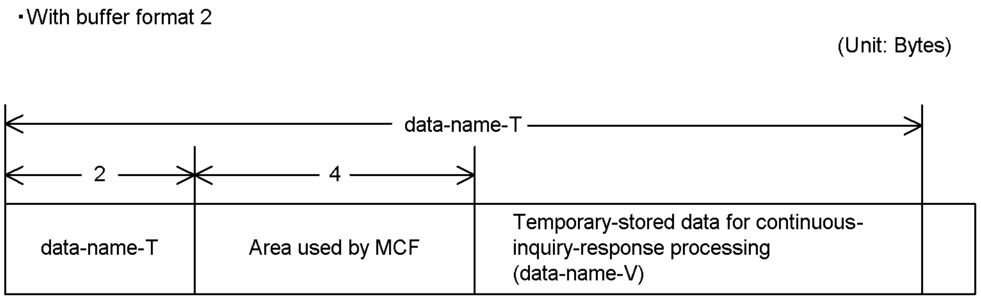
Description

`CBLDCMCF('TEMPPUT ')` updates data stored in the temporary memory area which is used for continuous-inquiry-response processing.

The figure below shows the format of the send segment area (indicated by *unique-name-3*).



・With buffer format 1
(Unit: Bytes)
4
8
data-name-T
data-name-T
Area used by MCF
Temporary-stored data for continuous-inquiry-response processing (data-name-V)



・With buffer format 2
(Unit: Bytes)
data-name-T
2
4
data-name-T
Area used by MCF
Temporary-stored data for continuous-inquiry-response processing (data-name-V)

Data area whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TEMPPUT∆'` for the request code indicating updating of temporarily-stored data.

- *data-name-C, data-name-D*

  Specify space.

- *data-name-E, data-name-F*

  These areas are used by the MCF.

- *data-name-G*

  Specify `0`.

- *data-name-H, data-name-I, data-name-J, data-name-K, data-name-L, data-name-M1, data-name-M2, data-name-M3*

  Specify space.

- *data-name-M4, data-name-M5*

  Specify `0`.

- *data-name-M6*

  Specify space.

- *data-name-M7*

  Specify the buffer format to be used. The following values are available:

  `'1'`: Specify this value to use buffer format 1.

  `'2'`: Specify this value to use buffer format 2.

- *data-name-N*

  Specify `LOW-VALUE`.

- *data-name-O, data-name-P, data-name-Q, data-name-R*

  Specify space.

- *data-name-S*

  Specify `LOW-VALUE`.

- *data-name-T*

  With buffer format 1: `PIC 9(9)`

  With buffer format 2: `PIC 9(4)`

  Specify the length of the temporary-stored data to be updated.

- *data-name-U*

  With buffer format 1: `PIC X(8)`

  With buffer format 2: `PIC X(4)`

  This area is used by the MCF.

- *data-name-V*

  Specify the area storing the temporary-stored data to be updated.

Data value whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71103`	The area for updating the temporary-stored data could not be acquired.
`72000`	The temporary-stored data cannot be updated from an SPP.
`72016`	The value specified for *data-name-N* or *S* is invalid.

Status code	Explanation
`72028`	The value specified for *data-name-A* is invalid.
`72035`	The value specified for the data update length (*data-name-T*) exceeds the value specified for the temporary-stored data, area for storing length in the MCF application definition.
	Alternatively, the value specified for the data update length is less than 1 byte (with buffer format 1) or less than 7 bytes (with buffer format 2).
`72101`	`CBLDCMCF('TEMPPUT ')` was called from an MHP for which `type=cont` (continuous-inquiry-response type) was not specified in the MCF application definition.
`72105`	`CBLDCMCF('TEMPPUT ')` was called before `CBLDCMCF('TEMPGET')`
`72106`	`CBLDCMCF('TEMPPUT ')` was called before `CBLDCMCF('RECEIVE')` for receiving the first segment.
`72107`	`CBLDCMCF('TEMPPUT ')` was called after `CBLDCMCF('CONTEND ')`.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCMCF('TIMERCAN') - Cancel user timer monitoring

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCMCF' USING unique-name-1 unique-name-2.
```

- DATA DIVISION specification

```
01 unique-name-1.
   02 data-name-A  PIC X(8) VALUE 'TIMERCAN'.
   02 data-name-B  PIC X(5).
   02 FILLER       PIC X(3).
   02 data-name-C  PIC X(16).
01 unique-name-2.
   02 data-name-D  PIC 9(9) COMP.
   02 data-name-E  PIC 9(9) COMP.
   02 data-name-F  PIC X(16).
   02 data-name-G  PIC X(8).
   02 data-name-H  PIC X(8).
   02 data-name-I  PIC X(16).
```

Description

`CBLDCMCF('TIMERCAN')` cancels the user timer monitor that was set by `CBLDCMCF('TIMERSET')`.

This program cancels the user timer monitor as soon as `CBLDCMCF('TIMERCAN')` returns normally.

If the user timer monitor has reached timeout and an MHP has already been started at the time `CBLDCMCF('TIMERCAN')` is called, `CBLDCMCF('TIMERCAN')` returns with the error 70910.

Only a user server can call `CBLDCMCF('TIMERCAN')`.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TIMERCAN'` for the request code indicating cancellation of user timer monitoring.

- *data-name-C*

  Specify `LOW-VALUE`.

- *data-name-D*

  Specify `0`.

- *data-name-E*

  Specify the same timer request identifier as that specified when the user timer monitor was set by `CBLDCMCF('TIMERSET')`.

- *data-name-F*

  Specify `LOW-VALUE`.

- *data-name-G*

  Specify the same logical terminal name as that specified when the user timer monitor was set by `CBLDCMCF('TIMERSET')`. When specifying a name consisting of fewer than 8 characters, pad the name by entering spaces after the name.

- *data-name-H*

  Specify `LOW-VALUE`.

- *data-name-I*

  Specify `LOW-VALUE`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`72016`	The value specified for *data-name-C* is invalid.
	The value specified for *data-name-I* is invalid.
`72028`	The value specified for *data-name-A* is invalid.
`70910`	The value specified for *data-name-E* is invalid.
	The timer request identifier specified for *data-name-E* is not registered.
	Timeout already occurred and the application has started, or the user timer monitor was already canceled.
`70911`	The value specified for *data-name-F* is invalid.
`70912`	The value specified for *data-name-G* is invalid.
`70916`	The requested facility is not defined in the MCF.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCMCF('TIMERSET') - Set user timer monitoring

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF' USING unique-name-1  unique-name-2
                       unique-name-3.
```

- DATA DIVISION specification

```
01 unique-name-1.
   02 data-name-A  PIC X(8) VALUE 'TIMERSET'.
   02 data-name-B  PIC X(5).
   02 FILLER     PIC X(3).
   02 data-name-C  PIC X(16).
01 unique-name-2.
   02 data-name-D  PIC 9(9) COMP.
   02 data-name-E  PIC 9(9) COMP.
   02 data-name-F  PIC X(16).
   02 data-name-G  PIC X(8).
   02 data-name-H  PIC X(8).
   02 data-name-I   PIC X(16).
01 unique-name-3.
   02 data-name-J   PIC 9(9) COMP.
   02 data-name-K  PIC X(n).
```

Description

Use `CBLDCMCF('TIMERSET')` from a UAP to set the user timer monitor for monitoring the desired interval. To call `CBLDCMCF('TIMERSET')`, you must specify `usertime=yes` in the `-p` option of the MCF communication configuration definition `mcfttim`.

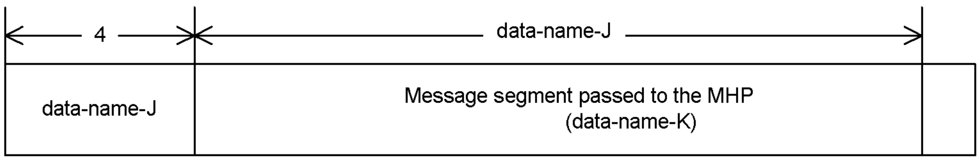
Only a user server can call `CBLDCMCF('TIMERSET')`.

When the time (in seconds) specified for *data-name-D* elapses (when timeout occurs), the logical terminal specified for *data-name-G* generates an event and starts the MHP having the application name specified for *data-name-H*.

The MHP to be started when timeout occurs must be a non-response-type (noans type) MHP. The figure below shows the format of the area indicated by *unique-name-3* and the segment area passed to the MHP when a message is passed to the MHP.

・Area indicated by unique-name-3

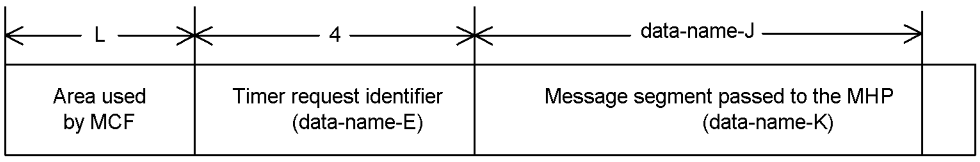
(Unit: Bytes)



・Segment area passed to the MHP

With buffer format 1, L is 8 bytes; with buffer format 2, L is 4 bytes.

(Unit: Bytes)


L
4
data-name-J
Area used by MCF
Timer request identifier (data-name-E)
Message segment passed to the MHP (data-name-K)

To cancel the user timer monitor set by `CBLDCMCF('TIMERSET')`, call the `CBLDCMCF('TIMERCAN')` function with the same values specified for *data-name-E* and *data-name-G* as specified in `CBLDCMCF('TIMERSET')`.

The time monitor starts as soon as `CBLDCMCF('TIMERSET')` is called.

The maximum number of time monitors you can run concurrently is indicated by the maximum number of time monitoring requests specified for the `timereqno` operand in the `-p` option of the MCF communication configuration definition `mcfttim`.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'TIMERSET'` for the request code indicating specification of user timer monitoring.

■ *data-name-C*

Specify `LOW-VALUE`.

■ *data-name-D*

Specify the number of seconds that are to elapse before the MHP is started after `CBLDCMCF('TIMERSET')` is called. The specifiable range is 1 to 360000 (from 1 second to 100 hours).

Since OpenTP1 monitors timeout at fixed intervals, an error arises between the time specified for *data-name-D* and the time that elapses before actual detection of timeout. The accuracy of time monitoring depends on the value of the interval of time monitoring specified for the `btim` operand in the `-t` option of the MCF communication configuration definition `mcfttim`.

- *data-name-E*

  Specify the timer request identifier.

  data-name-E provides information for identifying this timer. Be sure to specify a value for *data-name-E* that is unique in the logical terminal specified for *data-name-G*.

- *data-name-F*

  Specify `LOW-VALUE`.

- *data-name-G*

  Specify the name of the logical terminal that is to generate an event when timeout occurs. When specifying a name consisting of fewer than 8 characters, pad the name by entering spaces after the name.

- *data-name-H*

  Specify the application name of the MHP to be started. Define the attribute of the application in the application attribute definition (`mcfaalcap`) in the MCF application definition specified for the `-a` option of the MCF communication configuration definition `mcftenv` of the MCF communication server having the logical terminal name specified for *data-name-G*. When specifying a name consisting of fewer than 8 characters, pad the name by entering spaces after the name. The MHP must be a non-response-type (noans type) MHP. The specified application name must be a user event.

- *data-name-I*

  Specify `LOW-VALUE`.

- *data-name-J*

  Specify the length of the message segment to be passed to the MHP to be started. If no segment is to be passed to the MHP to be started, specify `0`. The specifiable range is 0 to 256. The maximum specifiable value depends on the maximum message length specified for the `msgsize` operand in the -p option of the MCF communication configuration definition `mcfttim`.

- *data-name-K*

  Specify the contents of the message segment to be passed to the MHP to be started. You cannot specify multiple segments. If no segment is to be passed to the MHP to be started, specify a null character.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`72016`	The value specified for *data-name-C* is invalid.
`72028`	The value specified for *data-name-A* is invalid.
`70900`	The value specified for *data-name-I* is invalid.
`70909`	The value specified for *data-name-D* is invalid.
`70910`	The value specified for *data-name-E* is invalid.
	The specified timer request identifier is already registered.
`70911`	The value specified for *data-name-F* is invalid.
`70912`	The value specified for *data-name-G* is invalid.
`70913`	The value specified for *data-name-H* is invalid.
`70914`	The value specified for *data-name-J* is invalid.
`70915`	The value specified for *data-name-K* is invalid.
`70916`	The requested facility is not defined in the MCF.
`70917`	User timer monitoring cannot be configured because there is no free space in the timer registration area. To obtain the necessary space in the timer registration area, change the value assigned to the `timereqno` operand of the `-p` option in the MCF communication configuration definition `mcfttim`. If necessary, also verify the value assigned to the `-p` option of the MCF manager definition `mcfmcomn` and the value assigned to the `static_shmpool_size` operand in the system environment definition.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCMCF('TLSCN   ') - Acquire a connection status

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING   unique-name-1   unique-name-2   unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8)  VALUE 'TLSCN   '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(4).
    02  data-name-D    PIC X(28) VALUE SPACE.
    02  data-name-E    PIC 9(9)  COMP.
    02  data-name-F1   PIC X(8).
    02  data-name-F2   PIC X(56) VALUE SPACE.
    02  data-name-G    PIC X(8)  VALUE SPACE.
    02  data-name-H    PIC X(8)  VALUE SPACE.
    02  data-name-I    PIC X(144) VALUE SPACE.
    02  data-name-J    PIC X(184) VALUE SPACE.
    02  data-name-K    PIC 9(9)  COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-L    PIC 9(9)  COMP VALUE ZERO.
01  unique-name-3.
    02  data-name-M    PIC 9(9)  COMP.
    02  unique-name-4.
        03  data-name-N    PIC X(8).
        03  data-name-O    PIC X(4).
        03  data-name-P    PIC X(4).
        03  data-name-Q    PIC X(40) VALUE LOW-VALUE.
```

Description

`CBLDCMCF('TLSCN    ')` acquires the status of a connection.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TLSCN`ΔΔΔ`'` for the request code indicating connection status acquisition.

- *data-name-C*

  Specifies how to specify the connection whose status is to be acquired.

`'LE∆∆'`

Specifies that a logical terminal name is specified for the connection whose status is to be acquired. This value cannot be specified for TP1/NET/NCSB or TP1/NET/X25-Extended.

`'CN∆∆'`

Specifies that a connection ID is specified for the connection whose status is to be acquired.

Space

`'LE∆∆'` (specification of a logical terminal name) is assumed.

- *data-name-D*

  Specify a space.

- *data-name-E*

  Specify the MCF communication process identifier of the MCF communication service for the connection that is to be processed. The permitted value range is from 0 to 239.

  This argument is ignored when a logical terminal name is used to request connection status acquisition.

  If you specify `0`, the system searches for the MCF communication service to which the specified connection ID belongs. In a configuration where many MCF communication services are running or when you issue this function many times from a UAP, we recommend that you specify the MCF communication process identifier.

- *data-name-F1*

  Specify the logical terminal name or connection ID of the connection whose status is to be acquired. Express the logical terminal name or connection ID as a maximum of 8 bytes. If the specified name is shorter than 8 bytes, pad the name with trailing spaces.

- *data-name-F2*, *data-name-G*, *data-name-H*, *data-name-I*, *data-name-J*

  Specify a space.

- *data-name-K*, *data-name-L*

  Specify `0`.

- *data-name-M*

  Specify 1 as the number of unique names from *unique-name-4* to *unique-name-n* (number of sets of *data-name-N*, *data-name-O*, *data-name-P*, and *data-name-Q*).

  When the processing is completed, the number of corresponding connections is returned.

- *data-name-Q*

  This is an area used by MCF.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-M*

  Returns the number of connections that were processed by this function.

- *data-name-N*

  Sets the connection ID of the requested connection.

- *data-name-O*

  Sets one of the following values as the protocol type of the requested connection:

  `'UA△△'`

  TP1/NET/User Agent (OSAS/UA protocol)

  `'hds△'`

  TP1/NET/HDLC (HDLC protocol)

  `'X25△'`

  TP1/NET/X25 (X.25 protocol)

  `'TP△△'`

  TP1/NET/OSI-TP (OSI TP protocol)

  `'XP△△'`

  TP1/NET/XMAP3

  `'HS1△'`

  TP1/NET/HSC (HSC1 protocol)

  `'HS2△'`

  TP1/NET/HSC (HSC2 protocol)

  `'CSB△'`

  TP1/NET/NCSB (NCSB protocol)

  `'NIF△'`

  TP1/NET/OSAS-NIF (NIF/OSI protocol)

  `'SL2△'`

TP1/NET/Secondary Logical Unit - TypeP2 (SLUTYPE-P protocol (secondary station))

`'TCP∆'`

TP1/NET/TCP/IP (TCP/IP protocol)

`'X2E∆'`

TP1/NET/X25-Extended (X.25 protocol)

■ *data-name-P*

Sets one of the following values as the status of the requested connection:

`'ACT∆'`

Connection has been established.

`'ACTB'`

Connection establishment processing is under way.

`'DCT∆'`

Connection has been released.

`'DCTB'`

Connection release processing is under way.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71001`	`CBLDCMCF('TLSCN    ')` cannot be accepted because MCF is under start processing.
`71004`	A memory shortage occurred during `CBLDCMCF('TLSCN    ')` processing.
`71005`	A communication error occurred. For the cause, see the message log file.
`71006`	An internal error occurred. For the cause, see the message log file.
`71007`	The specified connection name has not been registered.
`71008`	The specified logical terminal name has not been registered.
`71009`	`CBLDCMCF('TLSCN    ')` is not supported by the applicable communication process.
`71010`	Although the request to acquire the connection status was issued to the MCF communication process, the request was not accepted. For the cause, see the message log file.
`71011`	`CBLDCMCF('TLSCN    ')` cannot be accepted because the connection has been deleted.

Status code	Explanation
71014	The specified logical terminal name belongs to TP1/NET/NCSB or TP1/NET/X25-Extended; or, the specified connection group name belongs to TP1/NET/OSI-TP or TP1/NET/TCP/IP.
72028	The value specified for *data-name-A* is invalid.
72052	A nonzero value is specified for *data-name-K*.
72053	A nonzero value is specified for *data-name-L*.
72058	The value specified for *data-name-C* is not 'LEΔΔ', 'CNΔΔ', or a space.
72059	The value specified for *data-name-D* is not a space.
72061	A value smaller than 0 or greater than 239 is specified for *data-name-E*.
72063	*data-name-F1* begins with a space.
72065	The value specified for *data-name-F2* is not a space.
72066	The value specified for *data-name-G* is not a space.
72068	The value specified for *data-name-H* is not a space.
72070	The value specified for *data-name-I* is not a space.
72072	The value specified for *data-name-J* is not a space.
72074	The character string specified for *data-name-F1* contains an invalid character.
72076	The value 1 is not specified for *data-name-M*.

CBLDCMCF('TLSCOM ') - Acquire status of MCF communication services

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1  unique-name-2  unique-name-3
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8)  VALUE 'TLSCOM  '.
    02  data-name-B   PIC X(5).
    02  FILLER       PIC X(3).
    02  data-name-C   PIC X(4).
    02  data-name-D   PIC X(28) VALUE SPACE.
    02  data-name-E   PIC 9(9)  COMP VALUE ZERO.
    02  data-name-F   PIC X(64) VALUE SPACE.
    02  data-name-G   PIC X(8)  VALUE SPACE.
    02  data-name-H   PIC X(8)  VALUE SPACE.
    02  data-name-I   PIC X(144) VALUE SPACE.
    02  data-name-J   PIC X(184) VALUE SPACE.
    02  data-name-K   PIC 9(9)  COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-L   PIC 9(9)  COMP VALUE ZERO.
01  unique-name-3.
    02  data-name-M   PIC 9(9)  COMP.
    02  unique-name-4.
        03  data-name-N   PIC 9(9) COMP.
        03  data-name-O   PIC X(8).
        03  data-name-P   PIC X(20).
        03  data-name-Q   PIC X(12).
        03  data-name-R   PIC X(20) VALUE LOW-VALUE.
    :
    02  unique-name-n.
        03  data-name-N   PIC 9(9) COMP.
        03  data-name-O   PIC X(8).
        03  data-name-P   PIC X(20).
        03  data-name-Q   PIC X(12).
        03  data-name-R   PIC X(20) VALUE LOW-VALUE.
```

Description

`CBLDCMCF('TLSCOM  ')` acquires the statuses of the MCF communication services or application start services.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TLSCOM△△'` for the request code indicating aquisition of MCF communication service status or application start service status.

- *data-name-C*, *data-name-D*

  Specify a space.

- *data-name-E*

  Specify `0`.

- *data-name-F*, *data-name-G*, *data-name-H*, *data-name-I*, *data-name-J*

  Specify a space.

- *data-name-K*, *data-name-L*

  Specify `0`.

- *data-name-M*

  Specify the number of unique names from *unique-name-4* to *unique-name-n* (number of sets of *data-name-N*, *data-name-O*, *data-name-P*, *data-name-Q*, and *data-name-R*).

  When the processing is completed, the number of MCF communication services or application start services that have been registered in the MCF service is returned.

- *data-name-R*

  This is an area used by MCF.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-M*

  Returns the number of MCF communication services or application start services that have been registered in the MCF service.

- *data-name-N*

  Sets an MCF communication process identifier or application start process identifier.

- *data-name-O*

  Sets the MCF communication service name.

- *data-name-P*

  Sets the protocol type.

`'MCF`ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ`'`

Application start service for TP1/Message Control

`'User`ΔAgentΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/User Agent (OSAS/UA protocol)

`'HDLC`ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/HDLC (HDLC protocol)

`'X25`ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/X25 (X.25 protocol)

`'TP`ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/OSI-TP (OSI TP protocol)

`'XMAP3`ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/XMAP3

`'HSC`ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/HSC (HSC protocol)

`'NCSB`ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/NCSB (NCSB protocol)

`'OSAS-NIF`ΔΔΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/OSAS-NIF (NIF/OSI protocol)

`'NET/SLUP2`ΔΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/Secondary Logical Unit - TypeP2 (SLUTYPE-P protocol (secondary station))

`'TCP/IP`ΔΔΔΔΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/TCP/IP (TCP/IP protocol)

`'X25-EX`ΔΔΔΔΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/X25-Extended (X.25 protocol)

`'UDP/IP`ΔΔΔΔΔΔΔΔΔΔΔΔΔΔ`'`

TP1/NET/User Datagram Protocol (UDP protocol)

■ *data-name-Q*

Sets one of the following values as the status of the MCF communication service or application start service:

`'OFFLINE△△△△△'`

Service is stopped.

`'STARTING△△△△'`

Service is under preparation processing.

`'ONLINE△△△△△△'`

Service has started or is under preparation processing for termination.

`'PREENDING△△△'`

Service is under preparation processing for terminating partial stop.

`'ENDING△△△△△△'`

Service is under stop processing.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71001`	`CBLDCMCF('TLSCOM  ')` cannot be accepted because MCF is under start processing.
`71004`	A memory shortage occurred during `CBLDCMCF('TLSCOM  ')` processing.
`71005`	A communication error occurred. For the cause, see the message log file.
`71006`	An internal error occurred. For the cause, see the message log file.
`72013`	The number of MCF communication services or application start services exceeded the value specified in *data-name-M*. Information about the excess services was discarded.
`72028`	The value specified for *data-name-A* is invalid.
`72052`	A nonzero value is specified for *data-name-K*.
`72053`	A nonzero value is specified for *data-name-L*.
`72058`	The value specified for *data-name-C* is not a space.
`72059`	The value specified for *data-name-D* is not a space.
`72061`	A nonzero value is specified for *data-name-E*.
`72065`	The value specified for *data-name-F* is not a space.
`72066`	The value specified for *data-name-G* is not a space.
`72068`	The value specified for *data-name-H* is not a space.
`72070`	The value specified for *data-name-I* is not a space.

Status code	Explanation
`72072`	The value specified for *data-name-J* is not a space.
`72076`	A value of `0` or smaller is specified for *data-name-M*.

CBLDCMCF('TLSLE   ') - Acquire a logical terminal status

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1  unique-name-2  unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8)  VALUE 'TLSLE   '.
    02  data-name-B   PIC X(5).
    02  FILLER       PIC X(3).
    02  data-name-C   PIC X(4) VALUE SPACE.
    02  data-name-D   PIC X(28) VALUE SPACE.
    02  data-name-E   PIC 9(9)  COMP.
    02  data-name-F1  PIC X(8).
    02  data-name-F2  PIC X(56) VALUE SPACE.
    02  data-name-G   PIC X(8)  VALUE SPACE.
    02  data-name-H   PIC X(8)  VALUE SPACE.
    02  data-name-I   PIC X(144) VALUE SPACE.
    02  data-name-J   PIC X(184) VALUE SPACE.
    02  data-name-K   PIC 9(9)  COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-L   PIC 9(9)  COMP VALUE ZERO.
01  unique-name-3.
    02  data-name-M   PIC 9(9)  COMP.
    02  unique-name-4.
        03  data-name-N   PIC X(8).
        03  data-name-O   PIC X(4) VALUE LOW-VALUE.
        03  data-name-P   PIC X(4).
        03  data-name-Q   PIC X(40) VALUE LOW-VALUE.
```

Description

`CBLDCMCF('TLSLE   ')` acquires a logical terminal's status.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TLSLE`ΔΔΔ`'` for the request code indicating acquisition of logical terminal status.

- *data-name-C, data-name-D*

  Specify a space.

- *data-name-E*

  Specify the MCF communication process identifier of the MCF communication service that has the logical terminal to be processed. The permitted value range is from 0 to 239.

  If you specify `0`, the system searches for the MCF communication service to which the specified connection ID belongs. In a configuration where many MCF communication services are running or when you issue this function many times from a UAP, we recommend that you specify the MCF communication process identifier.

- *data-name-F1*

  Specify the name of the logical terminal whose status is to be acquired. Express the logical terminal name as a maximum of 8 bytes. If the specified name is shorter than 8 bytes, pad the name with trailing spaces.

- *data-name-F2*, *data-name-G*, *data-name-H*, *data-name-I*, *data-name-J*

  Specify a space.

- *data-name-K*, *data-name-L*

  Specify `0`.

- *data-name-M*

  Specify 1 as the number of unique names from *unique-name-4* to *unique-name-n* (number of sets of *data-name-N*, *data-name-O*, *data-name-P*, and *data-name-Q*).

  When the processing is completed, the number of corresponding logical terminals is returned.

- *data-name-O*, *data-name-Q*

  This is an area used by MCF.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-M*

  The number of logical terminals processed by this function is returned.

- *data-name-N*

  Sets the name of the requested logical terminal.

- *data-name-P*

  Sets one of the following values as the status of the requested logical terminal:

  `'ACT△'`

Logical terminal has been released from shutdown status.

'DCTΔ'

Logical terminal has been shut down.

Status codes

Status code	Explanation
00000	Normal termination.
71001	CBLDCMCF('TLSLE ') cannot be accepted because MCF is under start processing.
71004	A memory shortage occurred during CBLDCMCF('TLSLE ') processing.
71005	A communication error occurred. For the cause, see the message log file.
71006	An internal error occurred. For the cause, see the message log file.
71008	The specified logical terminal name has not been registered.
71009	CBLDCMCF('TLSLE ') is not supported by the applicable communication process.
71010	Although the request to acquire the logical terminal status was issued to the MCF communication process, the request was not accepted. For the cause, see the message log file.
71011	CBLDCMCF('TDLQLE ') cannot be accepted because the logical terminal has been deleted.
72028	The value specified for *data-name-A* is invalid.
72052	A nonzero value is specified for *data-name-K*.
72053	A nonzero value is specified for *data-name-L*.
72058	The value specified for *data-name-C* is not a space.
72059	The value specified for *data-name-D* is not a space.
72061	A value smaller than 0 or greater than 239 is specified for *data-name-E*.
72063	*data-name-F1* begins with a space.
72065	The value specified for *data-name-F2* is not a space.
72066	The value specified for *data-name-G* is not a space.
72068	The value specified for *data-name-H* is not a space.
72070	The value specified for *data-name-I* is not a space.
72072	The value specified for *data-name-J* is not a space.

Status code	Explanation
`72074`	The character string specified for *data-name-F1* contains an invalid character.
`72076`	The value `1` is not specified for *data-name-M*.

CBLDCMCF('TLSLN   ') - Acquire the acceptance status for a server-type connection establishment request

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1  unique-name-2  unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8)  VALUE 'TLSLN   '.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-C   PIC X(4) VALUE SPACE.
    02  data-name-D   PIC X(28) VALUE SPACE.
    02  data-name-E   PIC 9(9)  COMP.
    02  data-name-F   PIC X(64) VALUE SPACE.
    02  data-name-G   PIC X(8)  VALUE SPACE.
    02  data-name-H   PIC X(8)  VALUE SPACE.
    02  data-name-I   PIC X(144) VALUE SPACE.
    02  data-name-J   PIC X(184) VALUE SPACE.
    02  data-name-K   PIC 9(9)  COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-L   PIC 9(9)  COMP VALUE ZERO.
01  unique-name-3.
    02  data-name-M   PIC 9(9)  COMP VALUE 1.
    02  unique-name-4.
        03  data-name-N   PIC X(4).
        03  data-name-O   PIC X(60) VALUE LOW-VALUE.
```

Description

`CBLDCMCF('TLSLN   ')` acquires the acceptance status for a server-type connection establishment request.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TLSLN△△△'` for the request code indicating acquisition of acceptance status for a connection establishment request.

- *data-name-C*, *data-name-D*

  Specify a space.

- *data-name-E*

  Specify the MCF communication process identifier of the MCF communication service that is to be processed. The permitted value range is from 1 to 239.

- *data-name-F*, *data-name-G*, *data-name-H*, *data-name-I*, *data-name-J*

  Specify a space.

- *data-name-K*, *data-name-L*

  Specify `0`.

- *data-name-M*

  Specify 1.

- *data-name-O*

  This is an area used by MCF.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-N*

  Sets one of the following values as the acceptance status of the server-type connection establishment request:

  `'LSTN'`

  The acceptance process for the server-type connection establishment request has started.

  `'RTRY'`

  The acceptance process for the server-type connection establishment request is under start processing.

  `'ON_W'`

  The acceptance process for the server-type connection establishment request is in start request wait status.

  `'INIT'`

  The acceptance process for the server-type connection establishment request has ended.

  The table below shows the relationship between the status and function availability.

Value of *data-name-N*	Available COBOL-UAP creation program	
	CBLDCMCF('TONLN ')	**CBLDCMCF('TOFLN ')**
LSTN	N	Y
RTRY	N	Y
ON_W	Y	Y
INIT	Y	N

Legend:

Y: Can be used

N: Cannot be used

Status codes

Status code	Explanation
00000	Normal termination.
71001	CBLDCMCF('TLSLN ') cannot be accepted because MCF is under start processing.
71002	CBLDCMCF('TLSLN ') cannot be accepted because MCF is under termination processing.
71004	A memory shortage occurred during CBLDCMCF('TLSLN ') processing.
71005	A communication error occurred. For the cause, see the message log file.
71006	An internal error occurred. For the cause, see the message log file.
71009	CBLDCMCF('TLSLN ') is not supported by the applicable communication process.
71010	Although the request to acquire the acceptance status of the server-type connection establishment request was issued to the MCF communication process, the request was not accepted. For the cause, see the message log file.
72028	The value specified for *data-name-A* is invalid.
72052	A nonzero value is specified for *data-name-K*.
72053	A nonzero value is specified for *data-name-L*.
72058	The value specified for *data-name-C* is not a space.
72059	The value specified for *data-name-D* is not a space.
72061	A value of 0 or smaller or greater than 239 is specified for *data-name-E*.
72065	The value specified for *data-name-F* is not a space.

Status code	Explanation
`72066`	The value specified for *data-name-G* is not a space.
`72068`	The value specified for *data-name-H* is not a space.
`72070`	The value specified for *data-name-I* is not a space.
`72072`	The value specified for *data-name-J* is not a space.
`72076`	The value `1` is not specified for *data-name-M*.

CBLDCMCF('TOFLN   ') - Stop accepting server-type connection establishment requests

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8)  VALUE 'TOFLN   '.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-C   PIC X(4) VALUE SPACE.
    02  data-name-D   PIC X(28) VALUE SPACE.
    02  data-name-E   PIC 9(9)  COMP.
    02  data-name-F   PIC X(64) VALUE SPACE.
    02  data-name-G   PIC X(8)  VALUE SPACE.
    02  data-name-H   PIC X(8)  VALUE SPACE.
    02  data-name-I   PIC X(144) VALUE SPACE.
    02  data-name-J   PIC X(184) VALUE SPACE.
    02  data-name-K   PIC 9(9)  COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-L   PIC 9(9)  COMP VALUE ZERO.
```

Description

`CBLDCMCF('TOFLN   ')` stops accepting server-type connection establishment requests.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TOFLN∆∆∆'` for the request code indicating that acceptance of connection establishment requests is to be stopped.

- *data-name-C, data-name-D*

  Specify a space.

- *data-name-E*

  Specify the MCF communication process identifier of the MCF communication service that is to be processed. The permitted value range is from 1 to 239.

- *data-name-F*, *data-name-G*, *data-name-H*, *data-name-I*, *data-name-J*

  Specify a space.

- *data-name-K*, *data-name-L*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71001`	`CBLDCMCF('TOFLN    ')` cannot be accepted because MCF is under start processing.
`71002`	`CBLDCMCF('TOFLN    ')` cannot be accepted because MCF is under termination processing.
`71004`	A memory shortage occurred during `CBLDCMCF('TOFLN    ')` processing.
`71005`	A communication error occurred. For the cause, see the message log file.
`71006`	An internal error occurred. For the cause, see the message log file.
`71009`	`CBLDCMCF('TOFLN    ')` is not supported by the applicable communication process.
`71010`	Although the request to stop accepting server-type connection establishment requests was issued to the MCF communication process, the request was not accepted. For the cause, see the message log file.
`72028`	The value specified for *data-name-A* is invalid.
`72052`	A nonzero value is specified for *data-name-K*.
`72053`	A nonzero value is specified for *data-name-L*.
`72058`	The value specified for *data-name-C* is not a space.
`72059`	The value specified for *data-name-D* is not a space.
`72061`	A value of `0` or smaller or greater than `239` is specified for *data-name-E*.
`72065`	The value specified for *data-name-F* is not a space.
`72066`	The value specified for *data-name-G* is not a space.
`72068`	The value specified for *data-name-H* is not a space.
`72070`	The value specified for *data-name-I* is not a space.

Status code	Explanation
`72072`	The value specified for *data-name-J* is not a space.

CBLDCMCF('TONLN   ') - Start accepting server-type connection establishment requests

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCMCF'  USING  unique-name-1   unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8)  VALUE 'TONLN   '.
    02  data-name-B   PIC X(5).
    02  FILLER       PIC X(3).
    02  data-name-C   PIC X(4) VALUE SPACE.
    02  data-name-D   PIC X(28) VALUE SPACE.
    02  data-name-E   PIC 9(9)  COMP.
    02  data-name-F   PIC X(64) VALUE SPACE.
    02  data-name-G   PIC X(8)  VALUE SPACE.
    02  data-name-H   PIC X(8)  VALUE SPACE.
    02  data-name-I   PIC X(144) VALUE SPACE.
    02  data-name-J   PIC X(184) VALUE SPACE.
    02  data-name-K   PIC 9(9)  COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-L   PIC 9(9)  COMP VALUE ZERO.
```

Description

`CBLDCMCF('TONLN   ')` starts accepting server-type connection establishment requests.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'TONLN△△△'` for the request code indicating start of acceptance of connection establishment requests.

- *data-name-C*, *data-name-D*

  Specify a space.

- *data-name-E*

  Specify the MCF communication process identifier of the MCF communication service that is to be processed. The permitted value range is from 1 to 239.

- *data-name-F*, *data-name-G*, *data-name-H*, *data-name-I*, *data-name-J*

  Specify a space.

- *data-name-K*, *data-name-L*

  Specify `0`.

Data area to which a value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71001`	`CBLDCMCF('TONLN    ')` cannot be accepted because MCF is under start processing.
`71002`	`CBLDCMCF('TONLN    ')` cannot be accepted because MCF is under termination processing.
`71004`	A memory shortage occurred during `CBLDCMCF('TONLN    ')` processing.
`71005`	A communication error occurred. For the cause, see the message log file.
`71006`	An internal error occurred. For the cause, see the message log file.
`71009`	`CBLDCMCF('TONLN    ')` is not supported by the applicable communication process.
`71010`	Although the request to start accepting server-type connection establishment requests was issued to the MCF communication process, the request was not accepted. For the cause, see the message log file.
`72028`	The value specified for *data-name-A* is invalid.
`72052`	A nonzero value is specified for *data-name-K*.
`72053`	A nonzero value is specified for *data-name-L*.
`72058`	The value specified for *data-name-C* is not a space.
`72059`	The value specified for *data-name-D* is not a space.
`72061`	A value of `0` or smaller or greater than `239` is specified for *data-name-E*.
`72065`	The value specified for *data-name-F* is not a space.
`72066`	The value specified for *data-name-G* is not a space.
`72068`	The value specified for *data-name-H* is not a space.
`72070`	The value specified for *data-name-I* is not a space.

Status code	Explanation
`72072`	The value specified for *data-name-J* is not a space.

Performance verification trace (CBLDCPRF)

This section describes the programs available for the performance verification trace. The COBOL-UAP creation programs for the performance verification trace are as follows:

- `CBLDCPRF('PRFGETN ')` - Report the sequential number for an acquired performance verification trace
- `CBLDCPRF('PRFPUT  ')` - Acquire user-specific performance verification traces

The COBOL-UAP creation programs (CBLDCPRF) for the performance verification trace are available on UAPs that run TP1/Server Base or TP1/LiNK. However, you must have installed TP1/Extension 1 before you can use this facility. Note that if TP1/Extension 1 has not been installed, system operation is unpredictable.

You can use the COBOL language template as a sample when defining the data part (`DATA DIVISION`) of the COBOL-UAP creation programs. The COBOL language template of the performance verification trace (CBLDCPRF) is located in `DCPRF.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCPRF('PRFGETN ') - Report the sequential number for an acquired performance verification trace

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCPRF' USING unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01 unique-name-1.
   02 data-name-A  PIC X(8) VALUE 'PRFGETN '.
   02 data-name-B  PIC X(5).
   02 FILLER       PIC X(3).
   02 data-name-Z  PIC S9(9) COMP VALUE ZERO.

01 unique-name-2   PIC 9(4) COMP.
```

Description

`CBLDCPRF('PRFGETN ')` reports the acquired sequential trace number within the process of the latest performance verification trace (prf trace) acquired before the program was called. It reports this information to the `CBLDCPRF('PRFGETN ')` call source.

If no performance verification trace has been acquired in the process that called `CBLDCPRF('PRFGETN ')`, the acquired sequential trace number within the process is 0.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'PRFGETN∆'` for the request code indicating report the sequential number for an acquired performance verification trace.

- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *unique-name-2*

  The sequential number for an acquired performance verification trace is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`04601`	The value specified for the data name is invalid. This status code includes cases in which the request code (*data-name-A*) is invalid.

CBLDCPRF('PRFPUT ') - Acquire user-specific performance verification traces

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCPRF' USING unique-name-1  unique-name-2
```

- PROCEDURE DIVISION specification

```
01 unique-name-1.
   02 data-name-A  PIC X(8) VALUE 'PRFPUT  '.
   02 data-name-B  PIC X(5).
   02 FILLER       PIC X(3).
   02 data-name-Z  PIC S9(9) COMP VALUE ZERO.
01 unique-name-2.
   02 data-name-C  PIC 9(4) COMP.
   02 data-name-D  PIC 9(4) COMP.
   02 data-name-E  PIC X(n).
```

Description

CBLDCPRF('PRFPUT ') acquires a user-specific performance verification trace (prf trace).

Data areas whose values are set in the UAP

- *data-name-A*

  Specify VALUE 'PRFPUT△△' for the request code indicating acquisition of a user-specific performance verification trace.

- *data-name-Z*

  Specify 0.

- *data-name-C*

  Specify the event ID of the event to be acquired. The range of available event IDs is 0x0001 to 0x0040.

- *data-name-D*

  Specify the data length of the trace data to be acquired. The specifiable data length is 4 bytes to 256 bytes. The data length must be a multiple of 4 bytes.

- *data-name-E*

  Specify the trace data to be acquired. Data that fits within the length specified for *data-name-D* is valid as trace data.

Data area to which a value is returned from OpenTP1

- *data-name-B*

A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`04601`	The value specified for the data name is invalid.This status code includes cases in which the request code (*data-name-A*) is invalid.

Note

Even if `CBLDCPRF('PRFPUT   ')` returns the status code 00000, the trace has not necessarily been properly acquired. This is because data may be lost during trace acquisition processing if multiple processes issue acquisition requests simultaneously because no lock is used.

Remote API facilities (CBLDCRAP)

This section explains the programs to be used when the user uses remote API facilities to manage establishment and release of connections. The COBOL-UAP creation programs provided by the remote API facilities are as follows:

- `CBLDCRAP('CONNECT ')` - Establish a connection with a RAP-processing listener
- `CBLDCRAP('CONNECTX')` - Establish a connection with a RAP-processing listener
- `CBLDCRAP('DISCNCT ')` - Release a connection with a RAP-processing listener

The COBOL-UAP creation programs (CBLDCRAP) provided by the remote API facilities can be used in UAPs of TP1/Server Base or TP1/LiNK.

When the user defines `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language template for the remote API facilities (CBLDCRAP) is stored in `DCRAP.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCRAP('CONNECT ') - Establish connection with a RAP-processing listener

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCRAP' USING unique-name-1 unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8) VALUE 'CONNECT '.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-C   PIC S9(9) COMP VALUE ZERO.
    02  data-name-D   PIC S9(9) COMP.
01  unique-name-2.
    02  data-name-E   PIC X(64).
    02  data-name-F   PIC S9(9) COMP.
```

Description

`CBLDCRAP('CONNECT ')` establishes a connection between a RAP-processing listener and a RAP-processing client.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'CONNECT∆'` for the request code indicating the request for establishment of a connection with a RAP-processing listener.

- *data-name-C*

  Specify `0`.

- *data-name-E*

  Specify the host name of the OpenTP1 node on which the RAP-processing listener was activated.

  The specified host name must consist of 1 to 64 characters. To specify a host name longer than 65 characters, use `CBLDCRAP('CONNECTX')`.

- *data-name-F*

  Specify the port number of the well-known port being used by the RAP-processing listener.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-D*

  This area receives the service ID.

  A service ID is returned when a connection with the RAP-processing listener was established successfully. Use the same service ID in `CBLDCRAP('DISCNCT ')` when releasing the connection.

Status codes

Status code	Explanation
`00000`	Normal termination. A connection was established with the RAP-processing listener.
`05501`	Either the value specified in a data area is invalid, or a data area has not been set. Check the following items: • Values of *data-name-A* and *data-name-C* (`'CONNECT '` and `0`, respectively) • Number of characters set in *data-name-E* (1 to 64) • Allowable range of the *data-name-F* value (1 to 65535) • Data area *data-name-D*
`05502`	The protocol is invalid. Possible causes are as follows: • `CBLDCRPC('OPEN   ')` was not called. • Although the `rpc_rap_auto_connect` operand in the user service definition had been set to `Y`, the function `CBLDCRAP('CONNECT ')` was called. • The `-w` option was not specified in the `dcsvgdef` definition command in the user service network definition.
`05503`	The memory became insufficient.
`05517`	The specified value exceeds the maximum number of `CBLDCRAP('CONNECT ')` functions which can be called from a single process.
`05505`	A network error occurred during communication with the RAP-processing listener.
`05506`	A timeout occurred during communication with the RAP-processing listener.
`05507`	The number of sockets became insufficient.
`05508`	The host name cannot be resolved.
`05521`	The RAP-processing listener is being terminated.
`05522`	An error which prevents continuation of processing occurred. Possible causes of the error are as follows: • An unexpected message was received. • A message was received unexpectedly from a remote system.

Status code	Explanation
05523	An unexpected error occurred during system call.
05531	An attempt was made to establish a connection with a RAP-processing listener which is on an unconnected network.
05520	The memory became insufficient on the RAP-processing listener or RAP-processing server.
05532	A connection could not be established within the message exchange monitoring time specified in the `rap_watch_time` operand of the RAP-processing listener service definition.
05533	A system error occurred in the RAP-processing listener.
05528	The RAP-processing listener is being started or terminated.
05529	A connection has already been established with the RAP-processing listener.
05534	The specified value exceeds the maximum number of requests which can be accepted for connection with a RAP-processing client that is managed by a RAP-processing listener.

Note

If `CBLDCRAP('CONNECT ')` returns with an error and the status code is a value other than 05529, connection was not established with the RAP-processing listener.

The error code acquired by the UAP trace is as follows:

0: No error

1: No value was specified for *unique-name-1*.

2: The request code (*data-name-A*) is invalid.

3: No value was specified for *unique-name-2*.

6: `CBLDCRAP('CONNECT')` was called while the value `Y` was specified in the `rpc_rap_auto_connect` operand in the user service definition. Another possibility is that the user service network has not been defined.

CBLDCRAP('CONNECTX') - Establish connection with a RAP-processing listener

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCRAP' USING unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01 unique-name-1.
   02 data-name-A  PIC X(8) VALUE 'CONNECTX'.
   02 data-name-B  PIC X(5).
   02 FILLER    PIC X(3).
   02 data-name-C  PIC S9(9) COMP VALUE 1.
   02 data-name-D  PIC S9(9) COMP.
01 unique-name-2.
   02 data-name-E  PIC S9(9) COMP.
   02 data-name-F  PIC X(n).
   02 FILLER    PIC X(1) VALUE LOW-VALUE.
```

Description

`CBLDCRAP('CONNECTX')` establishes a connection between a RAP-processing listener and a RAP-processing client. The host name area is a variable-length area. It can accommodate a host name longer than 65 characters.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'CONNECTX'` for the request code indicating the request for establishment of a connection with a RAP-processing listener.

- *data-name-C*

  Specify 1.

- *data-name-E*

  Specify the port number of the well-known port being used by the RAP-processing listener.

- *data-name-F*

  Specify the host name of the OpenTP1 node on which the RAP-processing listener was activated. The specified host name must consist of 1 to 255 characters.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-D*

  This area receives the service ID.

  A service ID is returned when a connection with the RAP-processing listener was established successfully. Use the same service ID in `CBLDCRAP('DISCNCT ')` when releasing the connection.

Status codes

Status code	Explanation
`00000`	Normal termination. A connection was established with the RAP-processing listener.
`05501`	The value specified for a data area is invalid or a required data area is not specified. Review the following values: • Values of *data-name-A* and *data-name-C* (`'CONNECTX'` and `1`, respectively) • Allowable range of the *data-name-E* value (1 to 65535) • Number of characters in *data-name-F* (1 to 255) • Area of *data-name-D*
`05502`	The protocol is invalid. Possible causes are as follows: • `CBLDCRPC('OPEN ')` was not called. • Although the `rpc_rap_auto_connect` operand in the user service definition had been set to `Y`, the function `CBLDCRAP('CONNECTX')` was called. • The `-w` option was not specified in the `dcsvgdef` definition command in the user service network definition.
`05503`	The memory became insufficient.
`05505`	A network error occurred during communication with the RAP-processing listener.
`05506`	A timeout occurred during communication with the RAP-processing listener.
`05507`	The number of sockets became insufficient.
`05508`	The host name cannot be resolved. Verify *data-name-F* or the DNS server.
`05517`	The maximum number of `CBLDCRAP('CONNECTX')` calls from a single process was exceeded.
`05520`	The memory became insufficient on the RAP-processing listener or RAP-processing server.
`05521`	The RAP-processing listener is being terminated. Alternatively, verify *data-name-E*.

Status code	Explanation
`05522`	An error which prevents continuation of processing occurred. Possible causes of the error are as follows: • An unexpected message was received. • A message was received unexpectedly from a remote system.
`05523`	An unexpected error occurred during system call.
`05528`	The RAP-processing listener is being started or terminated.
`05529`	A connection has already been established with the RAP-processing listener.
`05531`	An attempt was made to establish a connection with a RAP-processing listener which is on an unconnected network.
`05532`	A connection could not be established within the message exchange monitoring time specified in the `rap_watch_time` operand of the RAP-processing listener service definition.
`05533`	A system error occurred in the RAP-processing listener.
`05534`	The specified value exceeds the maximum number of requests which can be accepted for connection with a RAP-processing client that is managed by a RAP-processing listener.

Note

If `CBLDCRAP('CONNECTX')` returns with an error and the status code is a value other than 05529, connection was not established with the RAP-processing listener.

The error code acquired by the UAP trace is as follows:

0: No error

1: No value was specified for *unique-name-1*.

2: The request code (*data-name-A*) is invalid.

3: No value was specified for *unique-name-2*.

6: `CBLDCRAP('CONNECT')` was called even though the value specified for the `rpc_rap_auto_connect` operand in the user service definition is Y. Alternatively, the user service network has not been defined.

CBLDCRAP('DISCNCT ') - Release a connection with a RAP-processing listener

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCRAP' USING unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8) VALUE 'DISCNCT '.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-C   PIC S9(9) COMP VALUE ZERO.
    02  data-name-D   PIC S9(9) COMP.
```

Description

`CBLDCRAP('DISCNCT ')` releases a connection established between a RAP-processing listener and a RAP-processing client.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DISCNCT∆'` for the request code indicating the request for release of a connection with a RAP-processing listener.

- *data-name-C*

  Specify `0`.

- *data-name-D*

  Specify the service ID that was received for `CBLDCRAP('CONNECT ')`.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status Code	Explanation
`00000`	Normal termination. The connection with the RAP-processing listener was released.

Status Code	Explanation
`05501`	Either the value specified in a data area is invalid, or a data area has not been set. Check the following items: • Values of *data-name-A* and *data-name-C* (`'CONNECT '` and `0`, respectively) • Data area *data-name-D*
`05502`	The protocol is invalid. Possible causes are as follows: • `CBLDCRPC('OPEN   ')` was not called. • Although the `rpc_rap_auto_connect` operand in the user service definition had been set to `Y`, the function `CBLDCRAP('DISCNCT ')` was called. • The `-w` option was not specified in the `dcsvgdef` definition command in the user service network definition.
`05503`	The memory became insufficient.
`05505`	A network error occurred during communication with the RAP-processing listener.
`05506`	A timeout occurred during communication with the RAP-processing listener.
`05521`	The RAP-processing listener is being terminated.
`05522`	An error which prevents continuation of processing occurred. Possible causes of the error are as follows: • An unexpected message was received. • A message was received unexpectedly from a remote system.
`05523`	An unexpected error occurred during system call.

Note

If the `CBLDCRAP('DISCNCT ')` function returns with an error and the status code is a value other than 05501 or 05502, the connection with the RAP-processing listener was released.

The error code acquired by the UAP trace is as follows:

0: No error

1: No value was specified for *unique-name-1*.

2: The request code (*data-name-A*) is invalid.

Remote procedure calls (CBLDCRPC, CBLDCRSV)

This section gives the syntax and other information of the following COBOL-UAP creation programs which are used via OpenTP1 remote procedure calls for client/ server communication:

- `CBLDCRPC('CALL     ')` - Request a remote service
- `CBLDCRPC('CLOSE    ')` - Terminate an application program
- `CBLDCRPC('CLTSEND  ')` - Report data to CUP unidirectionally
- `CBLDCRPC('DISCARDF ')` - Reject the receiving of processing results
- `CBLDCRPC('DISCARDS ')` - Reject acceptance of specific processing results
- `CBLDCRPC('GETCLADR ')` - Acquire the node address of a client UAP
- `CBLDCRPC('GETERDES ')` - Acquire the descriptor of an asynchronous response-type RPC request which has encountered an error
- `CBLDCRPC('GETGWADR ')` - Acquire the node address of a gateway
- `CBLDCRPC('GETSVPRI ')` - Reference the schedule priority of a service request
- `CBLDCRPC('GETWATCH ')` - Reference the service response waiting interval
- `CBLDCRPC('OPEN     ')` - Start an application program
- `CBLDCRPC('POLLANYR ')` - Receive processing results in asynchronous mode
- `CBLDCRPC('SETSVPRI ')` - Set a schedule priority of a service request
- `CBLDCRPC('SETWATCH ')` - Update the service response waiting interval
- `CBLDCRPC('SVRETRY  ')` - Retry a service program
- `CBLDCRSV('MAINLOOP ')` - Start an SPP service

The COBOL-UAP creation programs for remote procedure calls (`CBLDCRPC`, `CBLDCRSV`) can be used in UAPs of both TP1/Server Base and TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language templates for remote procedure calls (`CBLDCRPC`, `CBLDCRSV`) are stored in `DCRPC.cbl` and `DCRSV.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCRPC('CALL   ') - Request a remote service

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING  unique-name-1  unique-name-2
                         unique-name-3
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'CALL    '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
    02  data-name-D    PIC S9(9) COMP.
    02  data-name-E    PIC X(32).
    02  data-name-F    PIC X(n).
01  unique-name-2.
    02  data-name-G    PIC S9(9) COMP.
    02  data-name-H    PIC X(n).
01  unique-name-3.
    02  data-name-I    PIC S9(9) COMP.
    02  data-name-J    PIC X(n).
```

Description

`CBLDCRPC('CALL    ')` requests an SPP service. This function can be called without consideration of the node containing the requesting service.

Specify a service group name and service name for a data area of `CBLDCRPC('CALL    ')` to request a service. A service request is addressed to the service program corresponding to the specified names.

A UAP which calls `CBLDCRPC('CALL    ')` can be used regardless of whether it has been executed as a transaction. When a service is requested by `CBLDCRPC('CALL    ')` from the process which has been executed as a transaction, the requested service process runs as a transaction branch.

Before this function can be used, the OpenTP1 at the node containing the server UAP to which the service request is addressed must be active.

Receiving a signal while waiting for a response after execution of `CBLDCRPC('CALL    ')` does not cause `CBLDCRPC('CALL    ')` to be returned.

The following items are described after the list of status codes. See each description for details on `CBLDCRPC('CALL    ')`.

(1) Data areas of `CBLDCRPC('CALL    ')`

(2) Error cases of `CBLDCRPC('CALL    ')`

(3) Timing when `CBLDCRPC('CALL    ')` results in error

(4) Specification for reexecuting the service request if `CBLDCRPC('CALL    ')` results in error

(5) When a priority is given to a service request

(6) Difference between status codes `00310` and `00306`

(7) Specification for the return of status code `00378`

(8) Relationship between status codes and synchronization point processing

(9) Notes on requesting a service

(10) When a service is requested with domain qualification

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'CALL`ΔΔΔΔ`'` for the request code indicating that a remote service is requested.

- *data-name-C*

  Specify the RPC mode and option with a value:

  `0`: Synchronous response-type RPC

  `2`: Asynchronous response-type RPC

  `1`: Nonresponse-type RPC

  `4`: Chained RPC

  Add 32 to the value indicating the RPC mode to avoid having the requested processing handled as a transaction. When 32 is added, the processing of the service function is not handled as a transaction even if the service request is issued from the transaction.

  Add 256 to the value indicating the RPC mode when the service group name is specified with domain qualification. An RPC with domain qualification cannot be a transaction branch. Therefore, add (256 + 32) to the value indicating the RPC mode whenever `CBLDCRPC('CALL    ')` is used from the transaction.

  Example 1:

  When a nontransaction service is requested by using a synchronous response-type RPC, specify `32` (0 + 32) for *data-name-C*.

  Example 2:

When a service is requested by using a synchronous response-type RPC with domain qualification from the transaction, specify `290` (2 + 256 + 32) for *data-name-C*.

- *data-name-E*

  Specify the SPP service name with an ASCII character string of up to 31 bytes. End the character string with space. The space is not counted in the length of the character string.

- *data-name-F*

  Specify the SPP service group name with an ASCII character string of up to 31 bytes. End the character string with space. The space is not counted in the length of the character string.

  When requesting a service with domain qualification, specify the service group name suffixed by an at mark (@) and the DNS domain name, and end the character string with space.

- *data-name-G*

  Specify the input parameter length of the service (the length of *data-name-H*) within the range from 1 to `DCRPC_MAX_MESSAGE_SIZE`#. `DCRPC_MAX_MESSAGE_SIZE` is defined in `dcrpc.h`.

  #: If you used the `rpc_max_message_size` operand, the value of this data area is the value specified in the `rpc_max_message_size` operand and not the value of `DCRPC_MAX_MESSAGE_SIZE` (1 megabyte).

- *data-name-H*

  Specify the input parameter of the service.

- *data-name-I*

  Specify the length of the response (the length of *data-name-J*) within the range from 1 to `DCRPC_MAX_MESSAGE_SIZE`#. `DCRPC_MAX_MESSAGE_SIZE` is defined in `dcrpc.h`.

  #: If you used the `rpc_max_message_size` operand, the value of this data area is the value specified in the `rpc_max_message_size` operand and not the value of `DCRPC_MAX_MESSAGE_SIZE` (1 megabyte).

Data areas to which values are returned from server UAP

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-D*

  The descriptor is returned when asynchronous response-type RPCs are used.

- *data-name-I*

  The length of the response set by the service program is returned.

- *data-name-J*

  The response set by the service program is returned.

Status codes

The following status codes are returned from OpenTP1, not from the service program.

Status code	Explanation
`00000`	Normal termination. In the case of asynchronous response-type RPC, the descriptor was set in *data-name-D*.
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
`00302`	`CBLDCRPC('OPEN    ')` was not called.
`00304`	A memory shortage occurred. Or, a service request was not accepted because a space shortage occurred in the message storage buffer pool (`message_store_buflen` operand) of the SPP to which the service was requested. If necessary, revise the `message_store_buflen` operand in the user service default definition or in the user service definition of the SPP to which the service was requested.
`00306`	A communication failure occurred. Check if a network failure has occurred.
`00307`	The response wait time in `CBLDCRPC('CALL    ')` has elapsed. If necessary, revise the response wait time specified in `CBLDCRPC('CALL    ')` (`watch_time` operand and arguments in `CBLDCRPC('SETWATCH')`).
	The SPP to which the service was requested terminated abnormally during execution of a service program. Check the cause of abnormal termination of the SPP to which the service was requested.
`00308`	The input parameter length specified in *data-name-G* exceeded the maximum. If necessary, revise the *data-name-G* setting.
`00309`	The length of the response (*data-name-D* of the service program) set in the service program of the SPP to which the service was requested exceeded the response length (*data-name-I* of the service program) in `CBLDCRPC('CALL    ')`. If necessary, revise the response length (*data-name-D* of the service program) set in the service program of the SPP to which the service was requested.
`00310`	The service group name set in *data-name-F* is invalid, or the SPP to which the service was requested with the service group set in *data-name-F* was not running. If necessary, revise the *data-name-F* setting, or start the SPP to which the service was requested with the service group set in *data-name-F*.

Status code	Explanation
00311	The service name set in *data-name-E* is invalid, or the service name set in *data-name-E* by the SPP to which the service was requested has not been specified in the `service` operand in the user service definition file. If necessary, revise the *data-name-E* setting, or specify the service name set in *data-name-E* also in the `service` operand for the SPP to which the service was requested.
00312	The SPP to which the service was requested with the service group set in *data-name-F* is under server shutdown or service shutdown status. Check the cause of the shutdown, and then release the SPP from shutdown status.
00313	The SPP to which the service was requested is under termination processing.
00314	The SPP to which the service was requested with the service group set in *data-name-F* is not running, or a communication failure might have occurred during the service request send processing. Start the SPP to which the service was requested with the service group set in *data-name-F*. If the SPP is already running, check to see if a network failure has occurred.
	While 0 was specified for the service request response time (`watch_time` operand and an argument in `CBLDCRPC('SETWATCH')`), the SPP to which the service was requested terminated abnormally during execution of a service program. Check the cause of abnormal termination of the SPP to which the service was requested.
00315	OpenTP1 for the SPP to which the service was requested is not running. OpenTP1 might be under termination processing or a communication failure might have occurred during the service request send processing. Start OpenTP1 for the SPP to which the service was requested, or check for a network failure.
00316	A system error (internal conflict) occurred in the SPP to which the service was requested.
00317	A memory shortage occurred in the SPP to which the service was requested.
00318	A system error (internal conflict) occurred in the UAP that requested the service.
00319	The response length (*data-name-D* of the service program) set by a service program of the SPP to which the service was requested is outside the range from 1 to the value defined in `DCRPC_MAX_MESSAGE_SIZE`.# If necessary, revise the response length (*data-name-D* of the service program) in the service program of the SPP to which the service was requested.
00320	OpenTP1 for the SPP to which the service was requested is under start processing.
00323	A memory shortage occurred in the UAP that is requesting the service or the SPP to which the service was requested. When this value is returned, the transaction branch rolls back. Check if unneeded memory is allocated by the UAP that is requesting the service or the SPP to which the service was requested.

Status code	Explanation
`00324`	A system error (internal conflict) occurred in the UAP that requested the service. When this value is returned, the transaction branch rolls back.
`00325`	A system error (internal conflict) occurred in the SPP to which the service was requested. When this value is returned, the transaction branch rolls back.
`00326`	The response length (*data-name-D* of the service program) set in the service program of the SPP to which the service was requested exceeded the response length (*data-name-I* of the service program) in `CBLDCRPC('CALL    ')`. When this value is returned, the transaction branch rolls back. If necessary, revise the response length (*data-name-D* of the service program) set in the service program of the SPP to which the service was requested.
`00327`	When the inter-node load-balancing facility and the extended internode load-balancing facility are used, the transaction attributes (`atomic_update` operand) do not match among the SPPs with the same service group name to which the service was requested. Another possibility is that the inter-node load-balancing facility and the extended internode load-balancing facility cannot be used because the version of OpenTP1 at the node to which loads are to be distributed is earlier than that of the OpenTP1 for the UAP that is requesting the service. This value is returned only when the service request is issued to an SPP that uses the inter-node load-balancing facility and the extended internode load-balancing facility. If necessary, revise the transaction attribute (`atomic_update` operand) of the SPP that uses the inter-node load-balancing facility and the extended internode load-balancing facility, or revise if necessary the version of OpenTP1.
	The `dcsvgdef` definition command was used to issue a service request to a user server with the non-transaction attribute (`atomic_update` is `N` in the user service definition or `jnl_fileless_option` is `Y` in the system common definition). However, 32 was not added to *data-name-C* for `CBLDCRPC('CALL    ')`.
`00328`	The domain name of the service group name with the domain qualification in *data-name-F* is invalid. If necessary, revise the domain name.
`00329`	A service was requested with a domain qualification in *data-name-F*, but the port number of the domain representative schedule service was not found. If necessary, revise the `domain_masters_port` operand setting in the system common definition and the port number setting for the domain representative schedule service in `/etc/services`.
`00356`	The SPP to which the service was requested (the server that receives requests from socket) cannot receive the service request. If necessary, revise the `max_socket_msg` and `max_socket_msglen` operands in the user service definition or the user service default definition for the SPP to which the service was requested.

Status code	Explanation
`00366`	When the online tester was being used, a service request was issued from a UAP in the test mode to an SPP in the nontest mode or from a UAP in the nontest mode to an SPP in the test mode. If necessary, revise the UAP's test mode setting.
`00367`	`CBLDCRPC('CALL    ')` with nontransaction setting specified for *data-name-C* was called to request a service after a chained RPC with the transaction attribute was executed.
`00370`	The SPP to which the service was requested is protected by the security facility. The UAP that requested the service by executing `CBLDCRPC('CALL    ')` does not have permission to access the SPP to which the service was requested. If necessary, revise the access permissions for the SPP to which the service was requested.
`00372`	The transaction branch cannot be started because it exceeds the maximum number of transaction branches that can be activated concurrently in the OpenTP1 for the SPP to which the service was requested. If necessary, revise the setting in the `trn_tran_process_count` operand in the transaction service definition.
	The transaction branch cannot be started because it exceeds the maximum number of child transaction branches that can be activated from one transaction branch by the UAP that is requesting the service. If necessary, revise the setting in the `trn_max_subordinate_count` operand in the transaction service definition.
	32 is not added to the value specified for *data-name-C* when a service with domain qualification specified in a transaction is requested.
	Transaction branching cannot start because the SPP to which the service was requested encountered a resource manager (RM) error. Eliminate the cause of the resource manager (RM) error and then re-execute the function.
	In the System Environment window of TP1/LiNK, the **Transaction Facility** item is not set to **Yes**. If necessary, revise the **Transaction Facility** setting in the System Environment window of TP1/LiNK.
`00378`	The SPP that received the service request terminated abnormally during execution of the service program. If necessary, revise the service program processing of the SPP that received the service request. This status code is returned only when `"00000001"` is specified in the `pc_extend_function` operand in the user service definition of the UAP that requested the service. When `"00000000"` is specified in the `rpc_extend_function` operand or when this operand is omitted, status code `00307` or `00314` is returned, instead of `00378`.

#: If you used the `rpc_max_message_size` operand, the value of this data area is the value specified in the `rpc_max_message_size` operand and not the value of `DCRPC_MAX_MESSAGE_SIZE` (1 megabyte).

(1) Data areas of CBLDCRPC('CALL   ')

Data areas of `CBLDCRPC('CALL    ')` are explained below.

- Values passed to server UAP

Allocate an area (*data-name-J*) for the response from the service program before requesting a service. The client UAP sets the following values in `CBLDCRPC('CALL    ')`.

  - Input parameter (*data-name-H*)
  - Input parameter length (*data-name-G*)
  - Response length (*data-name-I*)

The input parameter, input parameter length, and response length values which are set in `CBLDCRPC('CALL    ')` of the client UAP are passed to the service program as is. Change the notation of character codes or digits in the processing of the client UAP or requested service program if required. If a service request is addressed to a service program which does not return any response, the specified response length is ignored.

The maximum values of input parameter length and response length are declared as `DCRPC_MAX_MESSAGE_SIZE`# in the header file `dcrpc.h`. See the contents of `dcrpc.h` to confirm the maximum values.

#: If you used the `rpc_max_message_size` operand, the value of this data area is the value specified in the `rpc_max_message_size` operand and not the value of `DCRPC_MAX_MESSAGE_SIZE` (1 megabyte).

- Values returned from server UAP

When the service program terminates and response is returned, the following values can be referenced:

  - Response from service program (*data-name-J*)
  - Length of response from service program (*data-name-I*)

The value of *data-name-I* is the length of the response which is actually returned from the service program. The values of *data-name-J* and *data-name-I* can be referenced in the following cases depending on the RPC mode:

  - In the case of synchronous response-type RPC and chained RPC

    The values of *data-name-J* and *data-name-I* can be referenced when `CBLDCRPC('CALL    ')` returns.

  - In the case of asynchronous response-type RPC

    The value of *data-name-J* can be referenced when `CBLDCRPC('POLLANYR')` which receives the response returns. The value of *data-name-I* cannot be referenced.

- In the case of nonresponse-type RPC

  The values of *data-name-J* and *data-name-I* cannot be referenced.

If `CBLDCRPC('CALL     ')` or `CBLDCRPC('POLLANYR')` returns with an error, the values of *data-name-J* and *data-name-I* cannot be referenced.

If the returned response is larger than the response area acquired by the client UAP (*data-name-I*), `CBLDCRPC('CALL     ')` returns with an error, giving the status code `00309`.

■ Value specified for data-name-C

The value specified for *data-name-C* and the execution result of `CBLDCRPC('CALL     ')` are explained below.

- Synchronous response-type RPC (when 0 is specified for *data-name-C*)

  `CBLDCRPC('CALL     ')` will not return until a response returns or a communication error occurs.

- Asynchronous response-type RPC (when 2 is specified for *data-name-C*)

  `CBLDCRPC('CALL     ')` will return immediately. The response can be referenced after the response is received asynchronously in `CBLDCRPC('POLLANYR')`. Do not free the response storage area (*data-name-J*) until the asynchronous response-type RPC is terminated due to one of the following causes:

  - A response is received by `CBLDCRPC('POLLANYR')`
  - The receiving of responses is rejected by `CBLDCRPC('DISCARDF')`
  - Commitment or rollback is performed when a service is requested from a transaction.

When an asynchronous response-type RPC is used in a transaction, receive responses by using `CBLDCRPC('POLLANYR')` before performing the synchronization point processing (commitment or rollback). No response can be received by `CBLDCRPC('POLLANYR')` after the synchronization point processing. To designate a specific response received by `CBLDCRPC('POLLANYR')`, specify the positive integer (descriptor), which is returned by `CBLDCRPC('CALL     ')`, as the argument of `CBLDCRPC('POLLANYR')`. Thus, hold the value of *data-name-D* of `CBLDCRPC('CALL     ')` to designate a specific response received.

To receive responses after the synchronization point processing in non-transaction processing, specify the corresponding option in rpc_extend_function of the system service definition. For details about rpc_extend_function, see the manual *OpenTP1 System Definition*.

- Nonresponse-type RPC (when 1 is specified for *data-name-C*)

`CBLDCRPC('CALL    ')` will return immediately without waiting for completion of the service program processing. The service program is treated as a function which does not return any response. Therefore, the UAP requesting a service cannot determine whether the service program has been performed. With this specification, the response (*data-name-J*) and its length (*data-name-I*) cannot be referenced.

- Chained RPC (when 4 is specified for *data-name-C*)

  `CBLDCRPC('CALL    ')` will not return until a response is returned or a communication error occurs. When the same service is requested more than once using a chained RPC, the process used for the first request can be used for the subsequent requests.

  There are the following restrictions on the use of chained RPCs:

  1. The shutdown state of the user server or service cannot be detected by the second and subsequent calls of `CBLDCRPC('CALL    ')`.
  2. The entire user server enters shutdown state if an error occurs during the service program processing of the second and subsequent calls of `CBLDCRPC('CALL    ')`. Services do not enter shutdown state individually.

(2) Error cases of CBLDCRPC('CALL ')

Reasons why `CBLDCRPC('CALL    ')` returns with an error are explained below.

■ If the OpenTP1 at the node containing the server UAP is not active

If the OpenTP1 to which the service is requested is not running, `CBLDCRPC('CALL    ')` returns with an error and sets one of the following status code values:

`00306`

`00314`

`00315`

`00320`

■ If the server UAP is not active

When the server UAP is a multiserver, the service request is dealt with a new process which is activated by the OpenTP1 even if the server UAP is being terminated abnormally or being partially recovered. However,`CBLDCRPC('CALL    ')` returns with an error in the following cases:

1. No service request can be addressed to the SPP in shutdown state. If the service group is shut down, `CBLDCRPC('CALL    ')` returns with an error, giving the status code `00312`.
2. If the SPP is being terminated or has been terminated by the stop command for

the user server (`dcsvstop` command) or for OpenTP1 (`dcstop` command), `CBLDCRPC('CALL    ')` returns with an error and sets one of the following status code values:

`00310`

`00312`

`00313`

Which status code returns depends on the timing when `CBLDCRPC('CALL    ')` was called.

3. If the OpenTP1 is being started, `CBLDCRPC('CALL    ')` returns with an error, giving the status code `00320`. In this case, a service may be requested normally after activation of the server UAP or OpenTP1 is completed. Since activation of the OpenTP1 is completed when a message log with the message ID KFCA01809-I is output, request a service again after this message appears.

- When a service request is issued in an environment where internode load-balancing and extended internode load-balancing are in use:

If a service request is issued in an environment where internode load- balancing and extended internode load-balancing are in use and the requested service at the addressed node is unavailable because of a closed schedule, the OpenTP1 automatically transfers that service request to another node. If one of the following events occurs, however, `CBLDCRPC('CALL    ')` returns with an error, giving the status code `00327`:

1. The service was requested from a transaction process and the transaction attribute of the service at the transfer destination node disagrees with that of the closed service.
2. The version of the OpenTP1 residing in the transfer destination node is earlier than the version of the OpenTP1 residing in the node that requested the service.

If the above error return occurs, take actions to reach consistency among:

1. Transaction attributes of SPPs that participate in internode load-balancing and extended internode load-balancing.
2. Versions of OpenTP1 that participate in internode load-balancing and extended internode load-balancing.

- When a service request is addressed to the server that receives requests from a socket

The server that receives requests from a socket controls message congestion according to the specified values for `max_socket_msg` and `max_socket_msglen` in the user service definition. It is probable that service requests cannot be accepted if a message reaches the defined value. In this case, `CBLDCRPC('CALL    ')` returns with an error, giving the status code `00356`. If this status code is returned, the client UAP can sometimes reissue the service request successfully after waiting for a while.

- When a chained RPC is used

  If `CBLDCRPC('CALL    ')` which is not a transaction is called from the UAP using a chained RPC which is processed as a transaction to the same server UAP, `CBLDCRPC('CALL    ')` returns with an error, giving the status code `00367`.

- When the online tester is used

  If the online tester is in use and `CBLDCRPC('CALL    ')` is called from a UAP in test mode to a UAP in nontest mode or vice versa, `CBLDCRPC('CALL    ')` returns with an error, giving the status code `00366`.

- When the security facility is used

  If the desired service is protected with the security facility and the client UAP which called `CBLDCRPC('CALL    ')` does not have the access permission for the SPP, `CBLDCRPC('CALL    ')` returns with an error, giving the status code `00370`.

(3) Timing when CBLDCRPC('CALL   ') results in error

The following explains the timing when an error is returned to the client UAP if the SPP to which the service request is addressed terminates abnormally.

- Synchronous response-type RPC or chained RPC (when 0 or 4 is specified for *data-name-C*)

  If an SPP which executes a service terminates abnormally before completion of the processing, `CBLDCRPC('CALL    ')` returns with an error, giving the status code `00307`. If an infinite period of time is specified in the `watch_time` operand in the user service definition of the client UAP, the function returns with an error, giving the status code `00314`.

- Asynchronous response-type RPC (when 2 is specified for *data-name-C*)

  If an SPP which executes a service terminates abnormally before completion of the processing, `CBLDCRPC('POLLANYR')` returns with an error, giving the status code `00307`. If an infinite period of time is specified in the `watch_time` operand in the user service definition of the client UAP, the function returns with an error, giving the status code `00314`.

- Nonresponse-type RPC (when 1 is specified for *data-name-C*)

  The client UAP cannot detect abnormal termination of server UAP.

- When CBLDCRPC('CALL   ') results in error due to time monitoring of the client UAP

  In the following cases, `CBLDCRPC('CALL    ')` returns with an error, giving the status code `00307`, after the time specified in the `watch_time` operand in the user service definition of the client UAP has elapsed:

  - The entire OpenTP1 at the node containing the SPP terminates abnormally.

- An error occurs before the server UAP receives service request data or before the client UAP receives the result after the server UAP processing is completed.

(4) Specification for reexecuting the service request if CBLDCRPC('CALL ') results in error

Even if the OpenTP1 to which the service request is issued is not active because it is being started or is engaged in system switching, you can have the OpenTP1 re-execute the requested search and service request transmission without treating `CBLDCRPC('CALL    ')` processing as an error.

To re-execute the requested search and service request transmission, specify `Y` in the `rpc_retry` operand in the system common definition. You use the `rpc_retry_count` and `rpc_retry_interval` operands to specify the re-executions count and re-execution interval, respectively, for a requested search and service request transmission. If this count value exceeds the re-executions count value specified in the system common definition, `CBLDCRPC('CALL    ')` returns with an error and sets one of the following status code values:

`00301`

`00306`

`00310`

`00314`

`00315`

`00320`

(5) When a priority is given to a service request

To specify a schedule priority for a service request, call `CBLDCRPC('SETSVPRI')` immediately before `CBLDCRPC('CALL    ')`. If no schedule priority is specified, the priority of the service request is determined according to the default interpretation of the schedule service.

(6) Difference between status codes 00310 and 00306

These status codes are returned if the user server corresponding to the service group name is not found.

- `00310`

  Indicates the user server is not found after searching all nodes specified for `all_node` in the system common definition.

- `00306`

  Indicates a communication error occurred on one or more nodes specified for `all_node` during the search. This return value may indicate the corresponding

OpenTP1 system is not found.

(7) Specification for the return of status code 00378

You can use the status code `00378` instead of `00307` or `00314` to check whether the SPP that was asked to offer its service abnormally terminated before processing was completed. For this purpose, assign `00000001` to the `rpc_extend_function` operand in the user service definition. With this specification, the status code `00378` will return if the above error occurs. If nothing or `00000000` is assigned to the `rpc_extend_function` operand, the status code `00307` or `00314` will be returned, instead of `00378`.

(8) Relationship between status codes and synchronization point processing

The relationship between status codes of `CBLDCRPC('CALL    ')` and synchronization point processing (commitment and rollback) is explained below. The description applies to the service request which is a transaction, rather than the service request which is not a transaction (including the case when 32 is added to the value of *data-name-C* indicating the RPC mode).

- When commitment is performed even though CBLDCRPC('CALL   ') returns with an error

  The status code `00307` may be returned due to abnormal termination of the service program which the service request is addressed, a node error, or network error. However, when the client UAP is not a transaction, the service program to which the service request is addressed may terminate normally and database may be updated.

- Status codes which require rollback processing

  If `CBLDCRPC('CALL    ')` called from a transaction returns with an error, some status codes always require rollback processing for the transaction (the server UAP enters `rollback_only` state). In this case, rollback processing is always performed even if either of the COBOL-UAP creation programs for commitment and rollback is used. The following status codes of `CBLDCRPC('CALL    ')` always require rollback processing for the transaction:

  `00311`

  `00317`

  `00319`

  `00326`

(9) Notes on requesting a service

1. Define the service group name and service name at server UAP environment setup. These names are set in `CBLDCRPC('CALL    ')`. If a service is requested while invalid service group name or service name is set in `CBLDCRPC('CALL`

`')`, the function returns with an error, giving the status code `00310` or `00311`. If the service program does not return response, `CBLDCRPC('CALL    ')` does not return with an error, giving the status code `00310`.

2. The process of the server UAP is different from that of the client UAP. Therefore, the following matters are different from ordinary procedure calls:
   - Attributes which are given to the process of the client UAP by the OS (environment variables, schedule priority (`nice` value), etc.) are not passed on to the server UAP.
   - Environment settings of the OpenTP1 specified at the node of the client UAP (existence of specification of transaction attribute, time limit of transaction branch, schedule priority, etc.) are not passed on to the OpenTP1 of the server UAP.
3. Do not specify the same buffer area for the input parameter and the response from the service program.
4. If 1 is specified for *data-name-C*, the following status codes will not return:
   - Errors which never occur

     `00309`

     `00319`
   - Errors which cannot be detected even though they could occur

     `00311`

     `00312`

     `00313`

     `00316`

     `00318`

     `00320`

     `00370`

   In addition, OpenTP1 does not output a message when an error occurs. If errors must be detected, consider specifying `0` to *data-name-C* (synchronous-response-type RPC).
5. When a service group is requested by `CBLDCRPC('CALL    ')` from a transaction, an SPP is occupied until the transaction terminates. When the same service is requested more than once by `CBLDCRPC('CALL    ')` from the same transaction, perform the following:
   - Re-estimate the values specified for the `balance_count` operand and `parallel_count` operand in the user service definition according to the

number of usages.

- Request a service by using chained RPCs so that the number of processes will not increase.

If the values specified for the `balance_count` operand and `parallel_count` operand are incorrect, the transaction will shut down abnormally and a deadlock may occur.

6. When an asynchronous response-type RPC is used, the server UAP may be occupied until `CBLDCRPC('POLLANYR')` receives all asynchronous responses or `CBLDCRPC('DISCARDF')` rejects the receiving of asynchronous responses. This may occur regardless of whether it is a transaction or not. Increase the number of resident processes according to how many times an asynchronous response-type RPC is used.

   An asynchronous response-type RPC requires many resources in addition to occupying the server UAP. To prevent responses from degrading performance of UAP processing and activation of unnecessary SPPs, ensure that responses are received or the receiving of responses is rejected after `CBLDCRPC('CALL    ')` of an asynchronous response-type RPC is used.

7. To receive a response after using the service request in asynchronous-response-type RPC for more than one time continuously, specify a different response storage area as shown below for each service request when requesting an asynchronous-response-type RPC service. If the same area is specified, the response that comes in later will be overwritten and an accurate response cannot be received.

```
01 unique-name-3-1.
   02 data-name-I1 PIC S9(9) COMP.
   02 data-name-J1 PIC X(n).
01 unique-name-3-2.
   02 data-name-I2 PIC S9(9) COMP.
   02 data-name-J2 PIC X(n).
01 unique-name-3-3.
   02 data-name-I3 PIC S9(9) COMP.
   02 data-name-J3 PIC X(n).
```

8. The server UAP (SPP) that requested a service using an asynchronous response-type RPC sends a response soon after the service function is executed, regardless of whether the process that executed the asynchronous response-type RPC issued the `CBLDCRPC('POLLANYR')`. If the same asynchronous response-type RPC is executed numerous times simultaneously without the `CBLDCRPC('POLLANYR')` being issued, the response sent by the SPP may stay in the TCP/IP buffer and the SPP may fail to send a response. If the SPP fails to send a response, no response can be received from the SPP even if the source of the asynchronous response-type RPC issues the `CBLDCRPC('POLLANYR')`.

9. If a large number of asynchronous response-type RPCs or non-response type RPCs having the transaction attribute are executed, messages about transactions sent by the SPP can no longer be received. In this case, the transactions may roll back.

(10) When a service is requested with domain qualification

Specifying a service group name with domain qualification enables requesting an OpenTP1 service in the DNS domain. Specify the service group name suffixed by an at mark (@) and the DNS domain name for domain qualification.

- Notes on requesting a service with domain qualification

1. To request a service with domain qualification, add (256 + 32) to the value of *data-name-C* of `CBLDCRPC('CALL    ')` indicating the RPC mode. If the service group name with domain qualification is specified without addition, `CBLDCRPC('CALL    ')` returns with an error, giving the status code `00310`.
2. If an RPC with domain qualification is used, a transaction cannot be extended even if the process which called `CBLDCRPC('CALL    ')` is a transaction. Therefore, to request a service with domain qualification from a transaction, add (256 + 32) to the value of *data-name-C* of `CBLDCRPC('CALL    ')` indicating the RPC mode in order not to have the transaction extended. When the local domain is specified for the domain name, the transaction also cannot be extended.
3. When an RPC with domain qualification is used, a service request can be addressed only to a server that receives requests from a queue, rather than a server that receives requests from a socket.
4. A service request with domain qualification is sent to the domain-alternate schedule service which is activated on the host registered with the `namdomainsetup` command. Obtain the port number of the domain-alternate schedule service from `/etc/services`. If an error occurs while transferring the service request and multiple host names are registered with the `namdomainsetup` command, the service request is transferred to other hosts sequentially. Even if the RPC with domain qualification terminates normally, an error may occur during transfer to the domain-alternate schedule service.

- Preparation for requesting a service with domain qualification

Perform the following environment setup for an RPC with domain qualification:

1. Register the name of the host on which the domain-alternate schedule service is activated by using the `namdomainsetup` command.
2. Define as follows the port number of the domain-alternate schedule service in `/etc/services` of the host on which the OpenTP1 which requests a service with domain qualification is activated:
`OpenTP1scd port-number/tcp`

3. Specify the well-known port of the domain-alternate schedule service for the `scd_port` operand in the schedule service definition for the OpenTP1 which activates the domain-alternate schedule service.

Example

Client UAP which executes the service `ADD`

```
01  unique-name-1.
    02  data-name-A      PIC X(8) VALUE 'CALL    '.
    02  STATUS-CODE      PIC X(5) VALUE SPACES.
    02  FILLER           PIC X(3).
    02  FLAGS            PIC S9(9) COMP VALUE ZERO.
    02  DESCRIPTOR       PIC S9(9) COMP VALUE ZERO.
    02  SERVICE-NAME     PIC X(32) VALUE SPACES.
    02  GROUP-NAME       PIC X(32) VALUE SPACES.
01  unique-name-2.
    02  NAME-LENG        PIC S9(9) COMP.
    02  NAME             PIC X(32) VALUE SPACES.
01  unique-name-3.
    02  RESULT-LENG      PIC S9(9) COMP.
    02  RESULT           PIC X(20).
*
MOVE 'ADDRESS-BOOK' TO GROUP-NAME.
MOVE 'ADD' TO SERVICE-NAME.
MOVE 'SATO' TO NAME.
MOVE 4 TO NAME-LENG.
MOVE 20 TO RESULT-LENG.
CALL 'CBLDCRPC' USING  unique-name-1  unique-name-2
                       unique-name-3.
IF STATUS-CODE NOT = '00000' THEN
   DISPLAY 'FAILED'
END-IF.
STOP RUN.
```

Server UAP which executes the service `ADD`

```
PROGRAM-ID. ADD.
DATA DIVISION.
LINKAGE SECTION.
01  unique-name-1.
    02  ADD-NAME    PIC X(20).
01  unique-name-2.
    02  NAME-LENG   PIC S9(9) COMP.
01  unique-name-3.
    02  RESULT      PIC X(20).
01  unique-name-4.
    02  RESULT-LENG PIC S9(9) COMP.
         :
         :
PROCEDURE DIVISION USING unique-name-1  unique-name-2
                          unique-name-3  unique-name-4.
         :
* Processes the contents of the received unique-name-1
         :
MOVE 'ADD-COMPLETE' TO RESULT.
MOVE 12 TO RESULT-LENG.
EXIT PROGRAM.
```

Note

Assume that you want to perform a transactional RPC on an OpenTP1 system other than the domain specified in the `all_node` clause of the system common definition. In this case, you must ensure that the node identifiers (`node_id` clause of the system common definition) of all OpenTP1 systems in the local domain and remote domain are unique. In addition, all the OpenTP1 systems must be version 03-02 or later. If these conditions are not met, the transaction may not recover properly.

CBLDCRPC('CLOSE   ') - Terminate an application program

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(8) VALUE 'CLOSE   '.
    02  data-name-B     PIC X(5).
    02  FILLER          PIC X(3).
    02  data-name-C     PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCRPC('CLOSE   ')` closes the environment for using various types of OpenTP1 COBOL-UAP creation programs. OpenTP1 COBOL-UAP creation programs cannot be called after `CBLDCRPC('CLOSE   ')`.

`CBLDCRPC('CLOSE   ')` must be called from the main program. Call it only once in the process.

`CBLDCRPC('CLOSE   ')` also informs OpenTP1 of normal termination of a UAP. If a UAP terminates without `CBLDCRPC('CLOSE   ')` called, OpenTP1 assumes that the UAP terminated abnormally. Consequently, the service group might be shut down or the process might be restarted. To make matters worse, various OpenTP1 resources might not be released, which affects the subsequent processing. If `CBLDCRPC('OPEN    ')` is called from any UAP used with OpenTP1, `CBLDCRPC('CLOSE   ')` must be called before the UAP terminates with `STOP RUN`.

Call `CBLDCRPC('CLOSE   ')` even if `CBLDCRPC('OPEN    ')` returns with an error.

After `CBLDCRPC('CLOSE   ')` is called, `CBLDCRPC('OPEN    ')` cannot be called.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'CLOSE`∆∆∆`'` for the request code indicating UAP termination.

- *data-name-C*

  Specify `0`.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCRPC('CLTSEND ') - Report data to CUP unidirectionally

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING unique-name-1  unique-name-2
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A  PIC X(8)  VALUE 'CLTSEND '.
    02  data-name-B  PIC X(5).
    02  FILLER       PIC X(3).
    02  data-name-C  PIC S9(9) COMP VALUE ZERO.
    02  data-name-D  PIC S9(9) COMP.
    02  data-name-E  PIC X(n).
01  unique-name-2.
    02  data-name-F  PIC S9(9) COMP.
    02  data-name-G  PIC X(n).
```

Description

`CBLDCRPC('CLTSEND ')` sends data to the CUP unidirectionally.
`CBLDCRPC('CLTSEND ')` sends data specified for *data-name-G* of the length specified for *data-name-F* to the process (CUP) corresponding to the port number of the host specified for *data-name-D* and *data-name-E*. The possible sending data length is in the range of bytes from 0 to `DCRPC_MAX_MESSAGE_SIZE`#.

#: If you used the `rpc_max_message_size` operand, the value of this data area is the value specified in the `rpc_max_message_size` operand and not the value of `DCRPC_MAX_MESSAGE_SIZE` (1 megabyte).

Data sent by `CBLDCRPC('CLTSEND ')` is received by the TP1/Client library function `dc_clt_chained_accept_notification()` or `dc_clt_accept_notification()`. For the function `dc_clt_chained_accept_notification()` or `dc_clt_accept_notification()`, see the *OpenTP1 TP1/Client User's Guide TP1/Client/W, TP1/Client/P* manual.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'CLTSEND'` for the request code indicating that data is reported to UAP unidirectionally.

- *data-name-C*

  Specify `0`.

- *data-name-D*

  Specify the number of the port to which data is sent.

- *data-name-E*

  Specify the name of the host to which data is sent. The specified host name must be a 1 to 255 byte character string. End the character string with space.

- *data-name-F*

  Specify the length of data to be sent.

- *data-name-G*

  Specify data to be sent.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value specified for the argument is invalid. This error also occurs if the status code (*data-name-A*) is invalid.
`00306`	A network error occurred.
`00302`	`CBLDCRPC('OPEN     ')` was not called.
`00304`	The memory became insufficient.
`00308`	The length of data to be sent exceeds `DCRPC_MAX_MESSAGE_SIZE`#.
`00314`	There is no process at the destination.
	A network error occurred.

#: If you used the `rpc_max_message_size` operand, the value of this data area is the value specified in the `rpc_max_message_size` operand and not the value of `DCRPC_MAX_MESSAGE_SIZE` (1 megabyte).

Notes

1. Use `CBLDCRPC('CLTSEND ')` only when the calling of the TP1/Client function

`dc_clt_chained_accept_notification()` or `dc_clt_accept_notification()` by the process of destination is obvious. If the process of the destination is not active, `CBLDCRPC('CLTSEND ')` returns with an error, giving the status code `00314`.

2. Normal return of `CBLDCRPC('CLTSEND ')` indicates that sending at RPC communication protocol (TCP/IP) level is completed. Therefore, normal termination of `CBLDCRPC('CLTSEND ')` does not guarantee that the data is received normally by the CUP using the function `dc_clt_chained_accept_notification()` or `dc_clt_accept_notification()`.
3. `CBLDCRPC('CLTSEND ')` can report data only to the function `dc_clt_chained_accept_notification()` or `dc_clt_accept_notification()` used by the CUP. Data cannot be sent to SPP processes and local processes.
4. To specify a host name consisting of 80 to 255 characters in TP1/Server Base for AIX, recompile and relink the UAP using `libbetran2.a`.

CBLDCRPC('DISCARDF') - Reject the receiving of processing results

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'DISCARDF'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCRPC('DISCARDF')` specifies that no more responses (which have not been returned) will be received through an asynchronous-response-type RPC. After `CBLDCRPC('DISCARDF')` is used, returned responses are discarded instead of being received.

To stop receiving further processing results of an asynchronous-response-type RPC, use `CBLDCRPC('DISCARDF')`. Otherwise, `CBLDCRPC('POLLANYR')` might receive unnecessary responses.

Use `CBLDCRPC('DISCARDF')` in the following cases:

- After a response wait timeout occurs, the buffer for holding the processing results is released.
- An asynchronous-response-type RPC has been used more than once, but only the first response is necessary.

Data areas whose values are set in the UAP

- *data-name-A*

Specify `VALUE 'DISCARDF'` for the request code indicating that the receiving of processing results is refused.

- *data-name-C*

Specify `0`.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCRPC('DISCARDS') - Reject acceptance of specific processing results

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCRPC' USING unique-name-1
```

- DATA DIVISION specification

```
01 unique-name-1.
   02 data-name-A   PIC X(8) VALUE 'DISCARDS'.
   02 data-name-B   PIC X(5).
   02 FILLER        PIC X(3).
   02 data-name-C   PIC S9(9) COMP VALUE ZERO.
   02 data-name-D   PIC S9(9) COMP.
```

Description

`CBLDCRPC('DISCARDS')` declares that the UAP will no longer receive a specific response which has not yet returned. This function is applicable when an asynchronous-response type RPC is used. To specify the asynchronous response whose acceptance is to be rejected, specify the descriptor returned when an asynchronous-response type RPC returned in *data-name-D*. Of the responses that return after `CBLDCRPC('DISCARDS')` is called, responses having the same descriptor as the specified descriptor are discarded without being received.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'DISCARDS'` for the request code indicating the acquisition of the node address of a gateway.

- *data-name-C*

  Specify `0`.

- *data-name-D*

  Specify the descriptor returned when `CBLDCRPC('CALL   ')` of an asynchronous-response type RPC terminated normally.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
`00302`	`CBLDCRPC('OPEN   ')` was not called.
`00322`	The descriptor specified for *data-name-D* does not exist. An asynchronous-response type RPC corresponding to the specified descriptor was not sent, or a response has already been received through an asynchronous-response type RPC, or acceptance of a response was rejected.

CBLDCRPC('GETCLADR') - Acquire the node address of a client UAP

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'GETCLADR'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
    02  data-name-D    PIC S9(9) COMP.
```

Description

`CBLDCRPC('GETCLADR')` allows the server UAP to acquire the address of the node at which the client UAP process is working. Security checking for the client UAP can be performed using the address obtained by this program.

The address obtained by `CBLDCRPC('GETCLADR')` cannot be used for sending a service response or error response.

`CBLDCRPC('GETCLADR')` must be called from a service program. Otherwise, processing is unpredictable.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'GETCLADR'` for the request code indicating that the node address of the client UAP be acquired.

- *data-name-C*

  Specify `0`.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-D*

  The node address of the client UAP is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
Other than the above	An unprecedented error (e.g., program damage) occurred.

Note

When both the following conditions occur, the node address of the client UAP returned by `CBLDCRPC('GETCLADR')` may differ from the node address actually used by the client UAP during communication.

- A service request was accepted using the remote API facility.
- The host containing the client UAP is a multi-homed host mode.

CBLDCRPC('GETERDES') - Acquire the descriptor of an asynchronous response-type RPC request which has encountered an error

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCRPC' USING unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'GETERDES'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
    02  data-name-D    PIC S9(9) COMP.
```

Description

`CBLDCRPC('GETERDES')` acquires the descriptor of a service request which has encountered an error when it is called just after `CBLDCRPC('POLLANYR')` without a particular asynchronous response specified returns with an error.

It can acquire the descriptor only when the error has occurred on the SPP. If an error has occurred on the `CBLRPC('POLLANYR')` caller, `CBLDCRPC('GETERDES')` cannot acquire the descriptor.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'GETERDES'` for the request code indicating descriptor acquisition.

- *data-name-C*

  Specify `0`.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-D*

  The descriptor is returned if it is acquired. Otherwise, 0 is set here.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value specified for the data name is invalid. This error also occurs when the request code (*data-name-A*) is invalid.

CBLDCRPC('GETGWADR') - Acquire the node address of a gateway

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCRPC' USING unique-name-1
```

- DATA DIVISION specification

```
01 unique-name-1.
    02  data-name-A   PIC X(8) VALUE 'GETGWADR'.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-C   PIC S9(9) COMP VALUE ZERO.
    02  data-name-D   PIC S9(9) COMP.
```

Description

`CBLDCRPC('GETGWADR')` acquires the node address of a gateway from the server UAP when a service request was received from a client UAP via a gateway, such as the application gateway FireWall.

The server UAP can acquire the node address of the gateway when a service was requested using the remote API facilities.

A service response or error response cannot be sent using the address that is returned.

Call `CBLDCRPC('GETGWADR')` from the service program. Processing is not guaranteed if `CBLDCRPC('GETGWADR')` is called from a program other than the service program.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'GETGWADR'` for the request code indicating the acquisition of the node address of a gateway.

- *data-name-C*

  Specify `0`.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-D*

  The node address is returned. The value 0 is set when the remote API facilities were not used.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
`00302`	`CBLDCRPC('GETGWADR')` was not called from the service program.

CBLDCRPC('GETSVPRI') - Reference the schedule priority of a service request

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'GETSVPRI'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCRPC('GETSVPRI')` references that schedule priority of a service request which was set by `CBLDCRPC('SETSVPRI')`. The value obtained by `CBLDCRPC('GETSVPRI')` remains unchanged until the UAP calls `CBLDCRPC('SETSVPRI')` again.

`CBLDCRPC('GETSVPRI')` returns the default value (4) to *data-name-C* in the following cases:

- The UAP has not called `CBLDCRPC('SETSVPRI')`.
- `CBLDCRPC('SETSVPRI')` has been called with 0 specified for *data-name-C*.

Data areas whose values are set in the UAP

- *data-name-A*

Specify `VALUE 'GETSVPRI'` for the request code indicating that the schedule priority of the service request be referenced.

Data areas whose values are returned from OpenTP1

- *data-name-B*

A status code of 5 digits is returned.

- *data-name-C*

The schedule priority is returned in a range from 1 to 8.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCRPC('GETWATCH') - Reference the service response waiting interval

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'GETWATCH'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCRPC('GETWATCH')` references the current response waiting interval of a service request. `CBLDCRPC('GETWATCH')` is used for saving the current value of the response waiting interval of a service request before temporarily changing it using `CBLDCRPC('SETWATCH')`.

`CBLDCRPC('GETWATCH')` returns the service response waiting interval changed by `CBLDCRPC('SETWATCH')`. When the interval has not been changed, the following value is returned:

- For TP1/Server Base: Value of the `watch_time` operand in the system common definition
- For TP1/LiNK: 180 seconds

The value obtained by `CBLDCRPC('GETWATCH')` can be used by the OpenTP1 remote service request (`CBLDCRPC('CALL     ')`).

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'GETWATCH'` for the request code indicating that the service response waiting interval be referenced.

- *data-name-C*

  Specify `0`.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-C*

  The current service response waiting interval is returned. If 0 is returned, it indicates that a response will be awaited indefinitely.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCRPC('OPEN   ') - Start an application program

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'OPEN    '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCRPC('OPEN    ')` prepares to use the various types of OpenTP1 service programs. `CBLDCRPC('OPEN    ')` must be called in the main program. `Call CBLDCRPC('OPEN    ')` only once in the process before any other OpenTP1 COBOL-UAP creation program. To initialization procedure in the main program is as follows:

1. Open the entry point for communication between processes.
2. Acquire shared memory used with OpenTP1.
3. Report the UAP start to OpenTP1 to request OpenTP1 to supervise processes.
4. Initialize the OpenTP1 facilities to be used according to the user service definition.

If the transaction attribute is specified in the user service definition, the OpenTP1 transaction service and the process service must be in progress at the node. `CBLDCRPC('OPEN    ')` can be called only after OpenTP1 starts normally when the OS starts or after OpenTP1 is started normally by entering the dcstart command. If `CBLDCRPC('OPEN    ')` is called before the normal start of OpenTP1, the program returns with the status code `00315`. In this case, SPP services cannot be requested.

UAP trace is acquired for all OpenTP1 COBOL-UAP creation programs called after `CBLDCRPC('OPEN    ')` terminates normally. If `CBLDCRPC('OPEN    ')` returns with an error, the UAP trace is not always acquired.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'OPEN`ΔΔΔΔ`'` for the request code indicating UAP service start.

- *data-name-C*

  Specify `0`.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
`00302`	`CBLDCRPC('OPEN    ')` has already been called.
`00303`	Initialization was unsuccessful. OpenTP1 COBOL-UAP creation programs can no longer be called.
`00315`	OpenTP1 of the node at which the UAP exists was not executed.
`00369`	The stand-by user server received a stand-by termination request.
`00371`	When OpenTP1 uses the security facility, an error occurred during initialization of security environment.

CBLDCRPC('POLLANYR') - Receive processing results in asynchronous mode

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'POLLANYR'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
    02  data-name-D    PIC S9(9) COMP.
    02  data-name-E    PIC S9(9) COMP.
    02  data-name-F    PIC S9(9) COMP.
```

Description

`CBLDCRPC('POLLANYR')` receives the processing results of a service requested through an asynchronous-response-type RPC.

To receive a particular asynchronous response, specify 1 or 17 for *data-name-C*. If one of these values is specified, `CBLDCRPC('POLLANYR')` receives the response from an asynchronous-response-type RPC which has returned the descriptor specified by *data-name-E*.

To receive any asynchronous response, specify `0` or `16` for *data-name-C*. In this case, the value assigned to *data-name-E* is ignored. When `CBLDCRPC('POLLANYR')` with 0 or 16 specified for *data-name-C* normally ends, it returns the same value as the descriptor of the asynchronous response it has received.

`CBLDCRPC('POLLANYR')` returns in the following cases:

- A response is received from an asynchronous-response-type RPC.
- A response wait timeout occurs. (The response wait time specified for *data-name-F* has passed.)

When `CBLDCRPC('POLLANYR')` terminate normally, a response is set to `CBLDCRPC('CALL    ')` in asynchronous-response-type RPC.

The following items are described after the list of status codes. See each description for details on `CBLDCRPC('POLLANYR')`.

(1) *data-name-F* of `CBLDCRPC('POLLANYR')`

(2) Timing when `CBLDCRPC('POLLANYR')` results in error

(3) Specification for the return of status code `00378`

(4) Relationship between status codes and synchronization point processing

(5) When a response cannot be received by `CBLDCRPC('POLLANYR')`

(6) Notes on using `CBLDCRPC('POLLANYR')`

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'POLLANYR'` for the request code indicating processing results are asynchronously received.

- *data-name-C*

  Specify one of the following:

  `0`

  The wait time is specified in seconds and `CBLDCRPC('POLLANYR')` will receive any asynchronous response.

  `1`

  The wait time is specified in seconds and `CBLDCRPC('POLLANYR')` will receive the response from an asynchronous-response-type RPC which returns the descriptor specified by *data-name-E*.

  `16`

  The wait time is specified in milliseconds and `CBLDCRPC('POLLANYR')` will receive any asynchronous response.

  `17`

  The wait time is specified in milliseconds and `CBLDCRPC('POLLANYR')` will receive the response from an asynchronous-response-type RPC which returns the descriptor specified by *data-name-E*.

- *data-name-E*

  Specify the descriptor which was returned when `CBLDCRPC('CALL    ')` (2 specified for *data-name-C*) carried on an asynchronous-response-type RPC terminated normally. If 0 or 16 is specified for *data-name-C*, the value specified here is ignored.

- *data-name-F*

  Specify the wait time in seconds from the calling of `CBLDCRPC('POLLANYR')` to the return of a response. The specified wait time must be in the range from -1 to the maximum value which can be indicated by S9(9) COMP.

When `CBLDCRPC('POLLANYR')` receives an asynchronous response, the response waiting interval specified in the UAP is not referenced.

If 0 is specified here, 0 or 1 is specified for *data-name-C*, and no response is returned, then `CBLDCRPC('POLLANYR')` will immediately return with the status code `00307`. If 16 or 17 is specified for *data-name-C*, the wait time will be 50 milliseconds.

When -1 is specified, `CBLDCRPC('POLLANYR')` continues to wait until a response is returned.

- *data-name-D*

  Specify `0`.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-D*

  The descriptor of the received asynchronous response is returned. This descriptor is returned when `CBLDCRPC('POLLANYR')` with 0 or 16 specified for *data-name-C* ends normally. If `CBLDCRPC('POLLANYR')` with 1 or 17 specified for *data-name-C* ends normally, 0 is set here.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00321`	The results of processing for the service requested with asynchronous response-type RPCs are received completely.
`00322`	The descriptor specified for *data-name-E* does not exist. This value is returned when 1 is specified for *data-name-C*.
`00301`	The value specified for the data-name is invalid. This error also occurs if the status code (*data-name-A*) is invalid.
`00302`	`CBLDCRPC('OPEN    ')` was not called.
`00304`	The memory became insufficient.
`00306`	A network error occurred.
`00307`	`CBLDCRPC('CALL    ')` encountered timeout.
	An SPP to which the service request was addressed terminated abnormally before completion of the requested service.

Status code	Explanation
`00308`	The input parameter length specified for *data-name-G* of `CBLDCRPC('CALL    ')` exceeded the maximum.
`00309`	The returned response is longer than the area prepared by the client UAP.
`00310`	The service group name specified for *data-name-F* of `CBLDCRPC('CALL    ')` is not defined.
`00311`	The service name specified for *data-name-E* of `CBLDCRPC('CALL    ')` is not defined.
`00312`	The service group containing the service of which name is specified for *data-name-E* of `CBLDCRPC('CALL    ')` is in shutdown state.
`00313`	The service specified for *data-name-E* of `CBLDCRPC('CALL    ')` is being terminated.
`00314`	The UAP process of the service specified for *data-name-E* of `CBLDCRPC('CALL    ')` is not active.
	An SPP to which the service request was addressed terminated abnormally before completion of the requested service when -1 is specified for *data-name-F*.
`00315`	The OpenTP1 at the node containing the service specified for *data-name-E* of `CBLDCRPC('CALL    ')` is not active. The cause may be one of the following: abnormal termination, being-suspended, being-terminated, or communication error.
`00316`	A system error occurred in the specified service for `CBLDCRPC('CALL    ')`.
`00317`	The memory became insufficient in the specified service for `CBLDCRPC('CALL    ')`.
`00318`	A system error occurred.
`00319`	The length of the response returned from the service function to the OpenTP1 is not in the range from 1 to `DCRPC_MAX_MESSAGE_SIZE`[#].
`00320`	The OpenTP1 at the node to which the service request is addressed is being started.
`00323`	The memory became insufficient. If this status code is returned, the transaction branch cannot be committed.
`00324`	A system error occurred. If this status code is returned, the transaction branch cannot be committed.
`00325`	A system error occurred when the specified service was executed. If this status code is returned, the transaction branch cannot be committed.
`00326`	The returned response is too large to be stored in the area prepared by the client UAP. If this status code is returned, the transaction branch cannot be committed.

Status code	Explanation
`00327`	The transaction attributes of multiple SPPs do not match in an environment where the inter-node load-balancing facility and the extended internode load-balancing facility are in use. This status code will be returned only when the service request is addressed to an SPP which uses the inter-node load-balancing facility and the extended internode load-balancing facility.
`00328`	The domain name of the service group name with domain qualification is invalid.
`00329`	When a service is requested with domain qualification, the port number of the domain-alternate schedule service is not found.
`00356`	The server that receives requests from the socket to which the service request is addressed cannot receive the service request.
`00366`	When the online tester was in use, a service was requested from a UAP in test mode to an SPP in nontest mode or from a UAP in nontest mode to an SPP in test mode.
`00370`	An SPP to which the service request is addressed is protected with the security facility. The UAP that requests the service by using `CBLDCRPC('CALL     ')` has no access permission for the SPP.
`00372`	The transaction branch cannot be started since it exceeds the maximum number of transaction branches which can be activated concurrently.
	The transaction branch cannot be started since it exceeds the maximum number of child transaction branches which can be activated from one transaction branch.
	Transaction branching cannot start because the resource manager (RM) has encountered an error.
`00378`	The SPP that was asked to offer its service abnormally terminated before processing was completed. This status code will be returned only when `00000001` is assigned to the `rpc_extend_function` operand in the user service definition of the client UAP. If nothing or `00000000` is assigned to the `rpc_extend_function` operand, the status code `00307` or `00314` will be returned, instead of `00378`.

#: If you used the `rpc_max_message_size` operand, the value of this data area is the value specified in the `rpc_max_message_size` operand and not the value of `DCRPC_MAX_MESSAGE_SIZE` (1 megabyte).

(1) data-name-C of CBLDCRPC('POLLANYR')

The monitoring time for receiving an asynchronous response is reset each time a response is returned. Therefore, when a specific asynchronous response received is designated (If 1 or 17 is specified for *data-name-C*), a response may be received even if the time specified for *data-name-F* has elapsed. Alternatively, `CBLDCRPC('POLLANYR')` may not return with an error, giving the status code `00307` even if the time specified for *data-name-F* has elapsed.

(2) Timing when CBLDCRPC('POLLANYR') results in error

The following explains the timing when an error is returned from the client UAP if the SPP to which the service request is addressed terminates abnormally.

If an SPP to execute a service terminates abnormally before completion of the processing, `CBLDCRPC('POLLANYR')` returns with an error, giving the status code `00307`. If -1 is specified for *data-name-F* of `CBLDCRPC('POLLANYR')`, `CBLDCRPC('POLLANYR')` returns with an error, giving the status code `00314`.

- When CBLDCRPC('POLLANYR') results in error due to time monitoring for CBLDCRPC('POLLANYR')

  In the following cases, `CBLDCRPC('POLLANYR')` returns with an error, giving the status code `00307`, after the time specified for *data-name-F* of `CBLDCRPC('POLLANYR')` has elapsed:

  - The entire OpenTP1 at the node containing the SPP terminates abnormally.
  - An error occurs before the server UAP receives service request data or before the client UAP receives the result after the server UAP processing is completed.

(3) Specification for the return of status code 00378

You can use the status code `00378` instead of `00307` or `00314` to check whether the SPP that was asked to offer its service abnormally terminated before processing was completed. For this purpose, assign `00000001` to the `rpc_extend_function` operand in the user service definition. With this specification, the status code `00378` will return if the above error occurs. If nothing or `00000000` is assigned to the `rpc_extend_function` operand, the status code `00307` or `00314` will be returned, instead of `00378`.

(4) Relationship between status codes and synchronization point processing

The relationship between status codes of `CBLDCRPC('POLLANYR')` and synchronization point processing (commitment and rollback) is explained below. The description applies to the service request which is a transactions, rather than service requests which are not transactions (including the case when 32 is added to the value of *data-name-C* of `CBLDCRPC('CALL    ')`).

- If commitment is performed even though CBLDCRPC('POLLANYR') returns with an error

  The status code `00307` may be returned due to abnormal termination of the service program which the service request is addressed, a node error, or network error. However, when the client UAP is not a transaction, the SPP which the service request is addressed may terminate normally and database may be updated.

- Status codes which require rollback processing

  If `CBLDCRPC('POLLANYR')` called from a transaction returns with an error, some status codes always require rollback processing for the transaction (the server UAP enters in `rollback_only` state). In this case, rollback processing is always performed even if either of commitment or rollback processing is executed. The following status codes of `CBLDCRPC('POLLANYR')` always require rollback processing for the transaction:

  `00309`

  `00311`

  `00317`

  `00319`

(5) When a response cannot be received by CBLDCRPC('POLLANYR')

`CBLDCRPC('POLLANYR')` cannot receive a response if either of the following COBOL-UAP creation programs is called by the UAP requesting a service with an asynchronous response-type RPC.

1. The receiving of asynchronous responses is rejected by `CBLDCRPC('DISCARDF')`
2. Commitment or rollback processing is performed in the COBOL-UAP creation program for synchronization point processing when a service is requested from a transaction.

The response returned after the above COBOL-UAP creation program is called is discarded. Receive all required asynchronous responses by using `CBLDCRPC('POLLANYR')` before calling the above COBOL-UAP creation program when an asynchronous response-type RPC is used.

(6) Notes on using CBLDCRPC('POLLANYR')

1. When `CBLDCRPC('POLLANYR')` is called with 0 specified for the wait time (0 specified for *data-name-F*), the response may not be received even if it arrives, due to the scheduling of the multi-thread environment. Note that the UAP may fall into an endless loop, which calls `CBLDCRPC('POLLANYR')` to receive all responses with 0 specified for the wait time.
2. If `CBLDCRPC('POLLANYR')` without a specific descriptor identified returns with an error, the descriptor of the response involving the error cannot be identified. Specify 1 or 17 for *data-name-C* if you want to identify the descriptor when `CBLDCRPC('POLLANYR')` returns with an error.

CBLDCRPC('SETSVPRI') - Set a schedule priority of a service request

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'SETSVPRI'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCRPC('SETSVPRI')` sets a priority of a service request. It is used when controlling schedule priorities for individual service requests. The priority set remains unchanged until it is updated by `CBLDCRPC('SETSVPRI')`. Therefore, if service requests are to be used at once with the same priority, use `CBLDCRPC('SETSVPRI')` only once.

The priority set by `CBLDCRPC('SETSVPRI')` will be reported to the server via the schedule queue by `CBLDCRPC('CALL     ')` which is called immediately after `CBLDCRPC('SETSVPRI')`.

If `CBLDCRPC('SETSVPRI')` is not called at all, the value 4, which is the default interpretation of the schedule service, is set as the priority of service requests.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'SETSVPRI'` for the request code indicating that a service request schedule priority be set.

- *data-name-C*

  Specify `0` or a number from 1 to 8 as a service request schedule priority. The specification of *data-name-C* is mandatory. The highest priority is represented by 1 and the lowest priority is represented by 8. If 0 is specified, the default interpretation by the schedule service is in effect. If any other value is specified, `CBLDCRPC('SETSVPRI')` is ignored.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
Other than the above	An unprecedented error (e.g., program damage) occurred.

Notes

1. In the case of a queue-receiving server, the specified service request priority is valid only if `service_priority_control=Y` (priority control in effect) is specified in the user service definition of the server UAP. If priority control is not used on the server UAP to which the service request is addressed, `CBLDCRPC('SETSVPRI')` has no effect even when used.
2. `CBLDCRPC('SETSVPRI')` is invalid if called for a service request specified in `CBLDCRPC('CALL    ')` carried on the second or subsequent chained RPC or `CBLDCRPC('CALL    ')` called to terminate chained RPCs.
3. `CBLDCRPC('CALL    ')` does not reset the value changed by `CBLDCRPC('SETSVPRI')`. To reset the service request priority, call `CBLDCRPC('SETSVPRI')` again with 0 assigned to *data-name-C*.

CBLDCRPC('SETWATCH') - Update the service response waiting interval

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING  unique-name-1
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'SETWATCH'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
```

Description

CBLDCRPC('SETWATCH') changes the response waiting interval of service request. The value set by CBLDCRPC('SETWATCH') remains valid until CBLDCRPC('CLOSE ') is called.

To reset the response waiting interval of service request to the value which was in effect before CBLDCRPC('SETWATCH') is called, supply CBLDCRPC('SETWATCH') with the original value returned by CBLDCRPC('GETWATCH').

CBLDCRPC('SETWATCH') does not affect the value assigned to the watch_time operand in the system common definition.

The value set by CBLDCRPC('SETWATCH') influences only CBLDCRPC('CALL ') which will be called later.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify VALUE 'SETWATCH' for the request code indicating that the response waiting interval of service request be changed.

■ *data-name-C*

Specify a number from 1 to 65535 as the new service response waiting interval. Specify 0 to wait for a response indefinitely.

Data area whose value is returned from OpenTP1

■ *data-name-B*

A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
Other than the above	An unprecedented error (e.g., program damage) occurred.

CBLDCRPC('SVRETRY ') - Retry a service program

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCRPC'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'SVRETRY '.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCRPC('SVRETRY ')` allows you to retry processing for a running service program. To retry a service program, invoke `CBLDCRPC('SVRETRY ')` on the service program to be retried and make it return. After this, the same service program is restarted from the same process.

If a service program invoked from a response-type RPC is retried, the values (response storage area and response length) set by the service program before the retry are invalidated.

If `CBLDCRPC('SVRETRY ')` is invoked after the retry count assigned to the `rpc_service_retry_count` operand in the user service definition is exceeded (or when 0 is assigned to the `rpc_service_retry_count` operand), `CBLDCRPC('SVRETRY ')` returns with an error, giving the status code `00377`. The service program will not be retried. If the service program was invoked by a response-type RPC, `CBLDCRPC('SVRETRY ')` returns the contents of the response storage area to the client UAP.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'SVRETRY∆'` as a request code for service program retry.

- *data-name-C*

  Specify `0`.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`00301`	The value (request code) specified for *data-name-A* is invalid.
`00377`	An attempt was made to invoke `CBLDCRPC('SVRETRY ')` beyond the maximum service retry count assigned to the `rpc_service_retry_count` operand in the user service definition. No more service program can be retried.
`00302`	The condition for invoking `CBLDCRPC('SVRETRY ')` is incorrect. Probable causes include: • `CBLDCRPC('SVRETRY ')` is not invoked from within the service program. • `CBLDCRPC('SVRETRY ')` is invoked within the range of a global transaction.

Notes

1. Before invoking `CBLDCRPC('SVRETRY ')`, make sure that the following conditions are fulfilled. Otherwise, `CBLDCRPC('SVRETRY ')` returns with an error.
   - `CBLDCRPC('SVRETRY ')` must be invoked from within the service program.
   - The running service program must not be within the range of a global transaction.
2. The service program which invokes `CBLDCRPC('SVRETRY ')` can reference data passed from a client UAP, but cannot modify it. If data in an input data area is modified, the system operation is unpredictable.
3. `CBLDCRPC('SVRETRY ')` can be invoked only from a service program which was asked to offer its service using the OpenTP1-specific remote procedure call `CBLDCRPC('CALL  ')`. Processing for other service programs cannot be retried using `CBLDCRPC('SVRETRY ')`.

CBLDCRSV('MAINLOOP') - Start an SPP service

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCRSV'  USING  unique-name-1
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'MAINLOOP'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC S9(9) COMP VALUE ZERO.
```

Description

`CBLDCRSV('MAINLOOP')` starts the receiving of service requests to a service program of the SPP which is being executed in the process. `CBLDCRSV('MAINLOOP')` must be called in the main program. Use `CBLDCRSV('MAINLOOP')` only once in the process.

`CBLDCRSV('MAINLOOP')` does not return until it receives a termination request from OpenTP1. `CBLDCRSV('MAINLOOP')` receives a termination request from OpenTP1 in the following cases:

- Termination processing starts because one of the following OpenTP1 stop commands has been accepted:

  `dcstop` command (normal termination)

  `dcstop-n` command (forced normal termination)

  `dcstop -a` command (planned termination A)

  `dcstop -b` command (planned termination B)

- The following server stop command is entered to start termination processing for the processes of the SPP:

  `dcsvstop` command (normal termination)

- OpenTP1 terminates the processes of the SPP because the number of processes exceeds the maximum number specified in the user service definition.
- Service processing terminates if the SPP is executed by a nonresident process.
- The number of service requests addressed to the service group is reduced if the SPP is subjected to load balancing in a multiserver configuration.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'MAINLOOP'` for the request code indicating SPP service start.

- *data-name-C*

  Specify `0`.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	A termination request was received from OpenTP1. Execute termination processing for the SPP immediately, then call `CBLDCRPC('CLOSE       ')` to enable exit().
`00301`	The value specified for the data-name is invalid. This error also occurs if the request code (*data-name-A*) is invalid.
`00302`	`CBLDCRPC('OPEN      ')` was not called, or `CBLDCMCF('MAINLOOP')` or `CBLDCRSV('MAINLOOP')` was called.
`00303`	The SPP service could not be started.

Notes

`CBLDCRSV('MAINLOOP')` returns when it receives a termination request from OpenTP1. However, `CBLDCRSV('MAINLOOP')` does not return but the process terminates in the following cases:

- The SPP enters termination processing because the OpenTP1 forced termination command (`dcstop -f` command) or server forced termination command (`dcsvstop -f` command) is executed.
- A process terminates abnormally because the UAP or OpenTP1 malfunctions.
- The service program uses `abort ()` or `exit ()`.
- Hardware, the operating system, or OpenTP1 causes an error.

Even if the SPP is created in such a way that termination processing will be executed after `CBLDCRSV('MAINLOOP')` returns normally, the processing is not executed in the above cases.

Real time statistical information service (CBLDCRTS)

This section gives the syntax and other information of the following COBOL-UAP creation programs which are used as real-time statistical information service programs:

- `CBLDCRTS('RTSPUT  ')` - Acquire real-time statistical information for arbitrary section

CBLDCRTS('RTSPUT ') - Acquire real-time statistical information for arbitrary section

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCRTS'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01 unique-name-1.
   02 data-name-A  PIC X(8) VALUE 'RTSPUT  '.
   02 data-name-B  PIC X(5).
   02 FILLER       PIC X(3).
   02 data-name-Z  PIC S9(9) COMP VALUE ZERO.

01 unique-name-2.
   02 data-name-C  PIC S9(9) COMP.
   02 data-name-D  PIC X(1).
   02 FILLER       PIC X(3).
```

Description

`CBLDCRTS('RTSPUT  ')` acquires the execution time and the number of executions for the item assigned to *data-name-C* while the UAP executions over an arbitrary section, as real-time statistical information.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'RTSPUT`ΔΔ`'` for the request code of the command to be executed.

- *data-name-C*

  Specify the ID of the real-time statistical item to be acquired.

  The specified ID must be within the range from 1000000 to 2147483647.

- *data-name-D*

  Specify the operation to be performed.

  `VALUE 'S'`

  Starts measuring the execution time for the item ID assigned to *data-name-C*.

  Note that real-time statistical information acquisition does not start when `CBLDCRTS('RTSPUT  ')` is called with `VALUE'S'` specified.

`VALUE 'E'`

Ends measurement after acquiring the execution time for the item ID assigned to *data-name-C*.

`VALUE '△'`

Specifies that only the number of executions for the item ID assigned to *data-name-C* be acquired. The returned execution time will be 0 (in seconds).

- *data-name-Z*

  Specify `0`.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`07801`	The request code (*data-name-A*) is invalid.
`07802`	The value specified for the data name is invalid.
`07803`	`CBLDCRPC('OPEN ')` was not called
	`CBLDCRTS('RTSPUT  ')` was called with `'S'` assigned to *data-name-D*, but execution time measurement has already started for the item identified by the ID assigned to *data-name-C*.
	`CBLDCRTS('RTSPUT  ')` was called with `'E'` assigned to *data-name-D*, but execution time measurement has not started for the item identified by the ID assigned to *data-name-C*.
`07804`	Information cannot be acquired because the number of items to be acquired exceeds the value assigned to the `rts_item_max` operand in the real-time statistical information service definition.
`07805`	Information cannot be acquired because the number of items to be acquired for each server exceeds the value assigned to the `rts_item_max` operand in the real-time statistical information service definition. If `CBLDCRTS('RTSPUT  ')` returns with this status code, it has already acquired statistical information on individual services or non-service processings.
`07806`	Information cannot be acquired because the number of items to be acquired for each service or non-service processings exceeds the value assigned to the `rts_item_max` operand in the real-time statistical information service definition. If `CBLDCRTS('RTSPUT ')` returns with this status code, it has already acquired statistical information on individual servers.

Status code	Explanation
`07807`	Processing cannot be performed because the process memory is insufficient.
`07808`	The real-time statistical information service is not started.
`07809`	The caller of the function `CBLDCRTS('RTSPUT  ')` has not been registered as a recipient for the acquisition of real-time statistical information on a server or service basis.
`07810`	The UAP is linked with a library whose version is not supported by the currently operating real-time statistical information service.

Notes

1. `CBLDCRTS('RTSPUT  ')` cannot acquire system-wide real-time statistical information.
2. If a UAP that uses multi-server features concurrently calls multiple `CBLDCRTS('RTSPUT  ')` instances for which the same caller service and the same data names are specified from multiple processes, it may not acquire statistical information for some processes. This is because multiple writes occur at the same time as no lock control is in effect on statistical information acquisition.
3. For a UAP that uses the XATMI interface, service-specific real-time statistical information cannot be acquired. All statistical information will be acquired as statistical information for non-service processings.
4. `CBLDCRTS('RTSPUT  ')` does not acquire UAP trace data.
5. This note applies after the function `CBLDCRTS('RTSPUT  '),` called by specifying `'S'` in *data-name-D*, returns status code `07808` or `07809`. If the real-time statistical information service is started and the calling UAP is added as a target of acquisition processing before `CBLDCRTS('RTSPUT  ')` is called by specifying `'E'` in *data-name-D* and specify the same ID (*data-name-C*), the function returns status code `07803`.

TAM file service (CBLDCTAM)

This section gives the syntax and other information of the following COBOL-UAP creation programs which are used as TAM file service programs:

- `CBLDCTAM('ERS '/'ERSR'/'ZRS '/'ZRSR')` - Delete a TAM table record
- `CBLDCTAM('FxxR'/'FxxU'/'VxxR'/'VxxU')` - Input a TAM table record
- `CBLDCTAM('GST ')` - Acquire TAM table status
- `CBLDCTAM('INFO')` - Acquire TAM table information
- `CBLDCTAM('MFY '/'MFYS'/'STR '/'WFY '/'WFYS'/'YTR ')` - Update/add a TAM table record

The COBOL-UAP creation programs for TAM file service (CBLDCTAM) can be used only in UAPs of TP1/Server Base. They cannot be used in UAPs of TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language template for TAM file service (CBLDCTAM) is stored in `DCTAM.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCTAM('ERS '/'ERSR'/'ZRS '/'ZRSR') - Delete a TAM table record

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCTAM' USING  unique-name-1  unique-name-2
                       unique-name-3  unique-name-4
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(4).
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-C    PIC X(32).
    02  FILLER         PIC X(68).
    02  data-name-D    PIC S9(4) COMP.
    02  FILLER         PIC X(398).
    02  data-name-J    PIC S9(9) COMP.
    02  FILLER         PIC X(2).
01  unique-name-2.
    02  data-name-E    PIC X(4).
    02  FILLER         PIC X(3).
    02  data-name-I    PIC X(1).
01  unique-name-3.
    02  data-name-F    PIC X(m).
01  unique-name-4.
    02  data-name-G    PIC X(n).
```

Description

`CBLDCTAM('ERS '/'ERSR'/'ZRS '/'ZRSR')` deletes a record indicated as a key value from a TAM table. The record to be deleted can be saved in the buffer. However, if the program that deletes TAM table record returns with an error, the buffer contents cannot be ensured.

When a record is to be deleted, lock in tables must be enabled with lock for update processing.

If the program that deletes TAM table record returns with an error, all the resources specified in this program are released, and the status before this program was called is regained. However, if an attempt is made to delete a TAM table which was acquired under lock for reference processing before this statement was called, lock for update processing is enabled. (Lock for reference processing is not regained.)

Data areas whose values are set in the UAP

- *data-name-A*

  Specify the TAM ID. The TAM service does not reference the value specified here.

- *data-name-C*

  Specify the name of the TAM table of the record to be deleted. The name can be specified with up to 32 characters. If the specified name comprises less than 32 characters, pad the remaining portion with space.

- *data-name-D*

  Specify a buffer length of 32767 bytes or less if the record to be deleted is to be saved. The buffer length must be equal to or greater than the record length. The setting here is valid only when `VALUE 'ZRSR'` is specified for *data-name-E*.

- *data-name-J*

  Specify a buffer length in the range from 1 to 32767 bytes if the record to be deleted is to be saved. The buffer length must be equal to or greater than the record length. The setting here is valid only when `VALUE 'ZRSR'` is specified for *data-name-E*.

- *data-name-E*

  Specify one of the following request codes:

  `VALUE 'ERS '` or `VALUE 'ZRS '`: The record to be deleted is not saved.

  `VALUE 'ERSR'` or `VALUE 'ZRSR'`: The record to be deleted is saved.

- *data-name-F*

  Specify the key value with the length of the key area of the record to be deleted.

- *data-name-G*

  If the record to be deleted is to be saved, specify the buffer for storing the record. The setting here is invalid if `VALUE 'ERS∆'` or `VALUE 'ZRS∆'` (the record to be deleted is not saved) is specified for *data-name-E*.

- *data-name-I*

  Specify whether to wait for release from lock with one of the following values. The value specified for *data-name-I* is valid when 2 is specified for `tam_cbl_level` in the TAM service definition.

  `VALUE 'W'`: Wait for release from lock

  `VALUE 'N'`: Error return without waiting for release from lock

  If 0 or 1 is specified for `tam_cbl_level` in the TAM service definition, specifying a value for *data-name-I* is unnecessary.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-G*

  If `VALUE 'ERSR'` or `VALUE 'ZRSR'` is specified and record deletion is normally completed, the deleted record is saved.

Status codes

Status code	Explanation
`00000`	The record was deleted normally.
`01701`	The table name specified for *data-name-C* is invalid.
`01702`	The key value specified for *data-name-F* is invalid.
`01704`	The value specified for *data-name-G* is invalid.
`01705`	The buffer length specified for *data-name-D* or *J* is too short.
`01708`	The value specified for *data-name-E* or *I* is invalid.
`01709`	The table specified for *data-name-C* is not a TAM table.
`01710`	The TAM table has not been defined.
`01720`	The TAM service is being terminated.
`01721`	The sequence of accessing the TAM table is invalid.
	The resource manager registration of the object file for control of transactions having a linkage with the UAP is invalid. Alternatively, there is no linkage between the object file for control of transactions and the UAP.
	`atomic_update=N` (nontransaction attribute) is specified in the user service definition of the UAP which called CBLDCTAM.
`01723`	The TAM table was deleted.
`01724`	The TAM table was not loaded.
`01727`	The TAM table is in logical descriptor state.
`01728`	The TAM table is in descriptor state due to an error.
`01730`	Execution is impossible in the access mode of the TAM table specified in the TAM service definition.
`01731`	The specified record does not exist.

Status code	Explanation
`01736`	A lock error occurred. If you specified `0` in the `tam_cbl_level` operand of the TAM service definition, or if you specified `2` in the `tam_cbl_level` operand and set `W` in *data-name-I*, the resource could not be acquired because the wait time specified in the lock service definition reached timeout.
`01737`	A deadlock occurred.
`01760`	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current TAM table.
`01761`	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current OpenTP1 file service.
`01762`	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current TAM service.
`01764`	The record has been damaged.
`01765`	The number of transactions exceeds the maximum number of transactions which can be managed by the TAM service.
`01766`	The number of open character special files exceeds the specified limit.
`01767`	The access permission for special files has not been granted.
`01768`	The access permission for TAM files has not been granted.
`01769`	The memory became insufficient.
`01770`	An input/output error occurred.
`01771`	A transaction service error occurred.
`01772`	The deleting TAM file is protected by the security facility. No ACL exists for the file.
`01773`	The accessing TAM file is protected by the security facility. The UAP attempting to delete a record from the TAM table has no access permission.

Notes

If the process used to delete all records in a hash-formatted TAM table involves repetitions of a sequence that calls for finding a target record using the first-retrieval method and deleting the found record, the CPU load may become high. To delete all records, use the following procedure:

1. Use the first-retrieval method to find a target record and save the key value of the found record as *variable-1*.
2. Search for the next record that matches the key value of *variable-1*.
3. Save the key value of the record found in step 2 above, as *variable-2*.

4. Delete the record with the key value saved as *variable-1*.
5. Save the key value of *variable-2* as *variable-1*.
6. Repeat steps 2 to 5 (search for the next record) until an error occurs in step 2.
7. After an error occurs in step 2, delete the record with the key value last saved as *variable-1*.

CBLDCTAM('FxxR'/'FxxU'/'VxxR'/'VxxU') - Input a TAM table record

Format

■ PROCEDURE DIVISION specification

```
CALL 'CBLDCTAM' USING  unique-name-1  unique-name-2
                       unique-name-3  unique-name-4
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(4).
    02  data-name-B     PIC X(5).
    02  FILLER          PIC X(3).
    02  data-name-C     PIC X(32).
    02  FILLER          PIC X(68).
    02  data-name-D     PIC S9(4) COMP.
    02  FILLER          PIC X(398).
    02  data-name-J     PIC S9(9) COMP.
    02  FILLER          PIC X(392).
01  unique-name-2.
    02  data-name-E     PIC X(4).
    02  data-name-F     PIC X(1).
    02  FILLER          PIC X(2).
    02  data-name-I     PIC X(1).
01  unique-name-3.
    02  data-name-G     PIC X(m).
01  unique-name-4.
    02  data-name-H     PIC X(n).
```

Description

According to the search type specified for *data-name-E*, `CBLDCTAM('FxxR'/'FxxU'/'VxxR'/'VxxU')` inputs a TAM table record for reference or update processing. Table 2-1 shows the relationship between search types and index types.

Table 2-2: Relationship between search types and index types

Search type	**Outline of search processing**	
	Index type: hash format	**Index type: tree format**
'key-value='search	The record having the specified key value is searched for. If the record having the specified key value is not found, an error is returned.	The record having the specified key value is searched for. If the record having the specified key value is not found, an error is returned.

Search type	Outline of search processing	
	Index type: hash format	Index type: tree format
'key-value<='search	An error is returned.	The record having a key value equal to or greater than the specified key value is searched for.
'key-value<'search	An error is returned.	The record having a key value greater than the specified key value is searched for.
'key-value>='search	An error is returned.	The record having a key value equal to or smaller than the specified key value is searched for.
'key-value>'search	An error is returned.	The record having a key value smaller than the specified key value is searched for.
First record search#	The first record that was hashed in correspondence with the key value is searched for. The key value specified for *data-name-G* is ignored.	An error is returned.
NEXT search#	The next record that was hashed in correspondence with the key value is searched for.	An error is returned.

#: All the records in the TAM table can be searched for by using the first record search and NEXT search in the following conditions:

The hash format is specified as the index type.

When a TAM table file is created, a key value is assigned to the data part (the `-s` option not specified in the `tamcre` command).

If lock is specified with input for reference processing, lock in tables and lock in records are enabled with lock for reference processing. If a TAM table open under lock in records is input for update processing, lock in tables is enabled with lock for reference processing, and lock in records is enabled with lock for update processing.

However, if table nonlock mode is specified as the access-time table lock mode in the TAM service definition, tables whose access type is reference or update without addition or deletion, their lock cannot be enabled.

If the program that inputs TAM table record returns with an error, all the resources specified in this program are released, and the status before this program was called is regained. However, if a record which was acquired under lock for reference processing before this program was called is input for update processing, lock for update processing is enabled. (Lock for reference processing is not regained.) If an error is returned, the buffer contents cannot be ensured.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify the TAM ID. The TAM service does not reference this value.

- *data-name-C*

  Specify the name of the TAM table of the record to be input. The name can be specified with up to 32 characters. If the specified name comprises less than 32 characters, pad the remaining portion with space.

- *data-name-D*

  Specify a buffer length if the input record is 32767 bytes or less. The buffer length must be equal to or greater than the record length. The setting here is valid only when `VALUE 'F`*xxx*`'` is specified for *data-name-E*.

- *data-name-J*

  Specify a buffer length if the input record is 32768 bytes or more. The buffer length must be equal to or greater than the record length. The setting here is valid only when `VALUE 'V`*xxx*`'` is specified for *data-name-E*.

- *data-name-E*

  Specify one of the following request codes:

  `VALUE 'FCHR', VALUE 'VCHR'`: Search for *key-value*= for reference (hash/tree format).

  `VALUE 'FGER', VALUE 'VGER'`: Search for *key-value*<= for reference (tree format).

  `VALUE 'FGTR', VALUE 'VGTR'`: Search for *key-value*< for reference (tree format).

  `VALUE 'FLER', VALUE 'VLER'`: Search for *key-value*>= for reference (tree format).

  `VALUE 'FLTR', VALUE 'VLTR'`: Search for *key-value*> for reference (tree format).

  `VALUE 'FTPR', VALUE 'VTPR'`: Search for the specified key value from the first record for reference (hash format).

  `VALUE 'FNXR', VALUE 'VNXR'`: Search for the specified key value from the next record for reference (hash/tree format).

  `VALUE 'FCHU', VALUE 'VCHU'`: Search for *key-value*= for updating (hash/tree format).

  `VALUE 'FGEU', VALUE 'VGEU'`: Search for *key-value*<= for updating (tree format).

  `VALUE 'FGTU', VALUE 'VGTU'`: Search for *key-value*< for updating (tree format).

  `VALUE 'FLEU', VALUE 'VLEU'`: Search for *key-value*>= for updating (tree format).

  `VALUE 'FLTU', VALUE 'VLTU'`: Search for *key-value*> for updating (tree format).

`VALUE 'FTPU', VALUE 'VTPU'`: Search for the specified key value from the first record for updating (hash format).

`VALUE 'FNXU', VALUE 'VNXU'`: Search for the specified key value from the next record for updating (hash/tree format).

- *data-name-F*

  If a record is input for reference processing, specify a lock enabled/disabled type.

  `VALUE '∆'`: Lock is enabled. (Lock is reset at the synchronization point.)

  `VALUE 'D'`: Lock is enabled. (Lock is reset when this CALL statement terminates.)

  `VALUE 'N'`: Lock is disabled.

- *data-name-G*

  Specify the key value with the length of the key area of the record to be searched for.

- *data-name-H*

  Specify the data area (buffer) to which the record is input.

- *data-name-I*

  Specify whether to wait for release from lock with one of the following values. The value specified for *data-name-I* is valid when 2 is specified for `tam_cbl_level` in the TAM service definition.

  `VALUE 'W'`: Wait for release from lock

  `VALUE 'N'`: Error return without waiting for release from lock

  If 0 or 1 is specified for `tam_cbl_level` in the TAM service definition, specifying a value for *data-name-I* is unnecessary.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-H*

  The input record is returned.

Status codes

Status code	Explanation
`00000`	The TAM table record was input normally.
`01701`	The TAM table name specified for *data-name-C* is invalid.
`01702`	The key value specified for *data-name-G* is invalid.

Status code	Explanation
01704	The value specified for *data-name-H* is invalid.
01705	The buffer length specified for *data-name-D* or *J* is too short.
01708	The value specified for *data-name-E*, *F* or *I* is invalid.
01709	The table specified for *data-name-C* is not a TAM table.
01710	The TAM table has not been defined.
01720	The TAM service is being terminated.
01721	The sequence of accessing the TAM table is invalid.
	The resource manager registration of the object file for control of transactions having a linkage with the UAP is invalid. Alternatively, there is no linkage between the object file for control of transactions and the UAP.
	`atomic_update=N` (nontransaction attribute) is specified in the user service definition of the UAP which called CBLDCTAM.
01723	The TAM table was deleted.
01724	The TAM table was not loaded.
01727	The TAM table is in logical descriptor state.
01728	The TAM table is in descriptor state due to an error.
01729	Execution is impossible with the index type of the TAM table specified for creation of a TAM table file.
01730	Execution is impossible in the access mode of the TAM table specified in the TAM service definition.
01731	A record satisfying the search conditions specified for *data-name-E* is not found.
01736	A lock error occurred. If you specified `0` in the `tam_cbl_level` operand of the TAM service definition, or if you specified `2` in the `tam_cbl_level` operand and set `W` in *data-name-I*, the resource could not be acquired because the wait time specified in the lock service definition reached timeout.
01737	A deadlock occurred.
01760	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current TAM table.
01761	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current OpenTP1 file service.
01762	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current TAM service.

Status code	Explanation
01764	The record has been damaged.
01765	The number of transactions exceeds the maximum number of transactions which can be managed by the TAM service.
01766	The number of open character special files exceeds the specified limit.
01767	The access permission for special files has not been granted.
01768	The access permission for TAM files has not been granted.
01769	The memory became insufficient.
01770	An input/output error occurred.
01771	A transaction service error occurred.
01772	The opening TAM file is protected by the security facility. No ACL exists for the file.
01773	The accessing TAM file is protected by the security facility. The UAP attempting to input a record from the TAM table has no access permission.

CBLDCTAM('GST ') - Acquire TAM table status

Format

■ PROCEDURE DIVISION specification

```
CALL  'CBLDCTAM'  USING   unique-name-1  unique-name-2
                          unique-name-3
```

■ DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(4).
    02  data-name-B     PIC X(5).
    02  FILLER          PIC X(35).
    02  data-name-C     PIC X(4).
    02  FILLER          PIC X(64).
    02  data-name-D     PIC S9(4) COMP.
    02  FILLER          PIC X(398).
01  unique-name-2.
    02  data-name-E     PIC X(4).
    02  FILLER          PIC X(4).
01  unique-name-3.
    02  unique-name-4.
        03  data-name-F     PIC X(32).
        03  data-name-G     PIC X(2).
        03  FILLER          PIC X(6).
            :
            :
    02  unique-name-n.
        03  data-name-F     PIC X(32).
        03  data-name-G     PIC X(2).
        03  FILLER          PIC X(6).
```

Description

`CBLDCTAM('GST ')` acquires the status of a TAM table. The TAM table status to be acquired includes the following:

- Open state
- Closed state
- Logical shutdown state
- Shutdown state due to an error

The program that acquires TAM table information can be called both outside and inside the transaction.

Even if the TAM table is accessed by another processing, the program that acquires

TAM table information returns assuming that the TAM table is open.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify the TAM ID. The TAM service does not reference this value.

- *data-name-D*

  Specify the length of the TAM table information area (length of unique-name-3).

- *data-name-E*

  Specify `VALUE 'GST '` for the request code indicating acquisition of TAM table information.

- *data-name-F*

  Specify the name of the TAM table whose status is to be acquired. The name can be specified with up to 32 characters. If the specified name comprises less than 32 characters, pad the remaining portion with space.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-C*

  The status of the specified TAM table is returned to the 4-byte area. Table 2-2 explains the status indicated by each byte.

Table 2-3: TAM table status

Byte	Status	Explanation
0	`L`	There is a TAM table in logical shutdown state.
	Space	There is not a TAM table in logical shutdown state.
1	`B`	There is a TAM table in shutdown state due to an error.
	Space	There is not a TAM table in shutdown state due to an error.
2	`O`	There is a TAM table in open state.
	Space	There is not a TAM table in open state.
3	`C`	There is a TAM table in closed state.
	Space	There is not a TAM table in closed state.

- *data-name-G*

  The status of the specified TAM table is returned.

  `VALUE 'RO'`: Open state

  `VALUE 'RC'`: Closed state

  `VALUE 'HL'`: Logical shutdown state

  `VALUE 'HB'`: Shutdown state due to an error

Status codes

Status code	Explanation
`00000`	TAM table information was acquired normally.
`01701`	The TAM table name specified for *data-name-F* is invalid.
`01705`	The value specified for *data-name-G* is invalid.
`01708`	The value specified for *data-name-E* is invalid.
`01710`	The TAM table has not been defined.
`01720`	The TAM service is being terminated.
`01721`	The sequence of accessing the TAM table is invalid.
	The resource manager registration of the object file for control of transactions having a linkage with the UAP is invalid. Alternatively, there is no linkage between the object file for control of transactions and the UAP.
	`atomic_update=N` (nontransaction attribute) is specified in the user service definition of the UAP which called CBLDCTAM.
`01762`	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current TAM service.
`01772`	The accessing TAM file is protected by the security facility. No ACL exists for the file.
`01773`	The accessing TAM file is protected by the security facility. The UAP attempting to get the status of the TAM table has no access permission.

CBLDCTAM('INFO') - Acquire TAM table information

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCTAM'  USING  unique-name-1  unique-name-2
                         unique-name-3
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(4).
    02  data-name-B     PIC X(5).
    02  FILLER          PIC X(103).
    02  data-name-C     PIC S9(4) COMP.
    02  FILLER          PIC X(398).
01  unique-name-2.
    02  data-name-D     PIC X(4).
    02  FILLER          PIC X(4).
01  unique-name-3.
    02  unique-name-4.
        03  data-name-E     PIC X(32).
        03  data-name-F     PIC X(2).
        03  data-name-G     PIC X(64).
        03  FILLER          PIC X(2).
        03  data-name-H     PIC S9(9) COMP.
        03  data-name-I     PIC S9(9) COMP.
        03  data-name-J     PIC X(1).
        03  data-name-K     PIC X(1).
        03  data-name-L     PIC X(1).

        03  FILLER          PIC X(1).
        03  data-name-M     PIC S9(9) COMP.
        03  data-name-N     PIC S9(9) COMP.
        03  data-name-O     PIC S9(9) COMP.
        03  data-name-P     PIC X(1).
        03  FILLER          PIC X(3).
                :
                :
                :
    02  unique-name-n.
        03  data-name-E     PIC X(32).
        03  data-name-F     PIC X(2).
                :
        03  data-name-P     PIC X(1).
        03  FILLER          PIC X(3).
```

Description

`CBLDCTAM('INFO')` returns TAM table information. The following values are returned by `CBLDCTAM('INFO')`:

- TAM file name
- TAM table status
- Number of records in use
- Maximum number of records
- Index type
- Access type
- Loading opportunity
- TAM record length
- Key length
- Key start position
- Security attribute

TAM table information can be acquired from inside or outside of a transaction.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify the TAM identifier. The TAM service does not reference this value.

- *data-name-C*

  Specify the length of area that receives TAM table information (the length of unique-name-3).

- *data-name-D*

  Specify `VALUE 'INFO'` for the request code indicating that TAM table information is acquired.

- *data-name-E*

  Specify the name of the TAM table from which information is acquired with up to 32 characters. End the character string with space.

Data areas whose values are returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

- *data-name-F*

  The TAM table status is returned as follows:

`VALUE 'RO'`: The TAM table is opened.

`VALUE 'RC'`: The TAM table is closed.

`VALUE 'HL'`: The TAM table is in logical shutdown state.

`VALUE 'HB'`: The TAM table is in shutdown state due to an error.

- *data-name-G*

  The TAM file name of the TAM table is returned.

- *data-name-H*

  The number of records currently used in the TAM table is returned. However, this value is not assured if a record is added or deleted after `CBLDCTAM('INFO')` is called.

- *data-name-I*

  The maximum number of records for the TAM table is returned.

- *data-name-J*

  The index type of the TAM table is returned as follows:

  `VALUE 'H'`: The TAM table adopts hash format.

  `VALUE 'T'`: The TAM table adopts tree format.

- *data-name-K*

  The access type of the TAM table is returned as follows:

  `VALUE 'R'`: The TAM table is reference-only type.

  `VALUE 'W'`: The TAM table is overwrite type (any record cannot be added or deleted).

  `VALUE 'A'`: The TAM table is update type (records can be added or deleted).

  `VALUE 'L'`: The TAM table is update type (records can be added and deleted without locking the table).

- *data-name-L*

  The loading opportunity of the TAM table is returned as follows:

  `VALUE 'S'`: The TAM table is loaded when the TAM service is started.

  `VALUE 'L'`: The TAM table is loaded when the TAM table is opened by the function `dc_tam_open()`.

  `VALUE 'C'`: The TAM table is loaded when the `tamload` command is executed.

- *data-name-M*

  The record length of the TAM table is returned.

- *data-name-N*

  The key length of the TAM table is returned.

- *data-name-O*

  The key start position in the TAM table data is returned.

- *data-name-P*

  The security attribute of the TAM table specified in the TAM service definition is returned as follows:

  `VALUE 'N'`: Security is not specified.

  `VALUE 'S'`: Security is specified.

Status codes

Status Code	Explanation
`00000`	Information was acquired from the TAM table normally.
`01701`	The value specified for *data-name-E* is invalid.
`01705`	The value specified for *data-name-C* is invalid.
`01708`	The value specified for *data-name-D* (request code) is invalid.
`01709`	The name specified for *data-name-E* is not a TAM table.
`01710`	The TAM table has not been defined.
`01720`	The TAM service is being terminated.
`01721`	The sequence of accessing the TAM table is invalid.
	The resource manager registration of the object file for control of transactions having a linkage with the UAP is invalid. Alternatively, there is no linkage between the object file for control of transactions and the UAP.
	`atomic_update=N` (nontransaction attribute) is specified in the user service definition of the UAP which called *CBLDCTAM*.
`01760`	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current TAM table.
`01762`	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current TAM service.
`01766`	The number of open character special files exceeds the specified limit.
`01767`	The access permission for special files has not been granted.
`01769`	The memory became insufficient.

Status Code	Explanation
`01770`	An input/output error occurred.
`01772`	The TAM table from which information is acquired is protected with the security facility. There is no ACL for the corresponding TAM table.
`01773`	The TAM table from which information is acquired is protected with the security facility. The UAP that called `CBLDCTAM('INFO')` has no access permission.

CBLDCTAM('MFY '/'MFYS'/'STR '/'WFY '/'WFYS'/'YTR ') - Update/add a TAM table record

Format

- PROCEDURE DIVISION specification

```
CALL 'CBLDCTAM' USING   unique-name-1  unique-name-2
                        unique-name-3  unique-name-4
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A     PIC X(4).
    02  data-name-B     PIC X(5).
    02  FILLER          PIC X(3).
    02  data-name-C     PIC X(32).
    02  FILLER          PIC X(68).
    02  data-name-D     PIC S9(4) COMP.
    02  FILLER          PIC X(2).
    02  data-name-J     PIC S9(9) COMP.
    02  FILLER          PIC X(392).
01  unique-name-2.
    02  data-name-E     PIC X(4).
    02  FILLER          PIC X(3).
    02  data-name-I     PIC X(1).
01  unique-name-3.
    02  data-name-F     PIC X(m).
01  unique-name-4.
    02  data-name-G     PIC X(n).
```

Description

`CBLDCTAM('MFY '/'MFYS'/'STR '/'WFY '/'WFYS'/'YTR ')` updates/adds a record indicated with a key value in/to a TAM table.

If a TAM table is open under lock in records, lock in records is enabled with lock for update processing as explained below.

- When the access type is "update" (`VALUE 'MFY '` or `VALUE 'WFY '` specified for *data-name-E*):

  Lock in tables is enabled with lock for reference processing, and lock in records is enabled with lock for update processing.

However, if table nonlock mode is specified as the access-time table lock mode in the TAM service definition, tables whose access type is reference or update without addition or deletion, their lock cannot be enabled.

- When the access type is "update or addition" or "addition" (`VALUE 'MFYS'`, `VALUE 'WFYS'`, `VALUE 'STR '` or `VALUE 'YTR '` specified for *data-name-E*):

  Lock in tables is enabled with lock for update processing.

If the program that updates/adds TAM table record returns with an error, all the resources specified in this program are released, and the status before this program was called is regained. However, if a TAM table which was acquired under lock for reference processing before this program was called is updated/added, lock for update processing is enabled. (Lock for reference processing is not regained.)

The key value storage location in the data to be updated/added and the key area length are as specified in the `tamcre` command used for creation of a TAM table file.

The data part has a key value if the key value is assigned to the data part (the `-s` option not specified in the `tamcre` command) when a TAM table file is created. Therefore, an error is returned if the key value specified in the program that updates/adds TAM table record is not found in the data to be updated/added. The data part has no key value if no key value is assigned to the data part (the `-s` option specified in the `tamcre` command). In this case, no check is made on the contents of the data to be updated/added.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify the TAM ID. The TAM service does not reference this value.

- *data-name-C*

  Specify the name of the TAM table whose record is to be updated/added. The name can be specified with up to 32 characters. If the specified name comprises less than 32 characters, pad the remaining portion with space.

- *data-name-D*

  Specify a data length in the range from 1 to 32767 bytes for the data to be updated or added. The data length must be equal to or greater than the record length. The setting here is valid only when `VALUE 'MFY '`, `VALUE 'MFYS'` or `VALUE 'STR '` is specified for *data-name-E*.

- *data-name-J*

  Specify a data length of 32768 bytes or more for the data to be updated or added. The data length must be equal to or greater than the record length. The setting here is valid only when `VALUE 'WFY '`, `VALUE'WFYS'` or `VALUE 'YTR '` is specified for *data-name-E*.

- *data-name-E*

  Specify one of the following request codes:

`VALUE 'MFY '`, `VALUE 'WFY '`: Record updating only

`VALUE 'MFYS'`, `VALUE 'WFYS'`: Record updating or addition

`VALUE 'STR '`, `VALUE 'YTR '`: Record addition only

- *data-name-F*

  Specify the key value with the length of the key area of the record to be updated/added.

- *data-name-G*

  Specify the data area (buffer) to/in which the record is updated/added.

- *data-name-I*

  Specify whether to wait for release from lock with one of the following values. The value specified for *data-name-I* is valid when 2 is specified for `tam_cbl_level` in the TAM service definition.

  `VALUE 'W'`: Wait for release from lock

  `VALUE 'N'`: Error return without waiting for release from lock

  If 0 or 1 is specified for `tam_cbl_level` in the TAM service definition, specifying a value for *data-name-I* is unnecessary.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	The TAM table record was updated/added normally.
`01701`	The TAM table name specified for *data-name-C* is invalid.
`01702`	The key value specified for *data-name-F* is invalid.
`01706`	The value specified for *data-name-G* is invalid.
`01707`	The value specified for *data-name-D* or *J* is invalid.
`01708`	The value specified for *data-name-E* or *I* is invalid.
`01709`	The table specified for *data-name-C* is not a TAM table.
`01710`	The TAM table has not been defined.
`01720`	The TAM service is being terminated.

Status code	Explanation
`01721`	The sequence of accessing the TAM table is invalid.
	The resource manager registration of the object file for control of transactions having a linkage with the UAP is invalid. Alternatively, there is no linkage between the object file for control of transactions and the UAP.
	`atomic_update=N` (nontransaction attribute) is specified in the user service definition of the UAP which called *CBLDCTAM*.
`01723`	The TAM table was deleted.
`01724`	The TAM table was not loaded.
`01727`	The TAM table is in logical shutdown state.
`01728`	The TAM table is in shutdown state due to an error.
`01730`	Execution is impossible in the access mode of the TAM table specified in the TAM service definition.
`01731`	The specified record does not exist.
`01735`	The record cannot be added because the key value specified for *data-name-F* exists in the TAM table.
`01736`	A lock error occurred. If you specified `0` in the `tam_cbl_level` operand of the TAM service definition, or if you specified `2` in the `tam_cbl_level` operand and set `W` in *data-name-I*, the resource could not be acquired because the wait time specified in the lock service definition reached timeout.
`01737`	A deadlock occurred.
`01760`	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current TAM table.
`01761`	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current OpenTP1 file service.
`01762`	The version of the TAM library linked to the UAP does not allow the UAP to operate with the current TAM service.
`01763`	The TAM table has no free record.
`01764`	The record has been damaged.
`01765`	The number of transactions exceeds the maximum number of transactions which can be managed by the TAM service.
`01766`	The number of open character special files exceeds the specified limit.
`01767`	The access permission for special files has not been granted.

Status code	Explanation
01768	The access permission for TAM files has not been granted.
01769	The memory became insufficient.
01770	An input/output error occurred.
01771	A transaction service error occurred.
01772	The opening TAM file is protected by the security facility. No ACL exists for the file.
01773	The accessing TAM file is protected by the security facility. The UAP attempting to update or add a record to the TAM table has no access permission.

Transaction control (CBLDCTRN)

This section gives the syntax and other information of the following COBOL-UAP creation programs which are used as OpenTP1-specific transaction control:

- `CBLDCTRN('BEGIN    ')` - Start a transaction
- `CBLDCTRN('C-COMMIT ')` - Enable commitment in chained mode
- `CBLDCTRN('C-ROLL   ')` - Enable rollback in chained mode
- `CBLDCTRN('INFO     ')` - Report the information about the current transaction
- `CBLDCTRN('U-COMMIT ')` - Enable commitment in unchained mode
- `CBLDCTRN('U-ROLL   ')` - Enable rollback in unchained mode

The COBOL-UAP creation programs for transaction control (`CBLDCTRN`) can be used in UAPs of both TP1/Server Base and TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language template for transaction control (`CBLDCTRN`) is stored in `DCTRN.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCTRN('BEGIN   ') - Start a transaction

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCTRN'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'BEGIN   '.
    02  data-name-B    PIC X(5).
```

Description

`CBLDCTRN('BEGIN   ')` starts a global transaction from the process that calls the program. `CBLDCTRN('BEGIN   ')` that started the transaction becomes the root transaction branch of the global transaction.

For the UAP that starts the transaction, specify transaction attribute. Once the transaction is started in a global transaction, the transaction cannot be restarted from any transaction branch of the global transaction. If the transaction is started more than once in a global transaction, an error is returned.

Data area whose value is set in the UAP

- *data-name-A*

Specify `VALUE 'BEGIN△△△'` for the request code indicating transaction start. The contents remain unchanged for processing after the transaction starts.

Data area whose value is returned from OpenTP1

- *data-name-B*

A status code of 5 digits is returned.

Status codes

Status code	Explanation
00000	Normal termination. A global transaction was generated, and the process that called `CBLDCTRN('BEGIN   ')` is in the range of the global transaction.
00905	A service was requested to the service program from an invalid context (e.g., already in the transaction). Alternatively, the transaction could not be started because the execution environment was in non-journal operation mode.
00906	A resource manager (RM) error occurred. A transaction could not be generated.

Status code	Explanation
00907	A transaction could not be generated because a transaction service error occurred. The value assigned to the `trn_tran_process_count` operand in the transaction service definition may be too small. If this code is returned, re-execute processing. The re-execution is very likely to be successful.
00908	The request code is invalid.

Example

```
01 MISC.
    02 CMD-CODE     PIC X(8).
    02 STATUS-CODE  PIC X(5).
      :
      :
MOVE 'BEGIN' TO CMD-CODE OF MISC.
CALL 'CBLDCTRN' USING MISC.
IF STATUS-CODE OF MISC NOT EQUAL TO '00000' THEN
    MOVE 'CANNOT BEGIN TRANSACTION' TO ERRMSG OF
                                       OUT-ERROR-REC
    WRITE OUT-ERROR-REC
END IF.
```

CBLDCTRN('C-COMMIT') - Enable commitment in chained mode

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCTRN'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'C-COMMIT'.
    02  data-name-B    PIC X(5).
```

Description

`CBLDCTRN('C-COMMIT')` acquires the synchronization point of a transaction. The normal termination of processing (commitment) is reported from the root transaction branch to the UAPs, transaction services, and resource managers of transaction branches which form the transaction.

When `CBLDCTRN('C-COMMIT')` terminates normally, a new global transaction is started. The process that called `CBLDCTRN('C-COMMIT')` is in the range of this transaction. However, this does not mean the specification of a transaction mode for a UAP other than the UAP that called `CBLDCTRN('C-COMMIT')`.

When a global transaction consists of multiple transaction branches (not only with the UAP that called `CBLDCTRN('C-COMMIT')`), commitment processing is executed only when the processing results of each transaction branch are committed.

`CBLDCTRN('C-COMMIT')` can be called only from the root transaction branch (the UAP that started the transaction. If `CBLDCTRN('C-COMMIT')` is called from another UAP, the status code `00905` is returned.

Only the process that started the UAP created correctly according to the specification in this manual is permitted to call `CBLDCTRN('C-COMMIT')`.

`CBLDCTRN('C-COMMIT')` returns with normal or abnormal termination when synchronization point processing is completed. To terminate the service that called `CBLDCTRN('C-COMMIT')` normally, the transaction attribute must be set in the UAP execution environment.

Data area whose value is set in the UAP

- *data-name-A*

Specify `VALUE 'C-COMMIT'` for the request code indicating transaction commit in chained mode. The contents remain unchanged for processing after the chained mode commit is executed.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination. Even after `CBLDCTRN('C-COMMIT')` terminates, this process is under the transaction and it is in the range of the global transaction.
`00902`	The current transaction was rolled back because it could not be committed. After the completion of processing, this process is under the transaction and it is in the range of the global transaction.
`00903`	The global transaction that called `CBLDCTRN('C-COMMIT')` was determined heuristically. Consequently, a transaction branch was committed, and another transaction branch was rolled back. This code is returned if the results of heuristic decision do not match the results of the synchronization point of the global transaction. Refer to the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. Even after this code is returned, this process is under the transaction and it is in the range of the global transaction.
`00904`	A transaction branch of the global transaction was completed heuristically. However, the results of the synchronization point of the heuristically completed transaction branch are not known due to an error. Refer to the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. Even after this code is returned, this process is under the transaction and it is in the range of the global transaction. This function returns the status code `00904` even when you specify `00000001` for the `trn_extend_function` operand in the transaction service definition and the return value from the resource manager at one-phase commit is `XAER_NOTA`.
`00905`	`CBLDCTRN('C-COMMIT')` was called from an invalid context (e.g., already not in the transaction). The transaction mode is not affected.
`00908`	The request code is invalid.
`00924`	The commitment processing terminated normally, but the new transaction could not be started. After this code is returned, this process is no longer under the transaction.
`00925`	The transaction to be committed was rolled back because it could not be committed. The new transaction could not be started. After this code is returned, this process is no longer under transaction.

Status code	Explanation
`00926`	The global transaction that called `CBLDCTRN('C-COMMIT')` was determined heuristically. Consequently, a transaction branch was committed, and another transaction branch was rolled back. This error is returned if the result of heuristic decision do not match the results of the synchronization point of the global transaction. Refer to the contents of the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. The new transaction could not be started. After this code is returned, this process is no longer under transaction.
`00927`	A transaction branch of the global transaction was completed heuristically. However, the results of the synchronization point of the heuristically completed transaction branch are not known, due to an error. Refer to the contents of the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. The new transaction could not be started. After this code is returned, this process is no longer under transaction. This function returns the status code `00927` even when you specify `00000001` for the `trn_extend_function` operand in the transaction service definition and the return value from the resource manager is `XAER_NOTA`.

Example

```
01 MISC.
    02  CMD-CODE      PIC X(8).
    02  STATUS-CODE   PIC X(5).
      :
      :
MOVE 'C-COMMIT' TO CMD-CODE OF MISC.
CALL 'CBLDCTRN' USING MISC.
IF STATUS-CODE OF MISC NOT EQUAL TO '00000' THEN
    MOVE 'CANNOT COMMIT TRANSACTION' TO ERRMSG OF
                                           OUT-ERROR-REC
    WRITE OUT-ERROR-REC
END IF.
```

CBLDCTRN('C-ROLL ') - Enable rollback in chained mode

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCTRN'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'C-ROLL  '.
    02  data-name-B    PIC X(5).
```

Description

`CBLDCTRN('C-ROLL  ')` rolls back a transaction. A transaction is generated immediately after `CBLDCTRN('C-ROLL  ')` is called.

By calling `CBLDCTRN('C-ROLL  ')`, rollback processing is reported from the root transaction branch of the global transaction to the UAPs, transaction services, and resource managers of transaction branches which form the transaction.

When `CBLDCTRN('C-ROLL  ')` terminates normally, the process that called the program returns after rollback processing. Then, a new global transaction is started. The process that calls `CBLDCTRN('C-ROLL  ')` is in the range of this transaction. However, this does not mean the specification of a transaction mode for a UAP other than the UAP that called this program.

`CBLDCTRN('C-ROLL  ')` can be called only from the root transaction branch (the UAP that called `CBLDCTRN('BEGIN   ')`) of a global transaction. If `CBLDCTRN('C-ROLL  ')` is called from another UAP, the status code `00905` is returned.

Only the process that started the UAP executable file correctly linked according to the specification in this manual is permitted to call `CBLDCTRN('C-ROLL  ')`. To terminate `CBLDCTRN('C-ROLL  ')`, specify the transaction attribute at UAP execution environment setup.

Data area whose value is set in the UAP

- *data-name-A*

Specify `VALUE 'C-ROLL△△'` for the request code indicating rollback in chained mode. The contents remain unchanged for processing after the chained mode rollback is executed.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status Code	Explanation
`00000`	Normal termination. Even after `CBLDCTRN('C-ROLL    ')` terminates, this process is under the transaction and it is in the range of the global transaction.
`00903`	The global transaction that called `CBLDCTRN('C-ROLL    ')` was determined heuristically. Consequently, a transaction branch was committed, and another transaction branch was rolled back. This code is returned if the results of heuristic decision do not match the results of the synchronization point of the global transaction. Refer to the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. Even after this code is returned, this process is under the transaction and it is in the range of the global transaction.
`00904`	A transaction branch of the global transaction was completed heuristically. However, the results of the synchronization point of the heuristically completed transaction branch are not known due to an error. Refer to the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. Even after this code is returned, this process is under the transaction and it is in the range of the global transaction.
`00905`	`CBLDCTRN('C-ROLL    ')` was called from an invalid context (e.g., already not in the transaction). The transaction mode is not affected.
`00908`	The request code is invalid.
`00924`	The rollback processing terminated normally, but the new transaction could not be started. After this code is returned, this process is no longer under the transaction.
`00926`	The global transaction that called `CBLDCTRN('C-ROLL    ')` was determined heuristically. Consequently, a transaction branch was committed, and another transaction branch was rolled back. This error is returned if the result of heuristic decision do not match the results of the synchronization point of the global transaction. Refer to the contents of the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. The new transaction could not be started. After this code is returned, this process is no longer under transaction.

Status Code	Explanation
00927	A transaction branch of the global transaction was completed heuristically. However, the results of the synchronization point of the heuristically completed transaction branch are not known, due to an error. Refer to the contents of the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. The new transaction could not be started. After this code is returned, this process is no longer under transaction.

Example

```
01 MISC.
    02 CMD-CODE     PIC X(8).
    02 STATUS-CODE  PIC X(5).
      :
      :
MOVE 'C-ROLL  ' TO CMD-CODE OF MISC.
CALL 'CBLDCTRN' USING MISC.
IF STATUS-CODE OF MISC NOT EQUAL TO '00000' THEN
    MOVE 'CANNOT ROLLBACK TRANSACTION' TO ERRMSG OF
                                            OUT-ERROR-REC
    WRITE OUT-ERROR-REC
END IF.
```

CBLDCTRN('INFO   ') - Report the information about the current transaction

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCTRN'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A      PIC X(8) VALUE 'INFO    '.
    02  data-name-B      PIC X(5).
    02  FILLER           PIC X.
    02  data-name-C.
        03  data-name-D  PIC S9(4) COMP VALUE ZERO.
```

Description

`CBLDCTRN('INFO    ')` reports whether the UAP that called `CBLDCTRN('INFO    ')` is operating as the current transaction.

Only the process that started the UAP created correctly according to the specification in this manual is permitted to call `CBLDCTRN('INFO    ')`. To terminate the service that called `CBLDCTRN('INFO    ')` normally, the transaction attribute must be set in the UAP execution environment.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'INFO`ΔΔΔΔ`'` for the request code indicating that information about the current transaction is reported. The contents remain unchanged for processing after `CBLDCTRN('INFO    ')` is executed.

- *data-name-C*

  This area stores information about the current transaction. Specify `0`.

- *data-name-D*

  Specify `0`.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00001`	The process that called `CBLDCTRN('INFO    ')` is operating as a transaction.
`00000`	The process that called `CBLDCTRN('INFO    ')` is not operating as a transaction.
`00908`	The request code is invalid.

Example

```
01 MISC.
    02  CMD-CODE      PIC X(8).
    02  STATUS-CODE   PIC X(5).
    02  FILLER        PIC X.
    02  TRAN-INFO.
        03 LEN        PIC S9(4) COMP.
           :
           :
MOVE ZERO TO LEN OF TRAN-INFO OF MISC.
MOVE 'INFO' TO CMD-CODE OF MISC.
CALL 'CBLDCTRN' USING MISC.
IF STATUS-CODE OF MISC NOT EQUAL TO '00001' THEN
    MOVE 'NOW IN TRANSACTION' TO ERRMSG OF OUT-ERROR-REC
    WRITE OUT-ERROR-REC
END IF.
```

Note

This API does not obtain a UAP trace.

CBLDCTRN('U-COMMIT') - Enable commitment in unchained mode

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCTRN'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'U-COMMIT'.
    02  data-name-B    PIC X(5).
```

Description

`CBLDCTRN('U-COMMIT')` reports the normal termination of a global transaction (commitment) to the UAPs, transaction services, and resource managers of transaction branches which form the transaction. After `CBLDCTRN('U-COMMIT')` terminates normally, a new global transaction is not generated.

When a global transaction consists of multiple transaction branches (not only with the UAP that called `CBLDCTRN('U-COMMIT')`), commitment processing is executed only when the processing results of each transaction branch is committed.

The `CBLDCTRN('U-COMMIT')` can be called only from the root transaction branch (the UAP that called `CBLDCTRN('U-COMMIT')`) of a global transaction. If `CBLDCTRN('U-COMMIT')` is called from another UAP, the status code `00905` is returned.

Only the process that started the UAP executable file correctly linked according to the specification in this manual is permitted to call `CBLDCTRN('U-COMMIT')`.

`CBLDCTRN('U-COMMIT')` returns with normal or abnormal termination when synchronization point processing is completed. To terminate `CBLDCTRN('U-COMMIT')` normally, specify the transaction attribute at UAP execution environment setup.

Data area whose value is set in the UAP

- *data-name-A*

Specify `VALUE 'U-COMMIT'` for the request code indicating transaction commit in unchained mode. The contents remain unchanged for processing after the unchained mode commit statement is executed.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination. This process is not under the transaction and it is not in the range of the global transaction.
`00902`	The current transaction was rolled back because it could not be committed. This process is not in the range of the global transaction.
`00903`	The global transaction that called `CBLDCTRN('U-COMMIT')` was determined heuristically. Consequently, a transaction branch was committed, and another transaction branch was rolled back. This code is returned if the results of heuristic decision do not match the results of the synchronization point of the global transaction. Refer to the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. After this code is returned, this process is not under the transaction and it is not in the range of the global transaction.
`00904`	A transaction branch of the global transaction was completed heuristically. However, the results of the synchronization point of the heuristically completed transaction branch are not known due to an error. Refer to the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. After this code is returned, this process is not under the transaction and it is not in the range of the global transaction. This function returns the status code `00904` even when you specify `00000001` for the `trn_extend_function` operand in the transaction service definition and the return value from the resource manager at one-phase commit is `XAER_NOTA`.
`00905`	`CBLDCTRN('U-COMMIT')` was called from an invalid context (e.g., already not in the transaction). The transaction mode is not affected.
`00908`	The request code is invalid.

Example

```
01 MISC.
    02  CMD-CODE      PIC X(8).
    02  STATUS-CODE   PIC X(5).
       :
       :
MOVE 'U-COMMIT' TO CMD-CODE OF MISC.
CALL 'CBLDCTRN' USING MISC.
IF STATUS-CODE OF MISC NOT EQUAL TO '00000' THEN
    MOVE 'CANNOT COMMIT TRANSACTION' TO ERRMSG OF
                                         OUT-ERROR-REC
    WRITE OUT-ERROR-REC
END IF.
```

CBLDCTRN('U-ROLL ') - Enable rollback in unchained mode

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCTRN'  USING  unique-name-1
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'U-ROLL  '.
    02  data-name-B    PIC X(5).
```

Description

CBLDCTRN('U-ROLL ') rolls back a transaction. After CBLDCTRN('U-ROLL ') terminates normally, a new global transaction is not generated.

CBLDCTRN('U-ROLL ') can be called from any transaction branch of a global transaction. If CBLDCTRN('U-ROLL ') is called from the root transaction branch, a new transaction is not started after CBLDCTRN('U-ROLL ') returns. If CBLDCTRN('U-ROLL ') is called from a transaction branch other than the root transaction branch, CBLDCTRN('U-ROLL ') puts the transaction branch into rollback_only state and returns to the client UAP. In this case, the transaction branch is in the range of the global transaction until the synchronization point processing of the root transaction branch is completed.

Only the process that started the UAP executable file correctly linked according to the specification in this manual is permitted to call CBLDCTRN('U-ROLL '). To terminate CBLDCTRN('U-ROLL ') normally, specify the transaction attribute at UAP execution environment setup.

Data area whose value is set in the UAP

- *data-name-A*

  Specify VALUE 'U-ROLL△△' for the request code indicating rollback in unchained mode. The contents remain unchanged for processing after the unchained mode rollback is executed.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination. If `CBLDCTRN('U-ROLL ')` is called from the root transaction branch, this process is not under the transaction and it is not in the range of the global transaction. If `CBLDCTRN('U-ROLL ')` is called from a transaction branch other than the root transaction branch, this process is put into `rollback_only` state.
`00903`	The global transaction that called `CBLDCTRN('U-ROLL ')` was determined heuristically. Consequently, a transaction branch was committed, and another transaction branch was rolled back. This code is returned if the results of heuristic decision do not match the results of the synchronization point of the global transaction. Refer to the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. After this code is returned, this process is not under the transaction and it is not in the range of the global transaction.
`00904`	A transaction branch of the global transaction was completed heuristically. However, the results of the synchronization point of the heuristically completed transaction branch are not known due to an error. Refer to the message log file for the results of the synchronization point of the UAP, resource manager, or global transaction that caused this code to be returned. After this code is returned, this process is not under the transaction and it is not in the range of the global transaction.
`00905`	`CBLDCTRN('U-ROLL ')` was called from an invalid context (e.g., already not in the transaction). The transaction mode is not affected.
`00908`	The request code is invalid.

Example

```
01 MISC.
    02 CMD-CODE     PIC X(8).
    02 STATUS-CODE  PIC X(5).
       :
       :
MOVE 'U-ROLL  ' TO CMD-CODE OF MISC.
CALL 'CBLDCTRN' USING MISC.
IF STATUS-CODE OF MISC NOT EQUAL TO '00000' THEN
    MOVE 'CANNOT ROLLBACK TRANSACTION' TO ERRMSG OF
                                         OUT-ERROR-REC
    WRITE OUT-ERROR-REC
END IF.
```

Online tester management (CBLDCUTO)

This section gives the syntax and other information of the following COBOL-UAP creation program which is used to manage the status of TP1/Online Tester from a user server when it is used by OpenTP1.

- `CBLDCUTO('T-STATUS')` - Report the test status of a user server

The COBOL-UAP creation program for online tester management (`CBLDCUTO`) can be used only in UAPs of TP1/Server Base. It cannot be used in UAPs of TP1/LiNK.

When defining `DATA DIVISION` of COBOL-UAP creation programs, the COBOL language templates can be used as samples. The COBOL language template for online tester management (`CBLDCUTO`) is stored in `DCUTO.cbl` under the `/BeTRAN/examples/COBOL/` directory.

CBLDCUTO('T-STATUS') - Report the test status of a user server

Format

- PROCEDURE DIVISION specification

```
CALL  'CBLDCUTO'  USING  unique-name-1  unique-name-2
```

- DATA DIVISION specification

```
01  unique-name-1.
    02  data-name-A    PIC X(8) VALUE 'T-STATUS'.
    02  data-name-B    PIC X(5).
    02  FILLER         PIC X(3).
    02  data-name-Z    PIC S9(9) COMP VALUE ZERO.
01  unique-name-2.
    02  data-name-C    PIC X(4).
    02  data-name-D    PIC X(1).
    02  data-name-E    PIC X(1).
    02  data-name-F    PIC X(1).
    02  data-name-G    PIC X(1).
    02  data-name-H    PIC X(1).
    02  FILLER         PIC X(22).
```

Description

`CBLDCUTO('T-STATUS')` reports the test status of the user server that called `CBLDCUTO('T-STATUS')`. When `CBLDCUTO('T-STATUS')` terminates normally, the test status is set to *data-name-D*, *data-name-E*, *data-name-F*, *data-name-G*, and *data-name-H*. If `CBLDCUTO('T-STATUS')` returns with an error, the test status information is not guaranteed.

Data areas whose values are set in the UAP

- *data-name-A*

  Specify `VALUE 'T-STATUS'` for the request code indicating the report of the test status of a user server.

- *data-name-Z*

  Specify `0`.

Data areas to which values are returned from OpenTP1

- *data-name-B*

  The status code of 5 digit is returned.

- *data-name-C*

  The test user ID (the value specified for the environment variable DCUTOKEY) is set here.

- *data-name-D*

  Whether the user server is operating in test mode or not is set here.

  `VALUE 'T'`: The user server is operating in test mode.

  `VALUE 'N'`: The user server is not operating in test mode.

- *data-name-E*

  The status of global transaction processing is set here.

  `VALUE 'C'`: The global transaction is committed in synchronization point processing.

  `VALUE 'R'`: The global transaction is rolled back in synchronization point processing.

  `VALUE 'N'`: Non-transaction status

  Space: The user server is not in test mode or an MHP linked to the MCF library is used.

- *data-name-F*

  The test type specified for `test_mode` operand in the user service definition is set here.

  `VALUE 'T'`: Test as UAP only for testing (`target`)

  `VALUE 'U'`: Test as usable UAP (`usable`)

  `VALUE 'S'`: Test as simulate MHP (`simmhp`)

  `VALUE 'N'`: MHP not intended for test (`no`)

- *data-name-G*

  The handling of the synchronization point of transaction specified for `test_transaction_commit` operand in the user service definition is set here.

  `VALUE 'C'`: The transaction is committed at synchronization point (Y).

  `VALUE 'R'`: The transaction is rolled back at synchronization point (N).

  Space: The user server is not in test mode or an MHP linked to the MCF library is used.

- *data-name-H*

  The handling of the results of command execution specified for `test_adm_call_command` operand in the user service definition is set here.

  `VALUE 'D'`: The command is executed. (`do`)

  `VALUE 'S'`: An assumed value is specified as the results of execution. (`skip`)

`VALUE 'F'`: Data from the operation command data results file is used. (`file`)

Space: The user server is not in test mode or an MHP linked to the MCF library is used.

Status codes

Status code	Explanation
`00000`	Normal termination. The test status is set to each data area.
`02701`	`CBLDCRPC('OPEN     ')` was not called.
`02734`	The version of the OpenTP1 library linked to the UAP does not allow the UAP to operate with the current transaction service.
`02757`	The value specified for *data-name-Z* is invalid.
`02759`	The request code (*data-name-A*) is invalid.

Note

When `CBLDCUTO('T-STATUS')` is called from the MHP linked to the MCF library, the following values are set to each data area:

- *data-name-C*: Test user ID
- *data-name-D*: Current operating service mode
- *data-name-E*: Space
- *data-name-F*: `'N'`
- *data-name-G*: Space
- *data-name-H*: Space

Chapter

3. Syntax of OpenTP1 Programs for COBOL-UAP Creation Programs (DML Interface)

This chapter explains the syntax of UAP creation programs (DML interface) which are used to create OpenTP1 UAPs in the COBOL language.

This chapter contains the following sections:

Coding in data manipulation language

When a UAP is created in COBOL language, OpenTP1 message exchange facilities can be created in Data Manipulation Language (DML). CALL statements and DML can coexist when one service is coded.

DML can be used only with the TP1/Server Base. It cannot be used with the TP1/LiNK.

The DML consists of a communication description entry for defining a work area and a communication statement used as an instruction.

Table 3-1 lists message exchange facilities provided in DML.

Table 3-1: DML provided by OpenTP1

Communication statement		Facility	Corresponding CALL interface
Data communication facility	RECEIVE[#1]	Receive a message.	CBLDCMCF('RECEIVE ')
		Receive a synchronous message.	CBLDCMCF('RECVSYNC')
	SEND[#1]	Send a message.	CBLDCMCF('SEND ')
		Send a response message.	CBLDCMCF('REPLY ')
		Send a synchronous message.	CBLDCMCF('SENDSYNC')
		Exchange a synchronous message.	CBLDCMCF('SENDRECV')
	ENABLE[#2]	Send a synchronous message.	CBLDCMCF('SENDSYNC')
	DISABLE[#2]	Send a synchronous message.	CBLDCMCF('SENDSYNC')
Service facility	DISABLE	Terminate continuous-inquiry-response processing.	CBLDCMCF('CONTEND ')
	RECEIVE	Accept temporary-stored data.	CBLDCMCF('TEMPGET ')
	ROLLBACK	Enable MHP rollback.	CBLDCMCF('ROLLBACK')
	SEND	Activate an application program.	CBLDCMCF('EXECAP ')
	SEND	Update temporary-stored data.	CBLDCMCF('TEMPPUT ')
	SEND	Execute an operation command.	CBLDCADM('COMMAND ')
	SEND	Acquire a user journal.	CBLDCJNL('UJPUT ')

Note

There is no DML interface for the message resend

`(CBLDCMCF('RESEND ' ) ).`

#1: For details on the syntax, see the applicable *OpenTP1 Protocol* manual.

#2: Can be used only when TP1/NET/OSI-TP is used. See the *OpenTP1 Protocol TP1/NET/OSI-TP* manual for details on the syntax.

The UAP trace information of a UAP created in DML is equivalent to the information of the facility library which is called by a UAP created with a CALL interface of the COBOL or C language.

General syntax rules

The DML of a UAP operating under OpenTP1 consists of a communication section, communication statements, and a part coded in the original COBOL syntax. This section explains the coding rules of the communication section and communication statements, and the syntax rules that must apply to coding of the original COBOL language. Items not explained in this section must comply with the syntax rules of the COBOL language. For details about the general syntax rules of the COBOL language, see an appropriate COBOL language guide.

■ Coding symbols

The following table explains the symbols used in the coding format shown in this section:

Coding symbol	Explanation
`[ ]`	Indicates that items enclosed in brackets `[ ]` are optional. Example: `[BEFORE ERASING]`
`{ }`	Indicates that items enclosed in braces `{ }` represent alternative items. Only one of the items can be specified.
_____	Indicates that a underlined reserved word is mandatory and cannot be omitted. A reserved word without double underlines is optional. (It is not always necessary to write the word.)

■ Unique names and data names

The unique names and data names in this manual have the same meanings as those used with COBOL85.

Example

A subscript and indicator are added to unique names, but not to data names. However, a subscript and indicator are not added to unique names in communication statements.

Formats

- Communication section coding rules

The communication section is written in the data division. The figure below shows the data division including the communication section. Each section of the data division must be written in the format shown below.

```
DATA DIVISION.
 [ FILE SECTION.
        :     ]
 [ WORKING-STORAGE SECTION.
        :     ]
 [ LINKAGE SECTION.
        :     ]
 [ COMMUNICATION SECTION.
  communication-description-entry
        :     ]
 [ REPORT SECTION.
        :     ]
```

Write the communication section beginning from the section header (a reserved word list called `COMMUNICATION SECTION` which is ended with a period and space). Write at least one communication description entry (CD) following the header.

- Communication description entry (CD) coding rules

A communication description entry defines the type of communication and the interface areas of UAP and OpenTP1. The following figure shows the general communication description entry format:

```
CD communication-description-name
FOR {INPUT|OUTPUT|I-O} [STORAGE|JOURNAL|PROGRAM|COMMAND]
    [ STATUS KEY IS data-name-1]
    [ SYMBOLIC TERMINAL IS data-name-2]
    [ MESSAGE DATE IS data-name-3]
    [ MESSAGE TIME IS data-name-4]
    [ MAP NAME IS data-name-5]
    [ SYNCHRONOUS MODE IS {SYNC|ASYNC|data-name-6}]
    [ SWITCHING MODE IS {NORMAL|PRIOR|data-name-7}]
    [ NEXT TRANSACTION IS data-name-8]
    [ ACTIVE INTERVAL IS data-name-9]
    [ DETAIL MODE IS data-name-10]
    [ WAITING TIME IS data-name-11]
```

- Write the communication description entry beginning from CD which is a level indication word, and end the level indication word CD with a period and space.
- Write the FOR clause following the level indication word CD. Clauses following the FOR clause can be written in any sequence.
- For the communication description entry name and data names 1 to 11, specify

names which can be used with the COBOL language to be used. Do not specify names which cannot be used with OpenTP1 UAPs.

- Write one communication description entry within 20 lines. The COBOL continuation line facility cannot be used. Also, a comment line or null line cannot be inserted into one communication description entry.
- End the communication description entry with a period and space.

■ Communication statement coding rules

A communication statement is used in the procedure division. The following figure shows the general communication statement format:

```
RECEIVE communication-description-name {[FIRST]SEGMENT|MESSAGE}
         [INTO  unique-name-1] [ BEFORE ERASING ].
SEND communication-description-name [FROM unique-name-1]
      WITH {ESI|EMI|unique-name-2}]
        [ BEFORE RECEIVING MESSAGE INTO unique-name-3].
DISABLE communication-description-name [ WITH unique-name-1].
ROLLBACK [WITH STOPPING]
```

- Write each communication statement clause in the sequence shown in the general format.
- The communication description name in a communication statement must be defined in the communication section.
- Only a unique name defined in the working storage section can be specified in a communication statement.
- A communication statement can be written extending over multiple lines. However, it is not permitted to write a hyphen (-) in the indicator area (continuation line coding method). Also, a comment line or a null line cannot be inserted into one communication statement.
- Do not write a communication statement in the line in which the original COBOL85 statement or a paragraph name is written.
- Only unique names at 01 or 77 level can be written in communication statements.

■ Communication description entry coding rules

Communication description entries used in a communication statement can be shared with multiple communication statements. The communication description entries include data names whose values are set in the UAP and data names whose values are returned from OpenTP1. The contents of data names other than data names whose values are returned from OpenTP1 are identical before and after the communication statement is issued. Therefore, there is no need to respecify a data name in the following case:

The same communication description entry is used in multiple lines, and the same contents as for the previously issued communication statement are used.

Table 3-2 shows clauses for specifying data names in a communication description entry, and their edit formats.

Table 3-2: Clauses for specifying data names in communication description entry and their edit formats

Clause for specifying data name	Data area format	Specification source of data area value#									
		1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
`STATUS KEY`	*data-name-1* `PIC X(5).`	B	B	B	B	B	B	B	B	B	B
`SYMBOLIC TERMINAL`	*data-name-2* `PIC X(8).`	B	--	U	U	U	U	U	U	--	--
`MESSAGE DATE`	*data-name-3* `PIC 9(6).`	B	--	--	--	--	--	--	--	--	--
`MESSAGE TIME`	*data-name-4* `PIC 9(8).`	B	--	--	--	--	--	--	--	--	--
`MAP NAME`	*data-name-5* `PIC X(8).`	b	--	U	U	--	--	--	--	--	--
`SYNCHRONOUS MODE`	*data-name-6* `PIC X(1).`	--	--	u	u	--	--	--	--	--	--
`SWITCHING MODE`	*data-name-7* `PIC X(1).`	--	--	u	u	--	--	--	--	--	--
`NEXT TRANSACTION`	*data-name-8* `PIC X(8).`	--	--	u	--	--	--	--	--	--	--
`ACTIVE INTERVAL`	*data-name-9* `PIC X(8).`	--	--	--	--	--	--	u	--	--	--
`DETAIL MODE`	*data-name-10* `PIC X(1).`	--	--	u	u	--	--	--	--	--	--
`WAITING TIME`	*data-name-11* `PIC 1(32) BIT.`	--	--	u	u	--	--	--	--	--	--

Legend:

1.: `RECEIVE` - Receive a message (the first segment)

2.: `RECEIVE` - Receive a message (An intermediate segment or the last segment)

3.: `SEND` - Send a message (the first segment)

4.: `SEND` - Send a message (An intermediate segment or the last segment)

5.: `DISABLE` - Terminate continuous-inquiry-response processing

6.: `RECEIVE` - Accept temporary-stored data

7.: `SEND` - Activate an application program

8.: `SEND` - Update temporary-stored data

9.: `SEND` - Execute an operation command

10.: `SEND` - Acquire a user journal

B: The value is returned from OpenTP1.

b: The value is returned from OpenTP1 under the specified conditions.

U: The value is specified in the UAP.

u: The value is specified in the UAP under the specified conditions.

--: Not applicable.

#: The `ROLLBACK` statement does not use a communication description entry.

Data communication facility

This section explains the programs used for the DML data communication facility. The COBOL-UAP creation programs for the data communication facility are as follows:

- RECEIVE - Receive a message
- SEND - Send a message

The above DML format, the values to be set in the data areas, and status codes vary from one protocol to another. For details, see the applicable *OpenTP1 Protocol* manual.

RECEIVE - Receive a message

Format

■ DATA DIVISION (communication description entry) specification

```
CD communication-description-name
   FOR {INPUT|I-O}
   [STATUS KEY IS data-name-1]
   [SYMBOLIC TERMINAL IS data-name-2]
   [MESSAGE DATE IS data-name-3]
   [MESSAGE TIME IS data-name-4].

01 unique-name-1.
   02 data-name-5  PIC  9(4) COMP.
   02 data-name-6  PIC  X(4).
   02 data-name-7  PIC  X(n).
```

■ PROCEDURE DIVISION (communication statement) specification

```
RECEIVE communication-description-name
    [FIRST] SEGMENT
       INTO unique-name-1.
```

Description

`RECEIVE` enables the following `CALL` interface facility:

- `CBLDCMCF('RECEIVE ')` - Receive a message.
- `CBLDCMCF('RECVSYNC')` - Receive a synchronous message.

The maximum length of a single segment that can be received is 32763 bytes. Note that the actual maximum length might be smaller than this value depending on the protocol. For details, see the relevant description in the applicable *OpenTP1 Protocol* manual.

In the case of receiving a message sent from a remote system via a protocol, the syntax of the `RECEIVE` statement varies from one protocol to another. For the syntax of the `RECEIVE` statement for receiving a message from a remote system, also see the relevant description in the applicable *OpenTP1 Protocol* manual.

Items specified in the communication description entry

■ `FOR` clause

Specify one of the following:

`INPUT`: Receive a non-inquiry message

`I-O`: Receive an inquiry message

- `STATUS KEY` clause

  Specify this clause to receive a status code. If this clause is omitted, a status code cannot be received.

- `SYMBOLIC TERMINAL` clause

  Specify the data item for referencing the logical terminal name of the message input source.

- `MESSAGE DATE` clause

  Specify the data item for referencing the date when the message is received in the format of `YYMMDD` (where `YY` indicates the year, `MM` indicates the month, and `DD` indicates the day).

- `MESSAGE TIME` clause

  Specify the data item for referencing the time when the message is received in the format of `HHMMSS00` (where `HH` indicates hours, `MM` indicates minutes, `SS` indicates second, and `00` is fixed).

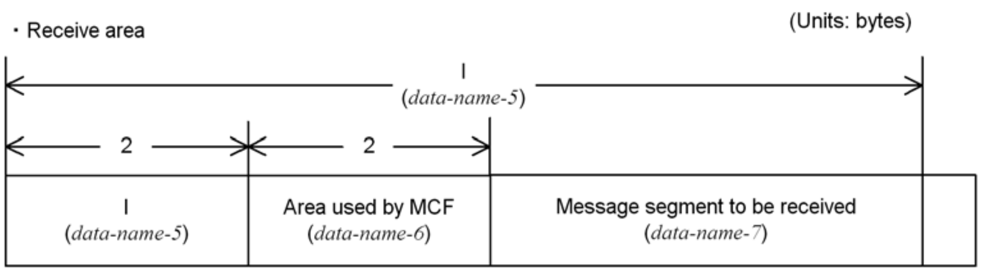
Items specified in the communication statement

- `FIRST`

  Specify this item to receive the first segment.

- *unique-name-1*

  Specify the data item indicating the area for receiving a segment. When the message is sent from the local system, the maximum length of receiving segment is 32000 bytes. When the message is sent from the remote system, the maximum length of receiving segment depends on the communication protocol supporting product.



Status codes

Status code	Explanation
`00000`	Normal termination.
	A segment shorter than the receive area was received. The blank area after the segment will be padded with spaces.
`71000`	The `RECEIVE` statement for receiving the first segment was executed more than once. To receive an intermediate segment or the last segment, execute the `RECEIVE` statement without specifying `FIRST`.
`71001`	The `RECEIVE` statement for receiving the next segment was executed after the last segment of the message was received. The `RECEIVE` statement executed immediately before receives a message completely. If the `RECEIVE` statement is executed again after this status code is returned, the status code "72000" is returned.
`71002`	An error occurred during input/output processing for the message queue.
	The message queue is in shutdown state.
`72000`	Return at MHP execution: The `RECEIVE` statement for receiving an intermediate segment or the last segment was executed before the `RECEIVE` statement for receiving the first segment was executed. To receive the first segment, execute the `RECEIVE` statement specifying `FIRST`. The `RECEIVE` statement was executed again after the status code "71000" was returned.
	Return at SPP execution: The `RECEIVE` statement cannot be executed from an SPP.
`72001`	The logical terminal name specified for the `SYMBOLIC TERMINAL` clause is invalid.
	The specified logical terminal cannot execute the `RECEIVE` statement.
`72013`	A segment exceeding the length of the receive area was received. The excess portion was truncated.
	A segment exceeding 32,767 bytes was received in the case of buffer format 2. The excess portion was truncated.
`72020`	The value specified for the `SYNCHRONOUS MODE` clause is invalid.
`72024`	The value specified for the `FOR` clause is invalid.
`72036`	The segment receive area (unique-name-1) is insufficient. Allocate an area of 5 bytes or more.
Other than the above	An unprecedented error (e.g., program damage) occurred.

SEND - Send a message

Format

For details on the format, see the applicable *OpenTP1 Protocol* manual.

Description

`SEND` enables the following `CALL` interface facilities:

- `CBLDCMCF('REPLY   ')` - Send a response message
- `CBLDCMCF('SEND    ')` - Send a message
- `CBLDCMCF('SENDRECV')` - Exchange a synchronous message
- `CBLDCMCF('SENDSYNC')` - Send a synchronous message

The maximum length of a single segment that can be received is 32763 bytes. The maximum length of a single message segment that can be sent is 32 kilobytes. Note that the actual maximum length might be smaller than this value depending on the protocol. For details, see the relevant description in the applicable *OpenTP1 Protocol* manual.

The syntax of the `SEND` statement for sending a message varies from one protocol to another. For the syntax of the `SEND` statement for sending a message to a remote system, also see the relevant description in the applicable *OpenTP1 Protocol* manual.

Note

When a capability equivalent to `CBLDCMCF('SEND')` is used, the message send order varies depending on the `mcfmuap -c order` specification in the UAP common definition of the MCF manager definition.

Service facility

This section gives the syntax and other information of the following COBOL-UAP creation programs which are used for the DML service facility:

- DISABLE - Terminate continuous-inquiry-response processing
- RECEIVE - Accept temporary-stored data
- ROLLBACK - Enable MHP rollback
- SEND - Activate an application program
- SEND - Update temporary-stored data
- SEND - Execute an operation command
- SEND - Acquire a user journal

DISABLE - Terminate continuous-inquiry-response processing

Format

- DATA DIVISION (communication description entry) specification

```
CD  communication-description-name
    FOR  I-O  STORAGE
    [STATUS KEY IS data-name-1]
```

- PROCEDURE DIVISION specification

```
DISABLE   communication-description-name.
```

Description

`DISABLE` enables the following `CALL` interface facility:

- `CBLDCMCF('CONTEND')`

  - Terminate continuous-inquiry-response processing.

Items specified in the communication description entry

- `STATUS KEY` clause

Specify this clause to receive a status code. If this clause is omitted, a status code cannot be received.

Item specified in the communication statement

None

Status codes

Status code	Explanation
`00000`	Normal termination.
`72000`	<Return at MHP execution> The `DISABLE` statement that terminates continuous-inquiry-response was used out of sequence. The `DISABLE` statement that terminates continuous-inquiry-response termination was used before the `RECEIVE` statement that receives first segment was called from the MHP.
	<Return at SPP execution> The `DISABLE` statement that terminates continuous-inquiry-response cannot be used from an SPP.

Status code	Explanation
`72101`	The `DISABLE` statement that terminates continuous-inquiry-response was used from an MHP for which `type=cont` (continuous-inquiry-response type) was not specified in the MCF application definition.
`72107`	The `DISABLE` statement that terminates continuous-inquiry-response termination was already used.
`72111`	The `SEND` statement that sends response message in which the name of the continuous-inquiry-response type application to be activated next was specified was used, then the `DISABLE` statement that terminates continuous-inquiry-response was used.
	The `SEND` statement that starts application program in which the name of the continuous-inquiry-response type application to be activated next was specified was used, then the `DISABLE` statement that terminates continuous-inquiry-response was used.
Other than the above	An unprecedented error (e.g., program damage) occurred.

RECEIVE - Accept temporarily-stored data

Format

- DATA DIVISION (communication description entry) specification

```
CD   communication-description-name
     FOR {INPUT|I-O} STORAGE
     [STATUS  KEY  IS  data-name-1] .

01 unique-name-1.
   02 data-name-2  PIC  9(4) COMP.
   02 data-name-3  PIC  X(4).
   02 data-name-4  PIC  X(n).
```

- PROCEDURE DIVISION specification

```
RECEIVE   communication-description-name   {MESSAGE|SEGMENT}
   INTO   unique-name-1.
```

Description

`RECEIVE` enables the following `CALL` interface facility:

- `CBLDCMCF('TEMPGET')` - Receive temporary-stored data.

Items specified in the communication description entry

- `FOR` clause

  Specify `INPUT STORAGE` or `I-O STORAGE`.

- `STATUS KEY` clause

  Specify this clause to receive a status code. If this clause is omitted, a status code cannot be received.

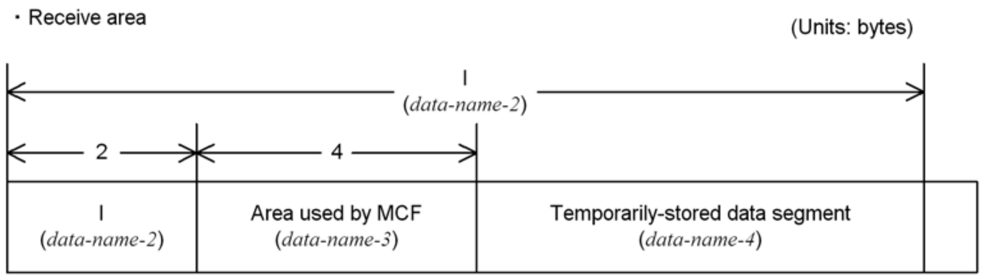
Items specified in the communication statement

- `MESSAGE`, `SEGMENT`

  Specify `MESSAGE` or `SEGMENT`.

- *unique-name-1*

  Specify the data item indicating the area for receiving temporary-stored data. The following figure shows the format of *unique-name-1* after the message is received:



Status codes

Status codes	Status codes
`00000`	Normal termination.
`72000`	The `RECEIVE` statement that receives temporary-stored data cannot be used from an SPP.
`72013`	Temporary-stored data exceeding the receive area length was received. The excess portion was truncated because the data exceeds 32,761 bytes.
`72036`	The value specified for the receive area length is less then 7 bytes.
`72101`	`RECEIVE` statement that receives the temporary-stored data was used by an MHP which is not continuous-inquiry-response type (`type=cont`).
`72106`	`RECEIVE` statement that receives the temporary-stored data was used before `RECEIVE` statement that receives the first segment.
`72107`	`RECEIVE` statement that receives the temporary-stored data was used after `DISABLE` statement that terminates the continuous inquiry response.
Other than the above	An unprecedented error (e.g., program damage) occurred.

ROLLBACK - Enable MHP rollback

Format

■ PROCEDURE DIVISION specification

```
ROLLBACK    [WITH STOPPING] .
```

Description

ROLLBACK enables the following CALL interface facility:

- CBLDCMCF('ROLLBACK') Enable MHP rollback.

The ROLLBACK statement used from an SPP is ignored. The ROLLBACK statement is ignored if it is used from an MHP with the nontransaction attribute.

Item specified in the communication statement

■ WITH STOPPING clause

Specify this clause to stop returning control to an MHP when the MHP is terminated abnormally. Omit this clause to return control to the MHP. If the ROLLBACK statement with this clause omitted is used before a message is received, the ROLLBACK statement is ignored.

Status codes

There is no status code of the ROLLBACK statement.

SEND - Activate an application program

Format 1

- (When specifying the message to be passed to the application program to be activated and activating the application program)
- DATA DIVISION (communication description entry) specification

```
CD   communication-description-name
     FOR  OUTPUT  PROGRAM
     [STATUS  KEY  IS  data-name-1]
     SYMBOLIC  TERMINAL  IS  data-name-2
     [ACTIVE {INTERVAL|TIME}  IS  data-name-9].

01 unique-name-1.
   02 data-name-10  PIC  9(4) COMP.
   02 data-name-11  PIC  X(2).
   02 data-name-12  PIC  X(n).
01 unique-name-2.
   02 data-name-13  PIC  X(1).
```

- PROCEDURE DIVISION specification

```
SEND  communication-description-name  FROM  unique-name-1
     [WITH {ESI|EMI|unique-name-2}] .
```

Format 2

- (When reporting termination of message transfer after a request to transfer a segment other than the last segment is sent)
- DATA DIVISION (communication description entry) specification

```
CD   communication-description-name
     FOR  OUTPUT  PROGRAM
     [STATUS  KEY  IS  data-name-1]
     SYMBOLIC  TERMINAL  IS  data-name-2.
```

- PROCEDURE DIVISION specification

```
SEND  communication-description-name  WITH  EMI.
```

Description

SEND enables the following CALL interface facility:

- CBLDCMCF('EXECAP ') Activate an application program.

The maximum length of a single message segment that can be sent is 32 kilobytes. Note that the actual maximum length might be smaller than this value depending on the protocol. For details, see the relevant description in the applicable *OpenTP1 Protocol* manual.

Items specified in the communication description entry

- `FOR` clause

  Specify `OUTPUT` indicating a send-only message.

- `STATUS KEY` clause

  Specify this clause to receive a status code. If this clause is omitted, a status code cannot be received.

- `SYMBOLIC TERMINAL` clause

  Specify the data item for which the logical terminal name was specified.

- `ACTIVE` clause

  This clause is specified to timer-start the application program.

  `INTERVAL`: Interval timer start

  `TIME`: Time point timer start

  data-name-9

  Interval timer start

  Specify a period of time in hours, minutes, and seconds. The MHP or SPP will start the specified period of time after the application program start statement is issued. The period of time is specified in the format of `HHMMSS00` where `HH` indicates hours, `MM` indicates minutes, `SS` indicates seconds, and `00` is fixed. The range of specifiable times is from `00000100` (start after one second) to `99595900` (start after 99 hours 59 minutes 59 seconds).

  Time point timer start

  Specify the time at which the MHP or SPP is to start. The time is specified in the format of `HHMMSS00` where `HH` indicates hours, `MM` indicates minutes, `SS` indicates seconds, and `00` is fixed. The range of specifiable times is from `00000000` (start at 00:00:00) to `23595900` (start at 23:59:59) in local time.

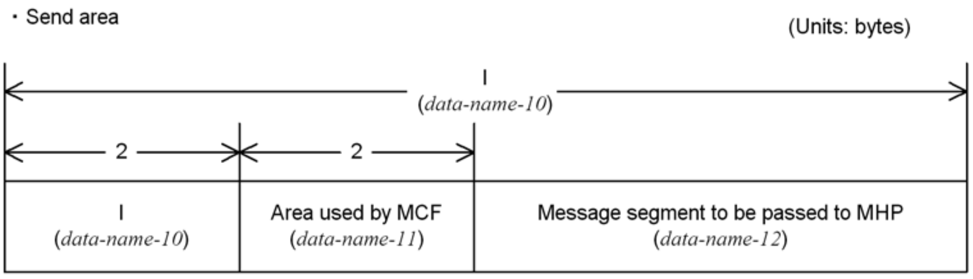
  If nothing is specified, immediate start is assumed.

  Since OpenTP1 monitors timeout at fixed intervals, an error arises between the time specified for `timer` (for COBOL, *data-name-D*) of the function `dc_mcf_timer_set()` [`CBLDCMCF('TIMERSET')`] and the time that elapses before actual detection of timeout. The accuracy of time monitoring depends on the value for the time monitoring interval specified for the `btim` operand in the `-t` option of the MCF communication configuration definition `mcfttim`.

Items specified in the communication statement

■ *unique-name-1*

Specify the data item indicating the send area of the message segment which is to be passed to the application program to be activated. The following figure shows the format of the segment which is passed to MHP to be activated.



■ `WITH` clause

Specify whether the segment to be passed to the application program to be activated is the last segment of a logical message.

`ESI`: Specify ESI for the first segment or an intermediate segment.

`EMI`: Specify EMI for the last segment. EMI must also be specified if the message to be passed consists of a single segment.

unique-name-2: Data item for which either of the following values was specified

`'1'`: `ESI` (first or intermediate segment)

`'2'`: `EMI` (last or single segment)

If this clause is omitted, `EMI` (last or single segment) is assumed.

Status codes

Status code	Explanation
`00000`	Normal termination.
`71002`	An error occurred during input/output processing for the message queue.
	The message queue is in shutdown state.
	The message queue was not allocated.
	The value specified for the segment length exceeds 32000 bytes.

Status code	Explanation
	The `SEND` statement that starts application program cannot be accepted because the MCF is being terminated.
`71003`	The message queue is full.
`71004`	The buffer for storing messages could not be acquired in the memory.
`71008`	An attempt was made to start the application program, but the management table of the send destination could not be acquired.
	The local memory of the process is insufficient.
`72000`	<Return at MHP execution> The `SEND` statement that starts application program was used out of sequence. The `SEND` statement that starts application program was used before the `RECEIVE` statement that receives first segment was used from the `MHP`.
	<Return at SPP execution> The `SEND` statement that starts application program was called from SPP processing which was not transaction.
`72001`	The specified application name is not defined with the MCF.
	The application name is incorrect.
	The application startup process name is not specified in the communication service definition (`mcfmcname` definition command) included in the MCF manager definition.
	No application startup process identifier is specified in the environment definition (`-p` option to the `mcfaenv` definition command) included in the MCF application definition for the application startup process.
	The following two values do not match: • Application startup process identifier specified in the application environment definition (`-p` option to the `mcfaenv` definition command). • Application startup process identifier specified in the communication configuration definition (`mcftenv` definition command) for the application startup process.

Status code	Explanation
	If a non-response type MHP or SPP is to be started: • No logical terminal (the `lname` operand of the `-n` option to the `mcfaalcap` definition command) is specified in the application attribute definition for the application to be started. • The logical terminal specified in the application attribute definition for the application to be started is not defined in the communication configuration definition (`mcftalcle` definition command) for the application startup process. • The logical terminal specified in the application attribute definition for the application to be started is not of the send-only (`=send`) type. • The logical terminal specified in the application attribute definition for the application to be started cannot use the application startup process.
	If an MHP of the response type or continuous inquiry-response type is to be started: • No internal communication path (`cname` operand of the `-n` option to the `mcfaalcap` definition command) is specified in the application attribute definition for the application to be started. • The internal communication path specified in the application attribute definition for the application to be started is not defined in the communication configuration definition (`-c` option to the `mcftpsvr` definition command) for the application startup process. • The logical terminal specified in the communication configuration definition (`mcftalcle` definition command) for the application startup process is not of the inquiry-response type (`=request`).
	If an application is to be started from an SPP: • The `mcf_psv_id` operand in the user service definition or user service default definition for the caller UAP is assigned no application startup process identifier. • The following two values do not match: Application startup process identifier assigned to the `mcf_psv_id` operand in the user service definition or user service default definition for the caller UAP Application startup process identifier specified in the communication configuration definition (`-s` option to the `mcftenv` definition command) for the application startup process or in the application environment definition (`-p` option to the `mcfaenv` definition command) • The following two values do not match: MCF manager identifier assigned to the `mcf_mgrid` operand in the user service definition or user service default definition for the caller UAP Identifier of the MCF manager to which the application startup process be/ longs
`72005`	The value specified for the send segment length was less than 5 bytes when the application program with `ESI` (first or intermediate segment) specified for the `WITH` clause was activated.
`72007`	From a response type (`type=ans`) MHP, another response type MHP was started after the response message is sent.

Status code	Explanation
	From a continuous-inquiry-response type (`type=cont`) MHP, another continuous-inquiry-response type MHP was started after the response message is sent.
`72009`	A response type (`type=ans`) MHP was started more than once.
	A continuous-inquiry-response type (`type=cont`) MHP was started more than once.
`72011`	From an MHP which is not response type (`type=cont`), a response type MHP was started.
	From an MHP which is not continuous-inquiry-response type (`type=cont`), a continuous-inquiry-response type MHP was started.
`72023`	The contents of *data-name-3* specified for the `ACTIVE TIME` clause are null.
`72024`	The value specified for the `FOR` clause is invalid.
`72026`	The value specified for the `WITH` clause is invalid.
`72041`	The `SEND` statement that starts application program was used incorrectly. The value specified for the send segment length was less than 4 bytes when a single segment was sent.
`72044`	The continuous-inquiry-response termination statement was used, the name of the application to be activated next was specified, then the `SEND` statement that starts application program was used.
`72108`	The value specified for *data-name-9* exceeds the limit.
`72109`	An MHP for which `type=cont` (continuous-inquiry-response type) was specified in the MCF application definition was activated with timer start specified.
`77001`	The logical terminal (LE) corresponding to the application to be activated is being started and cannot be used, or it is not available due to another factor.
Other than the above	An unprecedented error (e.g., program damage) occurred.

Notes

1. The activation order of application programs varies depending on the `mcfmuap -c order` specification in the UAP common definition of the MCF manager definition.
2. If you use a single service function to update a TAM or DAM file and call the `SEND` statement to start an application that will reference the updated file, make sure that the application will lock the file. If the application references the file without locking the file, the data existing before the file was updated might be referenced.

SEND - Update temporarily-stored data

Format

■ DATA DIVISION (communication description entry) specification

```
CD   communication-description-name
    FOR  I-O  STORAGE
     [STATUS  KEY  IS  data-name-1].

01 unique-name-1.
   02 data-name-2  PIC  9(4) COMP.
   02 data-name-3  PIC  X(4).
   02 data-name-4  PIC  X(n).
```

■ PROCEDURE DIVISION specification

```
SEND  communication-description-name  FROM  unique-name-1.
```

Description

SEND enables the following CALL interface facility:

- CBLDCMCF('TEMPPUT ') Updates temporary-stored data.

Items specified in the communication description entry

■ FOR clause

Specify I-O STORAGE.

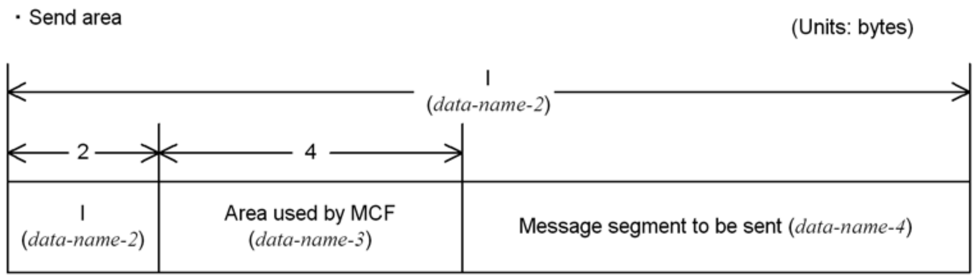
■ STATUS KEY clause

Specify this clause to receive a status code. If this clause is omitted, a status code cannot be received.

Item specified in the communication statement

■ *unique-name-1*

Specify the data item storing temporary-stored data. The following figure shows the send segment format:


・Send area
(Units: bytes)
l (data-name-2)
2
4
l (data-name-2)
Area used by MCF (data-name-3)
Message segment to be sent (data-name-4)

Status codes

Status code	Explanation
00000	Normal termination.
71103	The area for updating temporary-stored data could not be acquired.
72000	The SEND statement that updates temporary-stored data cannot be used from an SPP.
72024	The value specified for the FOR clause is invalid.
72035	The specified update data length exceeds the length of the temporary-stored data storage area specified in the MCF application definition.
	The value specified for the update data length is less than 7 bytes.
72101	SEND statement that updates the temporary-stored data was used by an MHP which is not continuous-inquiry-response type (type=cont).
72105	SEND statement that receives the temporary-stored data was used before RECEIVE statement.
72106	SEND statement that updates the temporary-stored data was used before RECEIVE statement that receives the first segment.
72107	SEND statement that updates the temporary-stored data was used after DISABLE statement that terminates the continuous inquiry response.
Other than the above	An unprecedented error (e.g., program damage) occurred.

SEND - Execute an operation command

Format

■ DATA DIVISION (communication description entry) specification

```
CD   communication-description-name
     FOR {OUTPUT|I-O} COMMAND
     [STATUS  KEY  IS  data-name-1].

01 unique-name-1.
   02 data-name-2  PIC  9(4) COMP.
   02 data-name-3  PIC  X(2) VALUE LOW-VALUE.
   02 data-name-4  PIC  X(n).
01 unique-name-2.
   02 data-name-5  PIC  9(4) COMP.
   02 data-name-6  PIC  X(2) VALUE LOW-VALUE.
   02 data-name-7  PIC  X(n).
```

■ PROCEDURE DIVISION specification

```
SEND  communication-description-name  FROM  unique-name-1
     [BEFORE  RECEIVING  MESSAGE  INTO  unique-name-2] .
```

Description

`SEND` enables the following `CALL` interface facility:

- `CBLDCADM('COMMAND')` Execute an operation command.

Note the following points on using commands with DML unlike when using `CALL` interface commands:

- Only the standard error information of command processing results is returned. The character string output to the standard output file (e.g., display command (`~ls`) execution results) cannot be received. To receive the character string output to the standard output file, use the `CBLDCADM('COMMAND ')`.
- When the command for output to the standard output file is executed, the status code is not `00000` even if the shell termination code is `0`.
- Status code `01804` or `01805` is returned indicating that the data output to the standard error output file was too large to be stored in the area in the following case:
    - The statement with `OUTPUT COMMAND` specified for the `FOR` clause was executed and the data was output to the standard error file.
- COBOL85 is a prerequisite for this DML.

Items specified in the communication description entry

- `FOR` clause

  Specify either of the following value.

  `OUTPUT COMMAND`: Specify `OUTPUT COMMAND` when not receiving the command execution results output to the standard error output file. When this value is specified, `0` is assumed for the length of the area for receiving data output to the standard error output file.

  `I-O COMMAND`: Specify `I-O COMMAND` when receiving the command execution results output to the standard error output file.

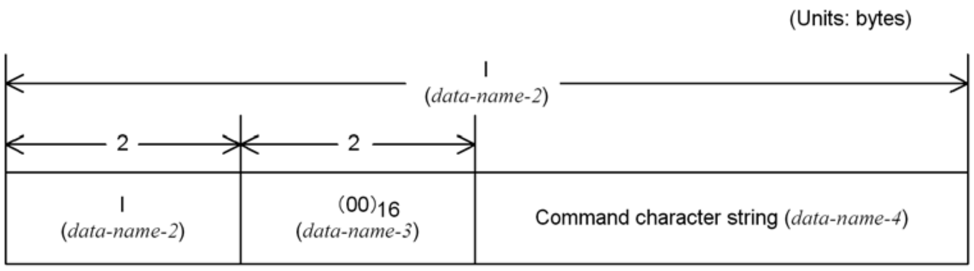
- `STATUS KEY` clause

  Specify this clause to receive a status code. If this clause is omitted, a status code cannot be received.

Items specified in the communication statement

- *unique-name-1*

  This area is used to specify the command to be executed by using `SEND` statement. The following figure shows the format for specifying the character string of the command:

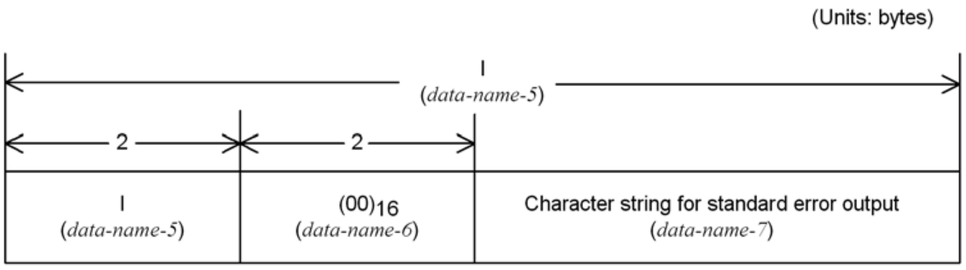


- `BEFORE` clause

  Specify this clause to receive the character string output to the standard error output file when the command is executed. The `BEFORE` clause must be specified when `I-O command` is specified for the `FOR` clause.

  unique-name-2: Data item for storing the character string output to the standard error output file. A character string which can be stored is up to (*unique-name-2* area length) - 4. The excess portion is truncated.

  The following figure shows the format of the receive character string (*unique-name-2*):



Status codes

Status code	Explanation
`00000`	The shell termination code is `0` (normal termination of command execution). When `I-O COMMAND` was specified for the `FOR` clause, the character string output to the standard error output file was stored.
`01801`	The shell termination code is not `0` (abnormal termination of command execution). When `I-O COMMAND` was specified for the `FOR` clause, the character string output to the standard error output file was stored.
`01802`	The value specified for the data-name is invalid. Verify that there are no errors in the area length specification set in unique name 1 and unique name 2 and in locations where `0` must be specified.
`01803`	Data was output to the standard output file. (This statement cannot receive the character string output to the standard output file.)
`01804`	The data output to the standard error output file was too large to be stored in the area.
`01805`	Data was output to the standard output file (this statement cannot receive the character string output to the standard output file), and the data output to the standard error output file was too large to be stored in the area.
`01806`	A system call (`close, pipe, dup,` or `read`) could not be used.
`01807`	`CBLDCRPC('OPEN      ')` was not called.
`01808`	The memory became insufficient.

SEND - Acquire a user journal

Format

■ DATA DIVISION (communication description entry) specification

```
CD   communication-description-name
     FOR  OUTPUT  JOURNAL
      [STATUS  KEY  IS  data-name-1] .

01 unique-name-1.
   02 data-name-2  PIC  9(4) COMP.
   02 data-name-3  PIC  X(2).
   02 data-name-4  PIC  X(1).
   02 data-name-5  PIC  X(n).
```

■ PROCEDURE DIVISION specification

```
SEND  communication-description-name  FROM  unique-name-1.
```

Description

SEND enables the following CALL interface facility:

- CBLDCJNL('UJPUT   ') - Acquire a user journal.

TP1/Message Control and COBOL85 are required to use this DML.

Items specified in the communication description entry

■ FOR clause

Specify OUTPUT JOURNAL.

■ STATUS KEY clause

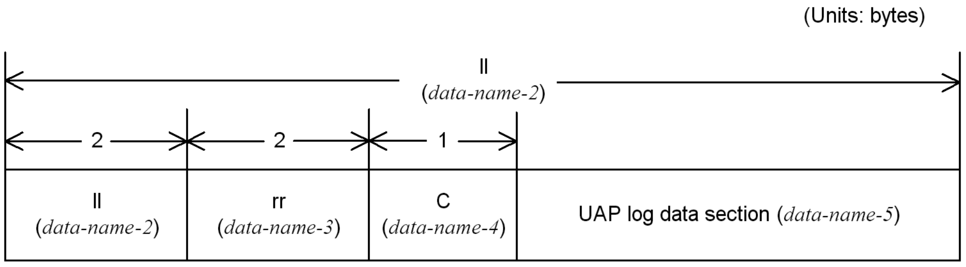
Specify this clause to receive a status code. If this clause is omitted, a status code cannot be received.

Item specified in the communication statement

■ *unique-name-1*

UAP historical information is stored in the specified area. For *unique-name-1*, specify the item defined in the working storage section or the linkage section.

The figure below shows the format of the UAP historical information (*unique-name-1*) to be stored. Create/specify this area in the UAP.



ll (*data-name-2*): Length of UAP log information (length of UAP log data section + 5)
$5 \leqq ll \leqq$ (`jnl_max_datasize` operand value in the target system's journal service definition - 8)
rr (*data-name-3*): No value is set
C (*data-name-4*): Set the UJ code in the range from $(00)_{16}$ to $(FF)_{16}$

Status codes

Status code	Explanation
`00000`	Normal termination.
`01101`	The parameter format is invalid.
`01102`	The value specified for the user journal length is less than 5 byte.
`01103`	The specified user journal length exceeds the limit.
`01104`	The `SEND` statement for acquiring user journals was used from UAP processing which was not a transaction.
`01105`	Preparation to start the UAP has not been done.

Chapter

4. X/Open-compliant Application Programming Interface

This chapter explains the syntax of COBOL-UAP creation programs compliant with application programming interfaces stipulated by X/Open.

This chapter contains the following sections:

X/Open-compliant function

Table 4-1 gives the relationship between X/Open-compatible APIs (XATMI and TX interfaces) and functions. Table 4-2 gives the relationship between X/ Open-compatible API functions and OpenTP1 UAPs.

Table 4-1: Relationship between X/Open-compatible APIs and functions

API	Name and role of X/Open-compatible function	
XATMI interface	`TPACALL`	Send a service request.
	`TPADVERTISE`	Advertise a service name.
	`TPCALL`	Send a service request and synchronously await its reply.
	`TPCANCEL`	Cancel a communication handle for an outstanding reply.
	`TPCONNECT`	Establish a conversational service connection.
	`TPDISCON`	Terminate a conversational service connection abortively.
	`TPGETRPLY`	Get a reply from a previous service request.
	`TPRECV`	Receive a message in a conversational connection.
	`TPRETURN`	Return from a service routine.
	`TPSEND`	Send a message in a conversational connection.
	`TPSVCSTART`	Start a service routine.
	`TPUNADVERTISE`	Unadvertise a service name.
TX interface	`TXBEGIN`	Begin a global transaction.
	`TXCLOSE`	Close a set of resource managers.
	`TXCOMMIT`	Commit a global transaction.
	`TXINFORM`	Return global transaction information.
	`TXOPEN`	Open a set of resource managers.
	`TXROLLBACK`	Roll back a global transaction.
	`TXSETCOMMITRET`	Set `commit_return` characteristic.
	`TXSETTIMEOUT`	Set `transaction_timeout` characteristic.

API	Name and role of X/Open-compatible function	
	TXSETTRANCTL	Set transaction_control characteristic.

Table 4-2: Relationship between X/Open-compatible API functions and OpenTP1 UAPs

X/Open-compatible API	SUP		SPP			MHP		UAP that handles offline work
	Outside transaction range	Inside transaction range (root)	Outside transaction range	Transaction range		Outside transaction range	Inside transaction range (root)	
				Root	Non root			
TPACALL	Y	Y	Y	Y	Y	--	--	--
TPADVERTISE	--	--	Y#1	Y#1	Y#1	--	--	--
TPCALL	Y	Y	Y	Y	Y	--	--	--
TPCANCEL	Y	Y	Y	Y	Y	--	--	--
TPCONNECT	Y	Y	Y	Y	Y	--	--	--
TPDISCON	Y	Y	Y	Y	Y	--	--	--
TPGETRPLY	Y	Y	Y	Y	Y	--	--	--
TPRECV	Y	Y	Y	Y	Y	--	--	--
TPRETURN	--	--	Y#2	Y#2	Y#2	--	--	--
TPSEND	Y	Y	Y	Y	Y	--	--	--
TPSVCSTART#3	--	--	--	--	--	--	--	--
TPUNADVERTISE	--	--	Y#1	Y#1	Y#1	--	--	--
TXBEGIN#4	Y	--	Y	--	--	Y	--	--
TXCLOSE	Y	--	Y	--	--	--	--	--
TXCOMMIT with TX_CHAINED specified#4	--	Y	Y	--	--	--	--	--

X/Open-compatible API	SUP		SPP			MHP		UAP that handles offline work
	Outside transaction range	Inside transaction range (root)	Outside transaction range	Transaction range		Outside transaction range	Inside transaction range (root)	
				Root	Non root			
TXCOMMIT with TX_UNCHAINED specified[#4]	--	Y	Y	--	--	--	--	--
TXINFORM	Y	Y	Y	Y	Y	--	--	--
TXOPEN	Y	--	Y	--	--	--	--	--
TXROLLBACK with TX_CHAINED specified[#4]	--	Y	--	Y	--	--	--	--
TXROLLBACK with TX_UNCHAINED specified[#4]	--	Y	--	Y	--	--	--	--
TXSETCOMMIITRET[#4]	Y	Y	Y	Y	Y	--	--	--
TXSETTIMEOUT[#4]	Y	Y	Y	Y	Y	--	--	--
TXSETTRANCTL[#4]	Y	Y	Y	Y	Y	--	--	--

Legend:

Y: The function can be called from the UAP.

--: The function cannot be called from the UAP.

Note:

Outside transaction range for MHP means the range of MHPs with the nontransaction attribute or the MHP main program.

#1: Functions marked[#3] can be invoked only from within service programs.

#2: Functions marked[#4] are used only to make XATMI-interfaced service programs return.

#3: TPSVCSTART is an API function that service programs always invoke.

#4: For the UAP which issues a function marked[#2], specify atomic_update=Y in the user service definition.

XATMI-interfaced application programming interface (TP~)

This section explains the syntax of the API functions which implement the XATMI interface. The text in this section is quoted from *7. COBOL Language Reference Manual Pages* which is the syntax reference section of the *X/Open CAE Specification Distributed TP: The XATMI Specification* published by X/Open Company Limited.

Additional notes on using these functions from OpenTP1 UAPs are enclosed in << >> symbols.

The XATMI interface has the following API functions. The function `TPINTRO` defines data areas invoked from API functions using the `COPY` statement.

- `TPINTRO` - COPY files for the XATMI interface
- `TPACALL` - Send a service request.
- `TPADVERTISE` - Advertise a service name.
- `TPCALL` - Send a service request and synchronously await its reply.
- `TPCANCEL` - Cancel a communication handle for an outstanding reply.
- `TPCONNECT` - Establish a conversational service connection.
- `TPDISCON` - Terminate a conversational service connection abortively.
- `TPGETRPLY` - Get a reply from a previous service request.
- `TPRECV` - Receive a message in a conversational connection.
- `TPRETURN` - Return from a service routine.
- `TPSEND` - Send a message in a conversational connection.
- `TPSVCSTART` - Start a service routine.
- `TPUNADVERTISE` - Unadvertise a service name.

XATMI-interfaced API functions whose names begin with TP can be used only with the TP1/Server Base. They cannot be used with the TP1/LiNK.

TPINTRO - COPY files for the XATMI interface

Description

The following return codes and setting definitions are used by the COBOL XATMI routines. XATMI interface providers supply these definitions in the four `COPY` files listed below. Shown for each are the minimum set of record definitions and settings that must be defined in each `COPY` file.

<<The COBOL records shown below are stored in the `$DCDIR/include/COBOL/` directory. When compiling the UAP, you must designate this directory as the location of the `COPY` file. For details about the specifications needed for compilation, see a manual for the COBOL language version you are using.>>

```
*
* TPSTATUS.cbl
*
 05  TP-STATUS  PIC S9(9) COMP-5.
          88  TPOK            VALUE 0.
          88  TPEBADDESC      VALUE 2.
          88  TPEBLOCK        VALUE 3.
          88  TPEINVAL        VALUE 4.
          88  TPELIMIT        VALUE 5.
          88  TPENOENT        VALUE 6.
          88  TPEOS           VALUE 7.
          88  TPEPROTO        VALUE 9.
          88  TPESVCERR       VALUE 10.
          88  TPESVCFAIL      VALUE 11.
          88  TPESYSTEM       VALUE 12.
          88  TPETIME         VALUE 13.
          88  TPETRAN         VALUE 14.
          88  TPEGOTSIG       VALUE 15.
          88  TPEITYPE        VALUE 17.
          88  TPEOTYPE        VALUE 18.
          88  TPEEVENT        VALUE 22.
          88  TPEMATCH        VALUE 23.
 05  TP-EVENT  PIC S9(9) COMP-5.
          88  TPEV-NOEVENT    VALUE 0.
          88  TPEV-DISCONIMM  VALUE 1.
          88  TPEV-SENDONLY   VALUE 2.
          88  TPEV-SVCERR     VALUE 3.
          88  TPEV-SVCFAIL    VALUE 4.
          88  TPEV-SVCSUCC    VALUE 5.
```

The following COBOL record is used whenever sending or receiving application data. `REC-TYPE` indicates the type of data record that is to be sent. `SUB-TYPE` indicates the name of the sub-type for a particular type. `LEN` contains the amount of data to send and the amount received.

```
*
* TPTYPE.cbl
*
 05  REC-TYPE       PIC X(8).
          88  X-OCTET         VALUE "X-OCTET".
          88  X-COMMON        VALUE "X-COMMON".
 05  SUB-TYPE       PIC X(16).
 05  LEN            PIC S9(9)  COMP-5.
          88  NO-LENGTH       VALUE 0.
 05  TPTYPE-STATUS  PIC S9(9)  COMP-5.
          88  TPTYPEOK        VALUE 0.
          88  TPTRUNCATE      VALUE 1.
```

The following COBOL record is used by functions to pass settings to and from the communication resource manager.

```
*
* TPSVCDEF.cbl
*
 05  COMM-HANDLE        PIC S9(9)    COMP-5.
 05  TPBLOCK-FLAG       PIC s9(9)    COMP-5.
          88  TPBLOCK      VALUE 0.
          88  TPNOBLOCK    VALUE 1.
 05  TPTRAN-FLAG        PIC S9(9)    COMP5.
          88  TPTRAN       VALUE 0.
          88  TPNOTRAN     VALUE 1.
 05  TPREPLY-FLAG       PIC S9(9)    COMP5.
          88  TPREPLY      VALUE 0.
          88  TPNOREPLY    VALUE 1.
 05  TPTIME-FLAG        PIC S9(9)    COMP5.
          88  TPTIME       VALUE 0.
          88  TPNOTIME     VALUE 1.

 05  TPSIGRSTRT-FLAG    PIC S9(9)    COMP5.
          88  TPNOSIGRSTRT VALUE 0.
          88  TPSIGRSTRT   VALUE 1.
 05  TPGETANY-FLAG      PIC S9(9)    COMP5.
          88  TPGETHANDLE  VALUE 0.
          88  TPGETANY     VALUE 1.
 05  TPSENDRECV-FLAG    PIC S9(9)    COMP5.
          88  TPSENDONLY   VALUE 0.
          88  TPRECVONLY   VALUE 1.
 05  TPNOCHANGE-FLAG    PIC S9(9)    COMP5.
          88  TPCHANGE     VALUE 0.
          88  TPNOCHANGE   VALUE 1.
 05  TPSERVICETYPE-FLAG PIC S9(9)    COMP5.
          88  TPREQRSP     VALUE 0.
          88  TPCONV       VALUE 1.
 05  SERVICE-NAME       PIC X(15).
```

The following COBOL record is used by `TPRETURN` to indicate the status of the transaction.

```
*
* TPSVCRET.cbl
*
 05  TP-RETURN-VAL  PIC S9(9)    COMP-5.
           88  TPSUCCESS    VALUE 0.
           88  TPFAIL       VALUE 1.
 05  APPL-CODE      PIC S9(9)    COMP-5.
```

TPACALL - Send a service request

Format

```
01  TPSVCDEF-REC.
    COPY  TPSVCDEF.
01  TPTYPE-REC.
    COPY  TPTYPE.
01  DATA-REC.
    COPY  Data record definition.
01  TPSTATUS-REC.
    COPY  TPSTATUS.

CALL "TPACALL" USING TPSVCDEF-REC TPTYPE-REC
                     DATA-REC  TPSTATUS-REC.
```

Description

TPACALL sends a request message to the service named by SERVICE-NAME. DATA-REC is the record to be sent and LEN specifies the amount of data in DATA-REC that should be sent. Note that if DATA-REC is a record of a type that does not require a length to be specified, LEN is ignored (and may be 0). If DATA-REC is a record type in which the length must be specified, do not specify 0 in LEN. If REC-TYPE does not have a subtype, SUB-TYPE is ignored (and may be SPACES). If REC-TYPE is SPACES, DATA-REC and LEN are ignored and a request is sent with no data portion, REC-TYPE and SUB-TYPE must match one of the types and sub-types recognized by SERVICE-NAME. Note that for each request sent while in transaction mode, a corresponding reply must ultimately be received.

<<Data areas>>

- <<TPSVCDEF-REC

  Specify a value indicating the TPACALL operation. The value specified here determines the return value. The specifiable values and their meanings will be explained later.>>

- <<TPTYPE-REC

  Indicates the record type and subtype record name of the send data.>>

- <<DATA-REC

  Points to the send data.>>

- <<TPSTATUS-REC

  Will be assigned the return value indicating the result of TPACALL execution.>>

  The valid settings of TPSVCDEF-REC are as follows:

`TPNOTRAN`

If the caller is in transaction mode and this setting is used, when `SERVICE-NAME` is invoked, it is not performed on behalf of the caller's transaction. If `SERVICE-NAME` does not support transactions, this setting must be used when the caller is in transaction mode. A caller in transaction mode that uses this setting is still subject to the transaction timeout (and no other). If a service fails that was invoked with this setting, the caller's transaction is not affected. Either `TPNOTRAN` or `TPTRAN` must be set.

`TPTRAN`

If the caller is in transaction mode and this setting is used, when `SERVICE-NAME` is invoked, it is performed on behalf of the caller's transaction. This setting is ignored if the caller is not in transaction mode. Either `TPNOTRAN` or `TPTRAN` must be set.

`TPNOREPLY`

This setting informs `TPACALL` that a reply is not expected. When `TPNOREPLY` is set, the routine returns `TPOK` on success and sets `COMM-HANDLE` to `0`, an invalid communication handle. When the caller is in transaction mode, this setting cannot be used when `TPTRAN` is also set. Either `TPNOREPLY` or `TPREPLY` must be set.

`TPREPLY`

This setting informs `TPACALL` that a reply is expected. When `TPREPLY` is set, the routine returns `TPOK` on success and sets `COMM-HANDLE` to a valid communication handle. When the caller is in transaction mode, this setting must be used when `TPTRAN` is also set. Either `TPNOREPLY` or `TPREPLY` must be set.

`TPNOBLOCK`

The request is not sent if a blocking condition exists (for example, the internal buffers into which the message is transferred are full). Either `TPNOBLOCK` or `TPBLOCK` must be set.

`TPBLOCK`

When `TPBLOCK` is specified and a blocking condition exists, the caller blocks until the condition subsides or a timeout occurs (either transaction or blocking timeout). Either `TPNOBLOCK` or `TPBLOCK` must be set.

`TPNOTIME`

This setting signifies that the caller is willing to block indefinitely and wants to be immune to blocking timeouts. Transaction timeouts may still occur. Either `TPNOTIME` or `TPTIME` must be set.

`TPTIME`

This setting signifies that the caller receives blocking timeouts if a blocking condition exists and the blocking time is reached. Either TPNOTIME or TPTIME must be set.

TPSIGRSTRT

If a signal interrupts any underlying system calls, the interrupted system call is re-issued. Either TPNOSIGRSTRT or TPSIGRSTRT must be set.

TPNOSIGRSTRT

If a signal interrupts any underlying system calls, the interrupted system call is not restarted and the routine fails. Either TPNOSIGRSTRT or TPSIGRSTRT must be set.

Return value

Upon successful completion, TPACALL sets TP-STATUS to TPOK. In addition, if TPREPLY was set in TPSVCDEF-REC, TPACALL returns a valid communication handle in COMM-HANDLE that can be used to receive the reply of the request sent.

- Errors

Under the following conditions, TPACALL fails and sets TP-STATUS to one of the values below. Unless otherwise noted, failure does not affect the caller's transaction, if one exists.

TPEINVAL

Invalid arguments were given (for example, settings in TPSVCDEF-REC are invalid).

TPENOENT

Cannot send to SERVICE-NAME because it does not exist.

TPEITYPE

The pair REC-TYPE and SUB-TYPE is not one of the allowed types and sub-types that SERVICE-NAME accepts.

TPELIMIT

The caller's request was not sent because the maximum number of outstanding asynchronous requests has been reached.

TPETRAN

SERVICE-NAME does not support transactions and TPTRAN was set.

TPETIME

A timeout occurred. If the caller is in transaction mode, a transaction timeout occurred and the transaction is marked rollback-only; otherwise, a blocking timeout occurred and both TPBLOCK and TPTIME were specified. If a transaction timeout occurred, any attempts to send new requests or receive outstanding replies fail with `TPETIME` until the transaction has been rolled back.

`TPEBLOCK`

A blocking condition exists and `TPNOBLOCK` was specified.

`TPEGOTSIG`

A signal was received and `TPNOSIGRSTRT` was specified.

`TPEPROTO`

`TPACALL` was called in an improper context.

`TPESYSTEM`

A communication resource manager system error has occurred. The exact nature of the error is determined in a product-specific manner.

`TPEOS`

An operating system error has occurred. The exact nature of the error is determined in a product-specific manner.

See also

`TPCALL`, `TPCANCEL`, `TPGETRPLY`

<<Notes on use with OpenTP1>>

1. <<The value `TPNOBLOCK` is invalid under the relevant version of the OpenTP1. Therefore, the error code `TPEBLOCK` will not be returned to `TP-STATUS`. The OpenTP1 is designed so that if communication is impossible because of blocking, `TPESYSTEM` is returned as when communication is impossible because of network failure.>>
2. <<The value `TPNOTIME` is invalid under the relevant version of the OpenTP1.>>
3. <<The value `TPSIGRSTRT` is invalid. Regardless of whether this value is set, when a signal is received, the interrupted system call is reinvoked. `TPEGOTSIG` will never return.>>
4. <<Under the relevant version of the OpenTP1, `TPEITYPE` will not return. If a record of a type unavailable with `SERVICE-NAME` is passed, the function `TPACALL` normally returns, but `TPESYSTEM` or `TPESVCERR` will return when the function `TPGETRPLY` is called. Therefore, the error condition is identified. If the calling program is in transaction mode, the rollback_only state comes into effect.>>
5. <<Under the OpenTP1, when a process encounters transaction timeout, it terminates abnormally. Therefore, `TPETIME` returns only when blocking timeout occurs.>>

6. <<Under the relevant version of the OpenTP1, a record which requires rollback causes the return of `TPESYSTEM` unless otherwise specified by the X/Open. However, the `rollback_only` state may not come into effect even when `TPESYSTEM` returns.>>
7. <<Under the relevant version of the OpenTP1, `TPELIMIT` will not return.>>
8. <<For OSI TP communication using the TP1/NET/OSI-TP-Extended, the length of the send data must not exceed the value assigned to the `length` operand of the NET buffer group definition (`nettbuf`) contained in the NET/Library common definition.>>

TPADVERTISE - Advertise a service name

Format

```
01  SERVICE-NAME  PIC  X(15).
01  PROGRAM-NAME  PIC  X(32).
01  TPSTATUS-REC.
    COPY  TPSTATUS.

CALL "TPADVERTISE" USING SERVICE-NAME PROGRAM-NAME
                         TPSTATUS-REC.
```

Description

`TPADVERTISE` allows a server to advertise the services that it offers. By default, a server's services are advertised when it is booted and unadvertised when it is shutdown.

`TPADVERTISE` advertises `SERVICE-NAME` for the server. `SERVICE-NAME` should be 15 characters or fewer, but cannot be `SPACES`. Longer names are accepted and truncated to 15 characters. Users should make sure that truncated names do not match other service names. `PROGRAM-NAME` is the name of a service program. This program is invoked whenever a request for `SERVICE-NAME` is received by the server. `PROGRAM-NAME` cannot be `SPACES`.

If `SERVICE-NAME` is already advertised for the server and `PROGRAM-NAME` matches its current program, `TPADVERTISE` returns success (this includes truncated names that match already advertised names). However, if `SERVICE-NAME` is already advertised for the server but `PROGRAM-NAME` does not match its current program, an error is returned (this can happen if truncated names match already advertised names).

<<Data areas>>

- <<`SERVICE-NAME`

  Specify the name of the service to be advertised.>>

- <<`PROGRAM-NAME`

  Specify the name of the service program.>>

- <<`TPSTATUS-REC`

  Will be assigned the return value indicating the result of `TPADVERTISE` execution.>>

Return value

Upon successful completion, `TPADVERTISE` sets `TP-STATUS` to `TPOK`.

■ Errors

Under the following conditions, `TPADVERTISE` fails and sets `TP-STATUS` to one of the following values:

`TPEINVAL`

Either `SERVICE-NAME` or `PROGRAM-NAME` is `SPACES`, or `PROGRAM-NAME` is not the name of a valid program.

`TPELIMIT`

`SERVICE-NAME` cannot be advertised because of space limitations.

`TPEMATCH`

`SERVICE-NAME` is already advertised for the server but with a program other than `PROGRAM-NAME`. Although `TPADVERTISE` fails, `SERVICE-NAME` remains advertised with its current program (that is, `PROGRAM-NAME` does not replace the current program).

`TPEPROTO`

`TPADVERTISE` was called in an improper context.

`TPESYSTEM`

A communication resource manager system error has occurred. The exact nature of the error is determined in a product-specific manner.

`TPEOS`

An operating system error has occurred. The exact nature of the error is determined in a product-specific manner.

See also

`TPSVCSTART`, `TPUNADVERTISE`.

<<Notes on use with OpenTP1>>

1. <<The function `TPADVERTISE` can be called only by SPPs. When the server starts, all services specified in the user service definition are automatically advertised. Combinations of service names and programs can be advertised only when they are specified in the user service definition of this function.>>
2. <<Under the OpenTP1, if the service group of UAPs which invoke the function `TPADVERTISE` is the same as the service group of UAPs which have advertised the services, this function returns normally on the assumption that advertisement is completed. If the two groups do not match, the function returns with an error.>>
3. <<Under the OpenTP1, the maximum length of `PROGRAM-NAME` is 20 characters.>>

TPCALL - Send a service request and synchronously await its reply

Format

```
01  TPSVCDEF-REC.
    COPY  TPSVCDEF.
01  ITPTYPE-REC.
    COPY  TPTYPE.
01  IDATA-REC.
    COPY  Data record definition.
01  OTPTYPE-REC.
    COPY TPTYPE.
01  ODATA-REC.
    COPY  Data record definition.
01  TPSTATUS-REC.
    COPY  TPSTATUS.

CALL  "TPCALL"  USING  TPSVCDEF-REC  ITPTYPE-REC  IDATA-REC
                       OTPTYPE-REC ODATA-REC TPSTATUS-REC.
```

Description

`TPCALL` sends a request and synchronously awaits its reply. A call to this routine is the same as calling `TPACALL` immediately followed by `TPGETRPLY`. `TPCALL` sends a request to the service named by `SERVICE-NAME`. The data portion of a request is specified by `IDATA-REC` and `LEN` in `ITPTYPE-REC` specifies how much of `IDATA-REC` to send. Note that if `ITYPE-REC` is a record of a type that does not require a length to be specified, `LEN` in `ITPTYPE-REC` is ignored (and may be 0). If `IDATA-REC` is a record of a type that does require a length, `LEN` in `ITPTYPE-REC` must not be zero. If `REC-TYPE` in `ITPTYPE-REC` does not have a subtype, `SUB-TYPE` in `ITPTYPE-REC` is ignored (and may be `SPACES`). If `REC-TYPE` in `ITPTYPE-REC` is `SPACES`, `IDATA-REC` and `LEN` in `ITPTYPE-REC` are ignored and a request is sent with no data portion. `REC-TYPE` in `ITPTYPE-REC` and `SUB-TYPE` in `ITPTYPE-REC` must match one of the types and sub-types recognized by `SERVICE-NAME`.

`ODATA-REC` specifies where the reply is read into, and, on input, `LEN` in `OTPTYPE-REC` indicates the maximum number of bytes that should be moved into `ODATA-REC`. If the same record is to be used for both sending and receiving, redefine `ODATA-REC` (`REDEFINES`) to `IDATA-REC`. Upon successful return from `TPCALL`, `LEN` in `OTPTYPE-REC` contains the actual number of bytes moved into `ODATA-REC`. `REC-TYPE` in `OTPTYPE-REC` and `SUB-TYPE` in `OTPTYPE-REC` contain the reply's type and sub-type, respectively. If the reply is larger than `ODATA-REC`, `ODATA-REC` contains only as many bytes as fit in the record. The remainder of the reply is discarded and `TPCALL` sets `TPTRUNCATE`.

If `LEN` in `OTPTYPE-REC` is 0 upon successful return, the reply has no data portion and `ODATA-REC` was not modified. It is an error for `LEN` in `OTPTYPE-REC` to be 0 on input.

<<Data areas>>

- <<`TPSVCDEF-REC`

  Specify a value indicating the `TPCALL` operation. The specifiable values and their meanings will be explained later.>>

- <<`ITPTYPE-REC`

  Indicates the record type and subtype record name of the send data.>>

- <<`IDATA-REC`

  Points to the send data.>>

- <<`OTPTYPE-REC`

  Indicates the record type and subtype record name of the receive data.>>

- <<`ODATA-REC`

  Points to the receive data.>>

- <<`TPSTATUS-REC`

  Will be assigned the return value indicating the result of `TPCALL` execution.>>

  The valid settings of `TPSVCDEF-REC` are as follows:

  `TPNOTRAN`

  If the caller is in transaction mode and this setting is used, when `SERVICE-NAME` is invoked, it is not performed on behalf of the caller's transaction. If `SERVICE-NAME` does not support transactions, this setting must be used when the caller is in transaction mode. A caller in transaction mode that uses this setting is still subject to the transaction timeout (and no other). If a service fails that was invoked with this setting, the caller's transaction is not affected. Either `TPNOTRAN` or `TPTRAN` must be set.

  `TPTRAN`

  If the caller is in transaction mode and this setting is used, when `SERVICE-NAME` is invoked, it is performed on behalf of the caller's transaction. This setting is ignored if the caller is not in transaction mode. Either `TPNOTRAN` or `TPTRAN` must be set.

  `TPNOCHANGE`

  When this setting is used, the type of `ODATA-REC` is not allowed to change. That is, the type and sub-type of the reply record must match `REC-TYPE` in `OTPTYPE-REC` and `SUB-TYPE` in `OTPTYPE-REC`, respectively. Either `TPNOCHANGE` or `TPCHANGE` must be set.

  `TPCHANGE`

The type and/or subtype of the reply record are allowed to differ from those specified in REC-TYPE in OTPTYPE-REC and SUB-TYPE in OTPTYPE-REC, respectively, so long as the receiver recognizes the incoming record type. Either TPNOCHANGE or TPCHANGE must be set.

TPNOBLOCK

The request is not sent if a blocking condition exists (for example, the internal buffers into which the message is transferred are full). Note that this setting applies only to the send portion of TPCALL: the routine may block waiting for the reply. Either TPNOBLOCK or TPBLOCK must be set.

TPBLOCK

When TPBLOCK is specified and a blocking condition exists, the caller blocks until the condition subsides or a timeout occurs (either transaction or blocking timeout). Either TPNOBLOCK or TPBLOCK must be set.

TPNOTIME

This setting signifies that the caller is willing to block indefinitely and wants to be immune to blocking timeouts. Transaction timeouts may still occur. Either TPNOTIME or TPTIME must be set.

TPTIME

This setting signifies that the caller receives blocking timeouts if a blocking condition exists and the blocking time is reached. Either TPNOTIME or TPTIME must be set.

TPSIGRSTRT

If a signal interrupts any underlying system calls, the interrupted system call is re-issued. Either TPNOSIGRSTRT or TPSIGRSTRT must be set.

TPNOSIGRSTRT

If a signal interrupts any underlying system calls, the interrupted system call is not restarted and the routine fails. Either TPNOSIGRSTRT or TPSIGRSTRT must be set.

Return value

Upon successful completion, TPCALL sets TP-STATUS to TPOK. When TP-STATUS is set to either TPOK or TPESVCFAIL, APPL-RETURN-CODE contains an application-defined value that was sent as part of TPRETURN. If the size of the incoming message is larger than the size specified in LEN in OTPTYPE-REC on input, then TPTRUNCATE is set in OTPTYPE-REC and only LEN in OTPTYPE-REC bytes are moved into ODATA-REC. The remaining bytes are discarded.

- Errors

Under the following conditions, `TPCALL` fails and sets `TP-STATUS` to one of the values below. Unless otherwise noted, failure does not affect the caller's transaction, if one exists.

`TPEINVAL`

Invalid arguments were given (for example, settings in `TPSVCDEF-REC` are invalid).

`TPENOENT`

Cannot send to `SERVICE-NAME` because it does not exist.

`TPEITYPE`

The pair `REC-TYPE` in `ITPTYPE-REC` and `SUB-TYPE` in `ITPTYPE-REC` is not one of the allowed types and sub-types that `SERVICE-NAME` accepts.

`TPEOTYPE`

Either the type and sub-type of the reply are not known to the caller, or `TPNOCHANGE` was set and `REC-TYPE` in `OTPTYPE-REC` and `SUB-TYPE` in `OTPTYPE-REC` do not match the type and sub-type of the reply sent by the service. Neither `ODATA-REC` nor `OTPTYPE-REC` are changed. If the service request was made on behalf of the caller's current transaction, the transaction is marked rollback-only since the reply is discarded.

`TPETRAN`

`SERVICE-NAME` does not support transactions and `TPTRAN` was set.

`TPETIME`

A timeout occurred. If the caller is in transaction mode, a transaction timeout occurred and the transaction is marked rollback-only; otherwise, a blocking timeout occurred and both `TPBLOCK` and `TPTIME` were specified. In either case, neither `ODATA-REC` nor `OTPTYPE-REC` are changed. If a transaction timeout occurred, any attempts to send new requests or receive outstanding replies fail with `TPETIME` until the transaction has been rolled back.

`TPESVCFAIL`

The service routine sending the caller's reply called `TPRETURN` with `TPFAIL`. This is an application-level failure. The contents of the service's reply, if one was sent, are available in `ODATA-REC`. If the service request was made on behalf of the caller's current transaction, the transaction is marked rollback-only. Note that so long as the transaction has not timed out, further communication may be attempted before rolling back the transaction. Such attempts may be processed normally or may fail (producing an error return to event). Such attempts should be made with `TPNOTRAN` set if they are to have any lasting effect. Any work performed on behalf of the caller's transaction is rolled back upon transaction completion.

`TPESVCERR`

An error was encountered either in invoking a service routine or during its completion in `TPRETURN` (for example, bad arguments were passed). No reply data is returned when this error occurs (that is, neither `ODATA-REC` nor `OTPTYPE-REC` are changed). If the service request was made on behalf of the caller's transaction, the transaction is marked rollback-only. Note that so long as the transaction has not timed out, further communication may be attempted before rolling back the transaction. Such attempts may be processed normally or may fail (producing an error return or event). Such attempts should be made with `TPNOTRAN` set if they are to have any lasting effect. Any work performed on behalf of the caller's transaction is rolled back upon transaction completion.

`TPEBLOCK`

A blocking condition was found on the send portion of `TPCALL` and `TPNOBLOCK` was specified.

`TPEGOTSIG`

A signal was received and `TPNOSIGRSTRT` was specified.

`TPEPROTO`

`TPCALL` was called in an improper context.

`TPESYSTEM`

A communication resource manager system error has occurred. The exact nature of the error is determined in a product-specific manner.

`TPEOS`

An operating system error has occurred. The exact nature of the error is determined in a product-specific manner.

See also

`TPACALL`, `TPGETRPLY`, `TPRETURN`.

<<Notes on use with OpenTP1>>

1. <<The value `TPNOBLOCK` is invalid under the relevant version of the OpenTP1. Therefore, the error code `TPEBLOCK` will not be returned to `TP-STATUS`. The OpenTP1 is designed so that if communication is impossible because of blocking, `TPESYSTEM` is returned as when communication is impossible because of network failure.>>
2. <<Under the relevant version of the OpenTP1, the value `TPNOTIME` is valid only

when a response is received. It is invalid when blocking occurs at the time of service request transmission.>>

3. <<The value `TPSIGRSTRT` is invalid. Regardless of whether this value is set, when a signal is received, the interrupted system call is reinvoked. `TPEGOTSIG` will never return.>>
4. <<Under the relevant version of the OpenTP1, `TPEITYPE` will not return. If a record of a type unavailable with `SERVICE-NAME` is passed, `TPESYSTEM` will return. If the calling program is in transaction mode, the `rollback_only` state comes into effect.>>
5. <<Under the OpenTP1, when a process encounters transaction timeout, it terminates abnormally. Therefore, `TPETIME` returns only when blocking timeout occurs.>>
6. <<Under the relevant version of the OpenTP1, an error which raises need for rollback causes the return of `TPESYSTEM` unless otherwise specified by the X/Open. However, the `rollback_only` state may not come into effect even when `TPESYSTEM` returns.>>
7. <<If the SPP that was asked to offer its service abnormally terminates, the function could return with a `TPETIME` error before the time assigned to `watch_time` in the definition elapses. If `watch_time` is assigned 0 (infinitely wait for a response), the function could return with a `TPEPROTO` error.>>
8. <<If the service request is not validated when OpenTP1 security is in use, the function returns with a `TPEPROTO` error. To determine whether the error return was caused by an invalidated service request, refer to the detailed information in the UAP trace.>>
9. <<If a line error occurs during OSI TP communication using the TP1/NET/OSI-TP-Extended, the function returns with a `TPESVCERR` error.>>
10. <<For OSI TP communication using the TP1/NET/OSI-TP-Extended, the length of the send or receive data must not exceed the value assigned to the `length` operand of the `NET` buffer group definition (`nettbuf`) contained in the NET/Library common definition.>>

TPCANCEL - Cancel a communication handle for an outstanding reply

Format

```
01  TPSVCDEF-REC.
    COPY  TPSVCDEF.
01  TPSTATUS-REC.
    COPY  TPSTATUS.

CALL "TPCANCEL" USING TPSVCDEF-REC TPSTATUS-REC.
```

Description

`TPCANCEL` cancels a communication handle, `COMM-HANDLE`, returned by `TPACALL`. It is an error to attempt to cancel a communication handle associated with a global transaction.

Upon success, `COMM-HANDLE` is no longer valid and any reply received (by the communication resource manager) on behalf of `COMM-HANDLE` is silently discarded.

<<Data areas>>

- <<`COMM-HANDLE`

  Specify the communication handle to be canceled.>>

- <<`TPSTATUS-REC`

  Will be assigned the return value indicating the result of `TPCANCEL` execution.>>

Return value

Upon successful completion, `TPCANCEL` sets `TP-STATUS` to `TPOK`.

- Errors

  Under the following conditions, `TPCANCEL` fails and sets `TP-STATUS` to one of the following values:

  `TPEBADDESC`

  > `COMM-HANDLE` contains an invalid communication handle.

  `TPETRAN`

  > `COMM-HANDLE` is associated with the caller's global transaction. `COMM-HANDLE` remains valid and the caller's current transaction is not affected.

  `TPEPROTO`

  > `TPCANCEL` was called in an improper context.

`TPESYSTEM`

A communication resource manager system error has occurred. The exact nature of the error is determined in a product-specific manner.

`TPEOS`

An operating system error has occurred. The exact nature of the error is determined in a product-specific manner.

See also

`TPACALL`.

TPCONNECT - Establish a conversational service connection

Format

```
01  TPSVCDEF-REC.
    COPY  TPSVCDEF.
01  TPTYPE-REC.
    COPY  TPTYPE.
01  DATA-REC.
    COPY  Data record definition.
01  TPSTATUS-REC.
    COPY  TPSTATUS.

CALL "TPCONNECT" USING TPSVCDEF-REC TPTYPE-REC DATA-REC
                       TPSTATUS-REC.
```

Description

`TPCONNECT` allows a program to set up a half-duplex connection to a conversational service, `SERVICE-NAME`.

As part of setting up a connection, the caller can pass application-defined data to the receiving service routine. If the caller chooses to pass data, `DATA-REC` contains the data and `LEN` specifies how much of the record to send. Note that if `DATA-REC` is a record of a type that does not require a length to be specified, `LEN` is ignored (and may be 0). If `DATA-REC` is a record of a type that does require a length, `LEN` must not be zero. If `REC-TYPE` does not have a subtype, `SUB-TYPE` is ignored (and may be `SPACES`). If `REC-TYPE` is `SPACES`, `DATA-REC` and `LEN` are ignored (no application data is passed to the conversational service). `REC-TYPE` and `SUB-TYPE` must match one of the types and sub-types recognized by `SERVICE-NAME`.

Because the conversational service receives `DATA-REC` and `LEN` upon successful return from `TPSVCSTART`, the service does not call `TPRECV` to get the data sent by `TPCONNECT`.

<<Data areas>>

- <<`TPSVCDEF-REC`

  Specify a value indicating the `TPCONNECT` operation. The value specified here determines the return value. The specifiable values and their meanings will be explained later.>>

- <<`TPTYPE-REC`

  Indicates the data type and subtype record name of the send data.>>

- <<`DATA-REC`

  Points to the send data.>>

- `<<TPSTATUS-REC`

  Will be assigned the return value indicating the result of `TPCONNECT` execution.>>

  The valid settings of `TPSVCDEF-REC` are as follows:

  `TPNOTRAN`

  > If the caller is in transaction mode and this setting is used, when `SERVICE-NAME` is invoked, it is not performed on behalf of the caller's transaction. If `SERVICE-NAME` does not support transactions, this setting must be used when the caller is in transaction mode. A caller in transaction mode that uses this setting is still subject to the transaction timeout (and no other). If a service fails that was invoked with this setting, the caller's transaction is not affected. Either `TPNOTRAN` or `TPTRAN` must be set.

  `TPTRAN`

  > If the caller is in transaction mode and this setting is used, when `SERVICE-NAME` is invoked, it is performed on behalf of the caller's transaction. This setting is ignored if the caller is not in transaction mode. Either `TPNOTRAN` or `TPTRAN` must be set.

  `TPSENDONLY`

  > The caller wants the connection to be set up initially such that it can send data and the called service can only receive data (that is, the caller initially has control of the connection). Either `TPSENDONLY` or `TPRECVONLY` must be specified.

  `TPRECVONLY`

  > The caller wants the connection to be set up initially such that it can only receive data and the called service can send data (that is, the service being called initially has control of the connection). Either `TPSENDONLY` or `TPRECVONLY` must be specified.

  `TPNOBLOCK`

  > The connection is not established and the data is not sent if a blocking condition exists (for example, the internal buffers into which the message is transferred are full). Either `TPNOBLOCK` or `TPBLOCK` must be set.

  `TPBLOCK`

  > When `TPBLOCK` is specified and a blocking condition exists, the caller blocks until the condition subsides or a timeout occurs (either transaction or blocking timeout). Either `TPNOBLOCK` or `TPBLOCK` must be set.

  `TPNOTIME`

  > This setting signifies that the caller is willing to block indefinitely and wants to be immune to blocking timeouts. Transaction timeouts may still occur. Either

TPNOTIME or TPTIME must be set.

TPTIME

This setting signifies that the caller receives blocking timeouts if a blocking condition exists and the blocking time is reached. Either TPNOTIME or TPTIME must be set.

TPSIGRSTRT

If a signal interrupts any underlying system calls, the interrupted system call is re-issued. Either TPNOSIGRSTRT or TPSIGRSTRT must be set.

TPNOSIGRSTRT

If a signal interrupts any underlying system calls, the interrupted system call is not restarted and the routine fails. Either TPNOSIGRSTRT or TPSIGRSTRT must be set.

Return value

Upon successful completion, TPCONNECT sets TP-STATUS to TPOK and returns a valid communication handle in COMM-HANDLE that is used to refer to the connection in subsequent calls.

- Errors

Under the following conditions, TPCONNECT fails and sets TP-STATUS to one of the values below. Unless otherwise noted, failure does not affect the caller's transaction, if one exists.

TPEINVAL

Invalid arguments were given (for example, settings in TPSVCDEF-REC are invalid).

TPENOENT

Cannot initiate a connection to SERVICE-NAME because it does not exist.

TPEITYPE

The pair REC-TYPE and SUB-TYPE is not one of the allowed types and sub-types that SERVICE-NAME accepts.

TPELIMIT

The connection was not established because the maximum number of outstanding connections has been reached.

TPETRAN

SERVICE-NAME does not support transactions and TPTRAN was set.

TPETIME

A timeout occurred. If the caller is in transaction mode, a transaction timeout occurred and the transaction is marked rollback-only; otherwise, a blocking timeout occurred and both `TPBLOCK` and `TPTIME` were specified. If a transaction timeout occurred, any attempts to send or receive messages on any connections or to start a new connection fail with `TPETIME` until the transaction has been rolled back.

`TPEBLOCK`

A blocking condition exists and `TPNOBLOCK` was specified.

`TPEGOTSIG`

A signal was received and `TPNOSIGRSTRT` was specified.

`TPEPROTO`

`TPCONNECT` was called in an improper context.

`TPESYSTEM`

A communication resource manager system error has occurred. The exact nature of the error is determined in a product-specific manner.

`TPEOS`

An operating system error has occurred. The exact nature of the error is determined in a product-specific manner.

See also

`TPDISCON`, `TPRECV`, `TPSEND`, `TPSVCSTART`.

<<Notes on use with OpenTP1>>

1. <<The value `TPNOBLOCK` is invalid under the relevant version of the OpenTP1. Therefore, the error code `TPEBLOCK` will not be returned to `TP-STATUS`. The OpenTP1 is designed so that if communication is impossible because of blocking, `TPESYSTEM` is returned as when communication is impossible because of network failure.>>
2. <<The value `TPNOTIME` is invalid under the relevant version of the OpenTP1.>>
3. <<The value `TPSIGRSTRT` is invalid. Regardless of whether this value is set, when a signal is received, the interrupted system call is reinvoked. `TPEGOTSIG` will never return.>>
4. <<Under the relevant version of the OpenTP1, `TPEITYPE` will not return. If a record of a type unavailable with `SERVICE-NAME` is passed, `TPESYSTEM` or `TPESVCERR` will return. If the calling program is in transaction mode, the `rollback_only` state comes into effect.>>
5. <<Under the OpenTP1, when a process encounters transaction timeout, it

terminates abnormally. Therefore, `TPETIME` returns only when blocking timeout occurs.>>

6. <<Under the relevant version of the OpenTP1, an error which raises need for rollback causes the return of `TPESYSTEM` unless otherwise specified by the X/Open. However, the `rollback_only` state may not come into effect even when `TPESYSTEM` returns.>>
7. <<If the function returns with a `TPEPROTO` error. To determine whether the error return was caused by an invalidated service request, refer to the detailed information in the UAP trace.>>
8. <<When OSI TP communication using the TP1/NET/OSI-TP-Extended is in progress, it cannot be used for conversational services. If an attempt is made to do so, the system operation is unpredictable.>>
9. <<If the server AP is in shutdown status, the system operates as follows depending on whether the request destination SPP that is shutdown is on a local node or on a remote node:

   When the request destination SPP on a local node is shutdown:

   > `tpconnect()` returns `-1` and sets the value `TPEPROTO` in `tperrno`.

   When the request destination SPP on a remote node is shutdown:

   > In the transaction mode, the server AP terminates abnormally due to transaction time-out.
   >
   > In the non-transaction mode, `tpconnect()` returns `-1` and sets the value `TPETIME` in `tperrno`.>>

TPDISCON - Terminate a conversational service connection abortively

Format

```
01  TPSVCDEF-REC.
    COPY  TPSVCDEF.
01  TPSTATUS-REC.
    COPY  TPSTATUS.

CALL "TPDISCON" USING TPSVCDEF-REC  TPSTATUS-REC.
```

Description

TPDISCON immediately terminates the connection specified by COMM-HANDLE and generates a TPEV-DISCONIMM event on the other end of the connection.

TPDISCON can be called only by the originator of the conversation. TPDISCON cannot be called within a conversational service on the communication handle with which it was invoked. Rather, a conversational service must use TPRETURN to signify that it has completed its part of the conversation. Similarly, even though a program communicating with a conversational service can issue TPDISCON, the preferred way is to let the service terminate the connection in TPRETURN; doing so ensures correct results.

TPDISCON causes the connection to be terminated immediately (that is, abortively rather than orderly). Any data that has not yet reached its destination may be lost. TPDISCON can be issued even when the program on the other end of the connection is participating in the caller's transaction. In this case, the transaction must be rolled back. Also, the caller does not need to have control of the connection when TPDISCON is called.

<<Data areas>>

- <<TPSVCDEF-REC

  Specify the communication handle of the connection indicated by COMM-HANDLE to be aborted.>>

- <<TPSTATUS-REC

  Will be assigned the return value indicating the result of TPDISCON execution.>>

Return value

Upon successful completion, TPDISCON sets TP-STATUS to TPOK.

- Errors

  Under the following conditions, TPDISCON fails and sets TP-STATUS to one of the following values:

`TPEBADDESC`

Either `COMM-HANDLE` is invalid or it is the communication handle with which a conversational service was invoked.

`TPETIME`

A timeout occurred. The communication handle is no longer valid.

`TPEPROTO`

`TPDISCON` was called in an improper context.

`TPESYSTEM`

A communication resource manager system error has occurred. The exact nature of the error is determined in a product-specific manner.

`TPEOS`

An operating system error has occurred. The exact nature of the error is determined in a product-specific manner.

See also

`TPCONNECT`, `TPRECV`, `TPRETURN`, `TPSEND`.

<<Notes on use with OpenTP1>>

1. <<The error code `TPETIME` will not be returned to `TP-STATUS` under the relevant version of the OpenTP1.>>
2. <<When OSI TP communication using the TP1/NET/OSI-TP-Extended is in progress, it cannot be used for conversational services. If an attempt is made to do so, the system operation is unpredictable.>>

TPGETRPLY - Get a reply from a previous service request

Format

```
01  TPSVCDEF-REC.
    COPY  TPSVCDEF.
01  TPTYPE-REC.
    COPY  TPTYPE.
01  DATA-REC.
    COPY  Data record definition.
01  TPSTATUS-REC.
    COPY  TPSTATUS.

CALL "TPGETRPLY" USING TPSVCDEF-REC TPTYPE-REC  DATA-REC
                       TPSTATUS-REC.
```

Description

`TPGETRPLY` returns a reply from a previously sent request. `TPGETRPLY` either returns a reply for a particular request, or it returns any reply that is available. Both options are described below.

`DATA-REC` specifies where the reply is read into, and, on input, `LEN` indicates the maximum number of bytes that should be moved into `DATA-REC`. Upon successful return from `TPGETRPLY`, `LEN` contains the actual number of bytes moved into `DATA-REC`. `REC-TYPE` and `SUB-TYPE` contain the data's type and sub-type, respectively. If the reply is larger than `DATA-REC`, `DATA-REC` contain only as many bytes as fit in the record. The remainder of the reply is discarded and `TPGETRPLY` sets `TPTRUNCATE`.

If `LEN` is 0 upon successful return, the reply has no data portion and `DATA-REC` was not modified. It is an error for `LEN` to be 0 on input.

<<Data areas>>

- <<`TPSVCDEF-REC`

  Specify a value indicating the `TPGETRPLY` operation and a communication handle. The specifiable values and their meanings will be explained later.>>

- <<`TPTYPE-REC`

  Indicates the data type and subtype record name of the data to be received.>>

- <<`DATA-REC`

  Points to the data to be received.>>

- <<`TPSTATUS-REC`

  Will be assigned the return value indicating the result of `TPGETRPLY` execution.>>

The valid settings of `TPSVCDEF-REC` are as follows:

`TPGETANY`

> This setting signifies that `TPGETRPLY` should ignore the communication handle indicated by `COMM-HANDLE` on input, return any reply available, and set `COMM-HANDLE` on output to the communication handle for the reply returned. If no replies exist, `TPGETRPLY` can optionally wait for one to arrive. Either `TPGETANY` or `TPGETHANDLE` must be set.

`TPGETHANDLE`

> This setting signifies that `TPGETRPLY` should use the communication handle identified by `COMM-HANDLE` on input and return a reply available for that handle only. If no replies exist, `TPGETRPLY` can optionally wait for one to arrive. Either `TPGETANY` or `TPGETHANDLE` must be set.

`TPNOCHANGE`

> When this setting is used, the type of `DATA-REC` is not allowed to change. That is, the type and sub-type of the reply record must match `REC-TYPE` and `SUB-TYPE`, respectively. Either `TPNOCHANGE` or `TPCHANGE` must be set.

`TPCHANGE`

> The type and/or subtype of the reply record are allowed to differ from those specified in `REC-TYPE` and `SUB-TYPE`, respectively, so long as the receiver recognizes the incoming record type. Either `TPNOCHANGE` or `TPCHANGE` must be set.

`TPNOBLOCK`

> `TPGETRPLY` does not wait for the reply to arrive. If a reply is available, `TPGETRPLY` gets the reply and returns. Either `TPNOBLOCK` or `TPBLOCK` must be set.

`TPBLOCK`

> When `TPBLOCK` is specified and no reply is available, the caller blocks until the reply arrives or a timeout occurs (either transaction or blocking timeout). Either `TPNOBLOCK` or `TPBLOCK` must be set.

`TPNOTIME`

> This setting signifies that the caller is willing to block indefinitely for its reply and wants to be immune to blocking timeouts. Transaction timeouts may still occur. Either `TPNOTIME` or `TPTIME` must be set.

`TPTIME`

> This setting signifies that the caller receives blocking timeouts if a blocking condition exists and the blocking time is reached. Either `TPNOTIME` or `TPTIME`

must be set.

`TPSIGRSTRT`

If a signal interrupts any underlying system calls, the interrupted system call is re-issued. Either `TPNOSIGRSTRT` or `TPSIGRSTRT` must be set.

`TPNOSIGRSTRT`

If a signal interrupts any underlying system calls, the interrupted system call is not restarted and the routine fails. Either `TPNOSIGRSTRT` or `TPSIGRSTRT` must be set.

Except as noted below, `COMM-HANDLE` is no longer valid after its reply is received.

Return value

Upon successful completion, `TPGETRPLY` sets `TP-STATUS` to `TPOK`. When `TP-STATUS` is set to either `TPOK` or `TPESVCFAIL`, `APPL-RETURN-CODE` contains an application-defined value that was sent as part of `TPRETURN`. If the size of the incoming message is larger than the size specified in `LEN` on input, `TPTRUNCATE` is set and only `LEN` bytes are moved into `DATA-REC`. The remaining bytes are discarded.

- Errors

Under the following conditions, `TPGETRPLY` fails and sets `TP-STATUS` as indicated below. Note that if `TPGETHANDLE` is set, `COMM-HANDLE` is invalidated unless otherwise stated. If `TPGETANY` is set, `COMM-HANDLE` identifies the descriptor for the reply on which the failure occurred; if an error occurred before a reply could be retrieved, `COMM-HANDLE` is set to 0, unless otherwise stated. Also, the failure does not affect the caller's transaction, if one exists, unless otherwise stated.

`TPEINVAL`

Invalid arguments were given (for example, settings in `TPSVCDEF-REC` are invalid).

`TPEBADDESC`

`COMM-HANDLE` contains an invalid communication handle.

`TPEOTYPE`

Either the type and sub-type of the reply are not known to the caller, or `TPNOCHANGE` was set and `REC-TYPE` and `SUB-TYPE` do not match the type and sub-type of the reply sent by the service. Neither `DATA-REC` nor `TPTYPE-REC` are changed. If the reply was to be received on behalf of the caller's current transaction, the transaction is marked rollback-only since the reply is discarded.

`TPETIME`

A timeout occurred. If the caller is in transaction mode, a transaction timeout occurred and the transaction is marked rollback-only; otherwise, a blocking timeout occurred and both TPBLOCK and TPTIME were specified. In either case, neither DATA-REC nor TPTYPE-REC are changed. If TPGETHANDLE was set, COMM-HANDLE remains valid unless the caller is in transaction mode. If a transaction timeout occurred, any attempts to send new requests or receive outstanding replies fail with TPETIME until the transaction has been rolled back.

TPESVCFAIL

The service routine sending the caller's reply called TPRETURN with TPFAIL. This is an application-level failure. The contents of the service's reply, if one was sent, are available in DATA-REC. If the reply was received on behalf of the caller's transaction, the transaction is marked rollback-only. Note that so long as the transaction has not timed out, further communication may be attempted before rolling back the transaction. Such attempts may be processed normally or may fail (producing an error return or event). Such attempts should be made with TPNOTRAN set if they are to have any lasting effect. Any work performed on behalf of the caller's transaction is rolled back upon transaction completion.

TPESVCERR

An error was encountered either in invoking a service routine or during its completion in TPRETURN (for example, bad arguments were passed). No reply data is returned when this error occurs (that is, neither DATA-REC nor TPTYPE-REC are changed). If the reply was received on behalf of the caller's transaction, the transaction is marked rollback-only. Note that so long as the transaction has not timed out, further communication may be attempted before rolling back the transaction. Such attempts may be processed normally or may fail (producing an error return or event). Such attempts should be made with TPNOTRAN set if they are to have any lasting effect. Any work performed on behalf of the caller's transaction is rolled back upon transaction completion.

TPEBLOCK

A blocking condition exists and TPNOBLOCK was specified. COMM-HANDLE remains valid.

TPEGOTSIG

A signal was received and TPNOSIGRSTRT was specified.

TPEPROTO

TPGETRPLY was called in an improper context.

TPESYSTEM

A communication resource manager system error has occurred. The exact nature of the error is determined in a product-specific manner.

TPEOS

An operating system error has occurred. The exact nature of the error is determined in a product-specific manner.

See also

`TPACALL`, `TPCANCEL`, `TPRETURN`.

<<Notes on use with OpenTP1>>

1. <<The value `TPSIGRSTRT` is invalid. Regardless of whether this value is set, when a signal is received, the interrupted system call is reinvoked. `TPEGOTSIG` will never return.>>
2. <<Under the OpenTP1, when a process encounters transaction timeout, it terminates abnormally. Therefore, `TPETIME` returns only when blocking timeout occurs.>>
3. <<Under the relevant version of the OpenTP1, data which requires rollback causes the return of `TPESYSTEM` unless otherwise specified by the X/Open. However, the `rollback_only` state may not come into effect even when `TPESYSTEM` returns.>>
4. <<If the function `TPACALL` passes a record of a type that cannot be used by the called service, it returns normally, but the function `TPGETRPLY` will encounter an error. If the function `TPGETRPLY` encounters a `TPESYSTEM` or `TPESVCERR` error, check the results of the function `TPACALL` as well.>>
5. <<If the SPP that was asked to offer its service abnormally terminates, the function could return with a `TPETIME` error before the time assigned to `watch_time` in the definition elapses. If `watch_time` is assigned `0` (infinitely wait for a response), the function could return with a `TPEPROTO` error.>>
6. <<If the service request is not validated when OpenTP1 security is in use, the function returns with a `TPEPROTO` error. To determine whether the error return was caused by an invalidated service request, refer to the detailed information in the UAP trace.>>
7. <<For OSI TP communication using the TP1/NET/OSI-TP-Extended, the length of the receive data must not exceed the value assigned to the `length` operand of the NET buffer group definition (`nettbuf`) contained in the NET/Library common definition.>>

TPRECV - Receive a message in a conversational connection

Format

```
01  TPSVCDEF-REC.
    COPY  TPSVCDEF.
01  TPTYPE-REC.
    COPY  TPTYPE.
01  DATA-REC.
    COPY  Data record definition.
01  TPSTATUS-REC.
    COPY  TPSTATUS.

CALL "TPRECV" USING TPSVCDEF-REC TPTYPE-REC  DATA-REC
                    TPSTATUS-REC.
```

Description

TPRECV is used to receive data sent across an open connection from another program. COMM-HANDLE specifies on which open connection to receive data. COMM-HANDLE is a communication handle returned from either TPCONNECT or TPSVCSTART. DATA-REC specifies where the message is read into, and, on input, LEN indicates the maximum number of bytes that should be moved into DATA-REC.

Upon successful return, and for several event types, LEN contains the actual number of bytes moved into DATA-REC. REC-TYPE and SUB-TYPE contain the data's type and sub-type, respectively. If the message is larger than DATA-REC, DATA-REC contains only as many bytes as fit in the record. The remainder of the message is discarded and TPRECV sets TPTRUNCATE.

If LEN is 0 upon successful return, the message has no data portion and DATA-REC was not modified. It is an error for LEN to be 0 on input.

TPRECV can be issued only by the program that does not have control of the connection.

<<Data areas>>

- <<TPSVCDEF-REC

  Specify a value indicating the TPRECV operation and a communication handle. The specifiable values and their meanings will be explained later.>>

- <<TPTYPE-REC

  Indicates the record type and subtype record name of the data to be received.>>

- <<DATA-REC

  Points to the record to be received.>>

- `<<TPSTATUS-REC`

  Will be assigned the return value indicating the result of `TPRECV` execution.>>

  The valid settings of `TPSVCDEF-REC` are as follows:

  `TPNOCHANGE`

  > When this setting is used, the type of `DATA-REC` is not allowed to change. That is, the type and sub-type of the message received must match `REC-TYPE` and `SUB-TYPE`, respectively. Either `TPNOCHANGE` or `TPCHANGE` must be set.

  `TPCHANGE`

  > The type or subtype of the message received is allowed to differ from those specified in `REC-TYPE` and `SUB-TYPE`, respectively, so long as the receiver recognizes the incoming record type. Either `TPNOCHANGE` or `TPCHANGE` must be set.

  `TPNOBLOCK`

  > `TPRECV` does not wait for data to arrive. If data is already available to receive, `TPRECV` gets the data and returns. Either `TPNOBLOCK` or `TPBLOCK` must be set.

  `TPBLOCK`

  > When `TPBLOCK` is specified and no data is available to receive, the caller blocks until data arrives. Either `TPNOBLOCK` or `TPBLOCK` must be set.

  `TPNOTIME`

  > This setting signifies that the caller is willing to block indefinitely and wants to be immune to blocking timeouts. Transaction timeouts may still occur. Either `TPNOTIME` or `TPTIME` must be set.

  `TPTIME`

  > This setting signifies that the caller receives blocking timeouts if a blocking condition exists and the blocking time is reached. Either `TPNOTIME` or `TPTIME` must be set.

  `TPSIGRSTRT`

  > If a signal interrupts any underlying system calls, the interrupted system call is re-issued. Either `TPNOSIGRSTRT` or `TPSIGRSTRT` must be set.

  `TPNOSIGRSTRT`

  > If a signal interrupts any underlying system calls, the interrupted system call is not restarted and the routine fails. Either `TPNOSIGRSTRT` or `TPSIGRSTRT` must be set.

  If an event exists for `COMM-HANDLE`, and `TPRECV` encounters no errors, `TPRECV` returns setting `TP-STATUS` to `TPEEVENT`. The event type is returned in `TP-EVENT`.

Data can be received along with the `TPEV-SVCSUCC`, `TPEV-SVCFAIL`, and `TPEV-SENDONLY` events. Valid events for `TPRECV` are as follows:

`TPEV-DISCONIMM`

Received by the subordinate of a conversation, this event indicates that the originator of the conversation has either issued an immediate disconnect on the connection via `TPDISCON`, or it issued `TPRETURN`, `TXCOMMIT` or `TXROLLBACK` with the connection still open. This event is also returned to the originator or subordinate when a connection is broken due to a communications error (for example, a server, machine, or network failure). Because this is an immediate disconnection notification (that is, abortive rather than orderly), data in transit may be lost. If the two programs were participating in the same transaction, the transaction is marked `rollback-only`. `COMM-HANDLE` is no longer valid.

`TPEV-SENDONLY`

The program on the other end of the connection has relinquished control of the connection. The recipient of this event is allowed to send data but cannot receive any data until it relinquishes control.

`TPEV-SVCERR`

Received by the originator of a conversation, this event indicates that the subordinate of the conversation has issued `TPRETURN`. `TPRETURN` encountered an error that precluded the service from returning successfully. For example, bad arguments may have been passed to `TPRETURN` or it may have been called while the service had open connections to other subordinates. Due to the nature of this event, any application-defined data or return code is not available. The connection has been terminated and `COMM-HANDLE` is no longer valid. If this event occurred as part of the recipient's transaction, the transaction is marked rollback-only.

`TPEV-SVCFAIL`

Received by the originator of a conversation, this event indicates that the subordinate service on the other end of the conversation has finished unsuccessfully as defined by the application (that is, it called `TPRETURN` with `TPFAIL`). If the subordinate service was in control of this connection when `TPRETURN` was called, it can pass a record back to the originator of the connection. As part of ending the service routine, the server has terminated the connection. Thus, `COMM-HANDLE` is no longer valid. If this event occurred as part of the recipient's transaction, the transaction is marked `rollback-only`.

`TPEV-SVCSUCC`

Received by the originator of a conversation, this event indicates that the subordinate service on the other end of the conversation has finished successfully as defined by the application (that is, it called `TPRETURN` with `TPSUCCESS`). As part of ending the service routine, the server has terminated the connection. Thus, `COMM-HANDLE` is no longer valid. If the recipient is in transaction mode, it can either commit (if it is also the initiator) or roll back the transaction causing the work done by the server (if also in transaction mode) to either commit or roll back.

Return value

Upon successful completion, `TPRECV` sets `TP-STATUS` to `TPOK`. If an event exists and no errors were encountered, `TPRECV` sets `TP-STATUS` to `TPEEVENT`. When `TP-STATUS` is set to `TPEEVENT` and `TP-EVENT` is either `TPEV-SVCSUCC` or `TPEV-SVCFAIL`, `APPL-RETURN-CODE` contains an application-defined value that was sent as part of `TPRETURN`. If the size of the incoming message is larger than the size specified in `LEN` on input, `TPTRUNCATE` is set and only `LEN` bytes are moved into `DATA-REC`. The remaining bytes are discarded.

- Errors

Under the following conditions, `TPRECV` fails and sets `TP-STATUS` to one of the values below. Unless otherwise noted, failure does not affect the caller's transaction, if one exists.

`TPEINVAL`

Invalid arguments were given (for example, settings in `TPSVCDEF-REC` are invalid).

`TPEBADDESC`

`COMM-HANDLE` contains an invalid communication handle.

`TPEOTYPE`

Either the type and sub-type of the incoming message are not known to the caller, or `TPNOCHANGE` was set and `REC-TYPE` and `SUB-TYPE` do not match the type and sub-type of the incoming message. If the conversation is part of the caller's current transaction, the transaction is marked rollback-only since the incoming message is discarded. When this error occurs, any event for `COMM-HANDLE` is dropped and the conversation may now be in an indeterminate state. The caller should terminate the conversation.

`TPETIME`

A timeout occurred. If the caller is in transaction mode, a transaction timeout occurred and the transaction is marked rollback-only; otherwise, a blocking timeout occurred and both `TPBLOCK` and `TPTIME` were specified. In either case, neither `DATA-REC` nor `TPTYPE-REC` are changed. If a transaction timeout occurred, any attempts to send or receive messages on any connections or to start a new connection fail with `TPETIME` until the transaction has been rolled back.

`TPEEVENT`

An event occurred and its type is available in `TP-EVENT`.

TPEBLOCK

A blocking condition exists and TPNOBLOCK was specified.

TPEGOTSIG

A signal was received and TPNOSIGRSTRT was specified.

TPEPROTO

TPRECV was called in an improper context.

TPESYSTEM

A communication resource manager system error has occurred. The exact nature of the error is determined in a product-specific manner.

TPEOS

An operating system error has occurred. The exact nature of the error is determined in a product-specific manner.

See also

TPCONNECT, TPDISCON, TPSEND.

<<Notes on use with OpenTP1>>

1. <<The value TPSIGRSTRT is invalid. Regardless of whether this value is set, when a signal is received, the interrupted system call is reinvoked. TPEGOTSIG will never return.>>
2. <<Under the OpenTP1, when a process encounters transaction timeout, it terminates abnormally. Therefore, TPETIME returns only when blocking timeout occurs.>>
3. <<Under the relevant version of the OpenTP1, an error which raises need for rollback causes the return of TPESYSTEM unless otherwise specified by the X/Open. However, the rollback_only state may not come into effect even when TPESYSTEM returns.>>
4. <<When OSI TP communication using the TP1/NET/OSI-TP-Extended is in progress, it cannot be used for conversational services. If an attempt is made to do so, the system operation is unpredictable.>>

TPRETURN - Return from a service routine

Format

```
01  TPSVCRET-REC.
    COPY  TPSVCRET.
01  TPTYPE-REC.
    COPY  TPTYPE.
01  DATA-REC.
    COPY  Data record definition.

COPY  TPRETURN  [REPLACING  TPSVCRET-REC  BY  TPSVCRET-REC]
                [REPLACING  TPTYPE-REC  BY  TPTYPE-REC]
                [REPLACING  DATA-REC  BY  DATA-REC].
```

Description

`TPRETURN` indicates that a service routine has completed. `TPRETURN` is a file containing the last sequence of COBOL code to be executed in the service. It contains references to three data record names: `TPSVCRET-REC`, `TPTYPE-REC` and `DATA-REC` that may be substituted by the record names effectively used in the service routine. Since `TPRETURN` contains an `EXIT PROGRAM` statement, it should be issued in the same routine that was invoked by the communication resource manager so that control can be returned to the communication resource manager (that is, `TPRETURN` should not be invoked in a sub-program of the service routine since control would not return to the communication resource manager).

`TPRETURN` is used to send a service's reply message. If the program receiving the reply is waiting in `TPCALL`, `TPGETRPLY` or `TPRECV`, after a successful call to `TPRETURN`, the reply is available in the receiver's record.

For conversational services, `TPRETURN` also terminates the connection. That is, the service routine cannot call `TPDISCON` directly. To ensure correct results, the program that connected to the conversational service should not call `TPDISCON`; rather, it should wait for notification that the conversational service has completed (that is, it should wait for one of the events, like `TPEV-SVCSUCC` or `TPEV-SVCFAIL`, sent by `TPRETURN`).

If the service routine was in transaction mode, `TPRETURN` places the service's portion of the transaction in a state where it may be either committed or rolled back when the transaction is completed. A service may be invoked multiple times as part of the same transaction so it is not necessarily fully committed nor rolled back until either `TXCOMMIT` or `TXROLLBACK` is called by the originator of the transaction.

`TPRETURN` should be called after receiving all replies expected from service requests initiated by the service routine. Otherwise, depending on the nature of the service, either a `TPESVCERR` error or a `TPEV-SVCERR` event are returned to the program that initiated communication with the service routine. Any outstanding replies that are not received are automatically dropped by the communication resource manager. In addition, the communication handles for those replies become invalid.

TPRETURN should be called after closing all connections initiated by the service. Otherwise, depending on the nature of the service, either a TPESVCERR or a TPEV-SVCERR event is returned to the program that initiated communication with the service routine. Also, an immediate disconnect event (that is, TPEV-DISCONIMM) is sent over all open connections to subordinates.

Concerning control of the connection, if the service routine does not have control over the connection with which it was invoked when it issues TPRETURN, two outcomes are possible. Firstly, if the service routine calls TPRETURN with TP-RETURN-VAL (in TPSVCRET-REC) set to TPFAIL and REC-TYPE (in TPTYPE-REC) set to SPACES (that is, no data is sent), a TPEV-SVCFAIL event is sent to the originator of this conversation. Secondly, if any other invocation of TPRETURN is used, a TPEV-SVCERR event is sent to the originator.

Since a conversational service has only one open connection that it did not initiate, the communication resource manager knows over which communication handle data (and any event) should be sent. For this reason, a communication handle is not passed to TPRETURN.

<<Data areas>>

- <<TPSVCRET-REC

  Specify TP-RETURN-VAL and APPL-CODE.>>

- <<TPTYPE-REC

  Indicates the record type and subtype record name of the send data.>>

- <<DATA-REC

  Points to the send data.>>

The following is a description of TPRETURN's arguments. TP-RETURN-VAL can be set to one of the following:

TPSUCCESS

The service has terminated successfully. If data is present, it is sent (barring any failures processing the return). If the caller is in transaction mode, TPRETURN places the caller's portion of the transaction in a state such that it can be committed when the transaction ultimately commits. Note that a call to TPRETURN does not necessarily finalize an entire transaction. Also, even though the caller indicates success, if there are any outstanding replies or open connections to subordinates, or if any work done within the service caused its transaction to be marked rollback-only, a failed message is sent (that is, the recipient of the reply receives a TPESVCERR indication or a TPEV-SVCERR event). Note that if a transaction becomes rollback-only while in the service routine for any reason, TP-RETURN-VAL should be set to TPFAIL. If TPSUCCESS is specified for a conversational service, a TPEV-SVCSUCC event is generated.

TPFAIL

The service has terminated unsuccessfully from an application standpoint. An error is reported to the program receiving the reply. That is, the call to get the reply has failed and the recipient receives a TPSVCFAIL indication or a TPEV-SVCFAIL event. If the caller is in transaction mode, TPRETURN marks the transaction as rollback-only (note that the transaction may already be marked rollback-only). Barring any failures in processing the return, the caller's data is sent, if present. One reason for not sending the caller's data is when a transaction timeout has occurred. In this case, the program waiting for the reply receives an error of TPETIME.

If TP-RETURN-VAL does not contain one of these two values, TPFAIL is assumed.

An application-defined return code, APPL-CODE (in TPSVCRET-REC), may be sent to the program receiving the service reply. This code is sent regardless of the setting of TP-RETURN-VAL as long as a reply can be successfully sent (that is, as long as the receiving call returns success or TPESVCFAIL, or receives one of the events TPEV-SVCSUCC or TPEV-SVCFAIL). The value of APPL-CODE is available to the receiver in APPL-RETURN-CODE in TPSTATUS-REC.

DATA-REC is the record to be sent and LEN (in TPTYPE-REC) specifies the amount of data in DATA-REC that should be sent. Note that if DATA-REC is a record of a type that does not require a length to be specified, LEN is ignored (and may be 0). If DATA-REC is a record of a type that does require a length, LEN must not be zero. If REC-TYPE does not specify subtype, SUB-TYPE is ignored (and may be SPACES). If REC-TYPE is SPACES, DATA-REC and LEN are ignored. In this case, if a reply is expected by the program that invoked the service, a reply is sent with no data portion. If no reply is expected, TPRETURN ignores any data passed to it and returns sending no reply.

If the service is conversational, there are two cases where the data record is not transmitted:

- If the connection has already been terminated when the call is made (that is, the caller has received TPEV-DISCONIMM on the connection), this call simply ends the service routine and rolls back the current transaction, if one exists. In this case, the caller's data record cannot be transmitted.
- If the caller does not have control of the connection, either TPEV-SVCFAIL or TPEV-SVCERR is sent to the originator of the connection as described above. Regardless of which event the originator receives, no data record is transmitted; however, if the originator receives the TPEV-SVCFAIL event, the return code is available in the originator's APPL-RETURN-CODE in TPSTATUS-REC.

Return value

Since TPRETURN contains an EXIT PROGRAM statement, no value is returned to the caller, nor does control return to the service routine. If a service routine returns without using TPRETURN (that is, it uses an EXIT PROGRAM statement directly or falls out of the service routine), the server returns a service error to the service requester. In addition, all open connections to subordinates are disconnected immediately, and any outstanding asynchronous replies are dropped. If the server was in transaction mode at the time of failure, the transaction is marked rollback-only. Note also that if TPRETURN is used outside a service routine (that is, by routines that are not services), it returns having no effect.

- Errors

Since TPRETURN ends the service routine, any errors encountered either in handling arguments or in processing cannot be indicated to the routine's caller. Such errors cause TP-STATUS to be set to TPESVCERR for a program receiving the service's outcome via either TPCALL or TPGETRPLY, and cause the event, TPEV-SVCERR, to be sent over the conversation to a program using TPSEND or TPRECV.

See also

TPCALL, TPCONNECT, TPDISCON, TPSEND, TPSVCSTART.

<<Notes on use with OpenTP1>>

1. <<The effects of the COPY statement and the REPLACING clause vary depending on the COBOL compiler. When COBOL85 is in use, write the following code:>>

```
COPY TPRETURN [{REPLACING  [TPSVCRET-REC BY TPSVCRET-REC]
                           [TPTYPE-REC BY TPTYPE-REC]
                           [DATA-REC BY DATA-REC]}].
```

TPSEND - Send a message in a conversational connection

Format

```
01  TPSVCDEF-REC.
    COPY  TPSVCDEF.
01  TPTYPE-REC.
    COPY  TPTYPE.
01  DATA-REC.
    COPY  Data record definition.
01  TPSTATUS-REC.
    COPY  TPSTATUS

CALL  "TPSEND"  USING  TPSVCDEF-REC  TPTYPE-REC  DATA-REC
                       TPSTATUS-REC.
```

Description

TPSEND is used to send data across an open connection to another program. The caller must have control of the connection. COMM-HANDLE specifies the open connection over which data is sent. COMM-HANDLE is a communication handle returned from either TPCONNECT or TPSVCSTART.

DATA-REC contains the data to be sent and LEN specifies how much of the data to send. Note that if DATA-REC is a record of a type that does not require a length to be specified, LEN is ignored (and may be 0). If DATA-REC is a record of a type that does require a length, LEN must not be zero. If REC-TYPE does not have a subtype, SUB-TYPE is ignored (and may be SPACES). If REC-TYPE is SPACES, DATA-REC and LEN are ignored and a message is sent with no data (this might be done, for instance, to grant control of the connection without transmitting any data).

<<Data areas>>

- <<TPSVCDEF-REC

  Specify a value indicating the TPSEND operation and a communication handle. The specifiable values and their meanings will be explained later.>>

- <<TPTYPE-REC

  Indicates the data type and subtype record name of the send data.>>

- <<DATA-REC

  Points to the send data.>>

- <<TPSTATUS-REC

  Will be assigned the return value indicating the result of TPSEND execution.>>

  The valid settings of TPSVCDEF-REC are as follows:

TPRECVONLY

This setting signifies that, after the caller's data is sent, the caller gives up control of the connection (that is, the caller cannot issue any more TPSEND calls). When the receiver on the other end of the connection receives the data sent by TPSEND, it also receives an event (TPEV-SENDONLY) indicating that it has control of the connection (and cannot issue any more TPRECV calls). Either TPRECVONLY or TPSENDONLY must be set.

TPSENDONLY

This setting signifies that the caller wants to remain in control of the connection. Either TPRECVONLY or TPSENDONLY must be set.

TPNOBLOCK

The data and any events are not sent if a blocking condition exists (for example, the internal buffers into which the message is transferred are full). Either TPNOBLOCK or TPBLOCK must be set.

TPBLOCK

When TPBLOCK is specified and a blocking condition exists, the caller blocks until the condition subsides or a timeout occurs (either transaction or blocking timeout). Either TPNOBLOCK or TPBLOCK must be set.

TPNOTIME

This setting signifies that the caller is willing to block indefinitely and wants to be immune to blocking timeouts. Transaction timeouts may still occur. Either TPNOTIME or TPTIME must be set.

TPTIME

This setting signifies that the caller receives blocking timeouts if a blocking condition exists and the blocking time is reached. Either TPNOTIME or TPTIME must be set.

TPSIGRSTRT

If a signal interrupts any underlying system calls, the interrupted system call is re-issued. Either TPNOSIGRSTRT or TPSIGRSTRT must be set.

TPNOSIGRSTRT

If a signal interrupts any underlying system calls, the interrupted system call is not restarted and the routine fails. Either TPNOSIGRSTRT or TPSIGRSTRT must be set.

If an event exists for COMM-HANDLE, TPSEND returns without sending the caller's data. The event type is returned in TP-EVENT. Valid events for TPSEND are as follows:

TPEV-DISCONIMM

Received by the subordinate of a conversation, this event indicates that the originator of the conversation has either issued an immediate disconnect on the connection via `TPDISCON`, or it issued `TPRETURN`, `TXCOMMIT` or `TXROLLBACK` with the connection still open. This event is also returned to the originator or subordinate when a connection is broken due to a communication error (for example, a server, machine, or network failure).

`TPEV-SVCERR`

Received by the originator of a conversation, this event indicates that the subordinate of the conversation has issued `TPRETURN` without having control of the conversation. In addition, `TPRETURN` was issued in a manner different from that described for `TPEV-SVCFAIL` below.

`TPEV-SVCFAIL`

Received by the originator of a conversation, this event indicates that the subordinate of the conversation has issued `TPRETURN` without having control of the conversation. In addition, `TPRETURN` was issued with the command `TPFAIL` and no data record (that is, the `REC-TYPE` passed to `TPRETURN` was set to `SPACES`).

Because each of these events indicates an immediate disconnection notification (that is, abortive rather than orderly), data in transit may be lost. The communication handle used for the connection is no longer valid. If the two programs were participating in the same transaction, the transaction has been marked rollback-only.

Return value

Upon successful completion, `TPSEND` sets `TP-STATUS` to `TPOK`. When `TP-STATUS` is set to `TPEEVENT` and `TP-EVENT` is `TPEV-SVCFAIL`, `APPL-RETURN-CODE` contains an application-defined value that was sent as part of `TPRETURN`.

- Errors

Under the following conditions, `TPSEND` fails and sets `TP-STATUS` to one of the values below. Unless otherwise noted, failure does not affect the caller's transaction, if one exists.

`TPEINVAL`

Invalid arguments were given (for example, settings in `TPSVCDEF-REC` are invalid).

`TPEBADDESC`

`COMM-HANDLE` contains an invalid communication handle.

`TPETIME`

A timeout occurred. If the caller is in transaction mode, a transaction timeout occurred and the transaction is marked rollback-only; otherwise, a blocking timeout occurred and both `TPBLOCK` and `TPTIME` were specified. In either case, neither `DATA-REC` nor `TPTYPE-REC` are changed. If a transaction timeout occurred, any attempts to send or receive messages on any connections or to start a new connection fail with `TPETIME` until the transaction has been rolled back.

`TPEEVENT`

An event occurred and its type is available in `TP-EVENT`. `DATA-REC` is not sent when this error occurs.

`TPEBLOCK`

A blocking condition exists and `TPNOBLOCK` was specified.

`TPEGOTSIG`

A signal was received and `TPNOSIGRSTRT` was specified.

`TPEPROTO`

`TPSEND` was called in an improper context.

`TPESYSTEM`

A communication resource manager system error has occurred. The exact nature of the error is determined in a product-specific manner.

`TPEOS`

An operating system error has occurred. The exact nature of the error is determined in a product-specific manner.

See also

`TPCONNECT`, `TPDISCON`, `TPRECV`, `TPRETURN`.

<<Notes on use with OpenTP1>>

1. <<The value `TPNOBLOCK` is invalid under the relevant version of the OpenTP1. Therefore, the error code `TPEBLOCK` will never return. The OpenTP1 is designed so that if communication is impossible because of blocking, `TPESYSTEM` is returned as when communication is impossible because of network failure.>>
2. <<The value `TPNOTIME` is invalid under the relevant version of the OpenTP1.>>
3. <<The value `TPSIGRSTRT` is invalid. Regardless of whether this value is set, when a signal is received, the interrupted system call is reinvoked. `TPEGOTSIG` will never return.>>
4. <<Under the OpenTP1, when a process encounters transaction timeout, it terminates abnormally. Therefore, `TPETIME` returns only when blocking timeout occurs.>>
5. <<Under the relevant version of the OpenTP1, an error which raises need for

rollback causes the return of `TPESYSTEM` unless otherwise specified by the X/Open. However, the `rollback_only` state may not come into effect even when `TPESYSTEM` returns.>>

6. <<Under the OpenTP1, even if the mate of conversation is running the function `TPDISCON` or `TPRETURN`, the function `TPSEND` cannot generate an event provided that the process which invoked the function `TPSEND` has not received an event.>>
7. <<When OSI TP communication using the TP1/NET/OSI-TP-Extended is in progress, it cannot be used for conversational services. If an attempt is made to do so, the system operation is unpredictable.>>

TPSVCSTART - Start a service routine

Format

```
01  TPSVCDEF-REC.
    COPY  TPSVCDEF.
01  TPTYPE-REC.
    COPY  TPTYPE.
01  DATA-REC.
    COPY  Data record definition.
01  TPSTATUS-REC.
    COPY  TPSTATUS

CALL  "TPSVCSTART"  USING  TPSVCDEF-REC  TPTYPE-REC  DATA-REC
                           TPSTATUS-REC.
```

Description

`TPSVCSTART` is the first routine called when writing a service routine. In fact, it is an error to issue any other XATMI call within a service routine before calling `TPSVCSTART`. `TPSVCSTART` is used to retrieve the service's parameters and data. This routine is used for services that receive requests via `TPCALL` or `TPACALL` routines as well as by services that communicate via `TPCONNECT`, `TPSEND` and `TPRECV` routines.

Service routines processing requests made via either `TPCALL` or `TPACALL` receive, at most, one incoming message (upon successfully returning from `TPSVCSTART`) and send, at most, one reply (upon existing the service routine with `TPRETURN`).

Conversational services, on the other hand, are invoked by connection requests with, at most, one incoming message along with a means of referring to the open connection. Upon successfully returning from `TPSVCSTART`, either the connecting program or the conversational service may send and receive data as defined by the application. The connection is half-duplex in nature meaning that one side controls the conversation (that is, it sends data) until it explicitly gives up control to the other side of the connection.

Concerning transactions, service routines can participate in, at most, one transaction if invoked in transaction mode. As far as the service routine writer is concerned, the transaction ends upon returning from the service routine. If the service routine is not invoked in transaction mode, the service routine may originate as many transactions as it wants using `TXBEGIN`, `TXCOMMIT` and `TXROLLBACK`. Note that `TPRETURN` is not used to complete a transaction. Thus, it is an error to call `TPRETURN` with an outstanding transaction that originated within the service routine.

`DATA-REC` specifies where the service's data is read into, and, on input, `LEN` indicates the maximum number of bytes that should be moved into `DATA-REC`. Upon successful return from `TPSVCSTART`, `LEN` contains the actual number of bytes moved into `DATA-REC`. `REC-TYPE` and `SUB-TYPE` contain the data's type and sub-type, respectively. If the message is larger than DATA-REC, DATA-REC contains only as many bytes as will fit the record. The remainder of the message is discarded and TPSVCSTART sets TPTRUNCATE.

If LEN is 0 upon successful return, the service has no incoming data and DATA-REC was not modified. It is an error for LEN to be 0 on input.

Upon successful return, SERVICE-NAME is populated with the service name that the requesting program used to invoke the service.

<<Data areas>>

- <<TPSVCDEF-REC

  Will be assigned a value that indicates how the service routine was invoked. The specifiable values and their meanings will be explained later.>>

- <<TPTYPE-REC

  Will be assigned the data type and subtype record name of the receive data.>>

- <<DATA-REC

  Will be assigned the receive data.>>

- <<TPSTATUS-REC

  Will be assigned the return value indicating the result of TPSVCSTART execution.>>

The possible settings of TPSVCDEF-REC upon the return of TPSVCSTART are as follows:

TPREQRSP

The service was invoked with either TPCALL or TPACALL. This setting is mutually exclusive with TPCONV.

TPCONV

The service was invoked with TPCONNECT. The communication handle for the conversation is available in COMM-HANDLE. This setting is mutually exclusive with TPREQRSP.

TPNOTRAN

The service routine is not in transaction mode. This setting is mutually exclusive with TPTRAN.

TPTRAN

The service routine is in transaction mode. This setting is mutually exclusive with TPNOTRAN.

TPNOREPLY

The program invoking the service routine is not expecting a reply. This setting is meaningful only when TPREQRSP is set. This setting is mutually exclusive with TPREPLY.

TPREPLY

The program invoking the service routine is expecting a reply. This setting is meaningful only when TPREQRSP is set. This setting is mutually exclusive with TPNOREPLY.

TPSENDONLY

The service is invoked such that it can send data across the connection and the program on the other end of the connection can only receive data. This setting is meaningful only when TPCONV is set. This setting is mutually exclusive with TPRECVONLY.

TPRECVONLY

The service is invoked such that it can only receive data from the connection and the program on the other end of the connection can send data. This setting is meaningful only when TPCONV is set. This setting is mutually exclusive with TPSENDONLY.

Return value

Upon successful completion, TPSVCSTART sets TP-STATUS to TPOK. If the size of the incoming message is larger than the size specified in LEN on input, TPTRUNCATE is set and only LEN bytes are moved into DATA-REC. The remaining bytes are discarded.

- Errors

Under the following conditions, TPSVCSTART fails and sets TP-STATUS to one of the following values:

TPEINVAL

Invalid arguments were given (for example, LEN is 0).

TPEPROTO

TPSVCSTART was called in an improper context.

TPESYSTEM

A communication resource manager system error has occurred. The exact nature of the error is determined in a product-specific manner.

TPEOS

An operating system error has occurred. The exact nature of the error is determined in a product-specific manner.

See also

`TPACALL`, `TPCALL`, `TPCONNECT`, `TPRETURN`.

TPUNADVERTISE - Unadvertise a service name

Format

```
01  SERVICE-NAME  PIC X(15).
01  TPSTATUS-REC.
    COPY  TPSTATUS.

CALL  "TPUNADVERTISE"  USING  SERVICE-NAME  TPSTATUS-REC.
```

Description

TPUNADVERTISE allows a server to unadvertise a service that it offers. By default, a server's services are advertised when it is booted and they are unadvertised when it is shutdown.

TPUNADVERTISE removes SERVICE-NAME as an advertised service for the server. SERVICE-NAME cannot be SPACES. Also, SERVICE-NAME should be 15 characters or fewer. Longer names are accepted and truncated to 15 characters. Care should be taken such that truncated names do not match other service names.

<<Data areas>>

- <<SERVICE-NAME

  Specify the service name of the service.>>

- <<TPSTATUS-REC

  Will be assigned the return value indicating the result of TPUNADVERTISE execution.>>

Return value

Upon successful completion, TPUNADVERTISE sets TP-STATUS to TPOK.

- Errors

  Under the following conditions, TPUNADVERTISE fails and sets TP-STATUS to one of the following values:

  TPEINVAL

  SERVICE-NAME is SPACES.

  TPENOENT

  SERVICE-NAME is not currently advertised by the server.

  TPEPROTO

  TPUNADVERTISE was called in an improper context.

`TPESYSTEM`

A communication resource manager system error has occurred. The exact nature of the error is determined in a product-specific manner.

`TPEOS`

An operating system error has occurred. The exact nature of the error is determined in a product-specific manner.

See also

`TPADVERTISE`.

<<Notes on use with OpenTP1>>

1. <<Suppose that load balancing is used on one node (multiserver configuration). When the function `TPUNADVERTISE` is invoked from one of the processes, the service becomes unavailable to all processes which undergo load balancing. When the function `TPADVERTISE` is later invoked to advertise the service, service requests from the processes can be accepted.>>
2. <<Suppose that load balancing is used on multiple nodes (internode load-balancing and extended internode load-balancing). When the function `TPUNADVERTISE` is invoked from a process on a node, the service becomes unavailable on that node. However, the servers at other nodes can accept service requests. When the function `TPADVERTISE` is later invoked to advertise the service, service requests are acceptable.>>

TX-interfaced application programming interface (TX~)

This section explains the syntax of the API functions which implement the TX interface. The text in this section is quoted from *5. COBOL Reference Manual Pages* which is the syntax reference section of the *X/Open CAE Specification Distributed TP: The TX (Transaction Demarcation) Specification published by X/Open Company Limited*.

The TX interface has the following API functions:

- `TXINTRO` - COBOL data structure
- `TXBEGIN` - Begin a global transaction.
- `TXCLOSE` - Close a set of resource managers.
- `TXCOMMIT` - Commit a global transaction.
- `TXINFORM` - Return global transaction information.
- `TXOPEN` - Open a set of resource managers.
- `TXROLLBACK` - Roll back a global transaction.
- `TXSETCOMMITRET` - Set `commit_return` characteristic.
- `TXSETTIMEOUT` - Set `transaction_timeout` characteristic.
- `TXSETTRANCTL` - Set `transaction_control` characteristic.

TX-interfaced API functions whose names begin with TX can be used with UAPs under either the TP1/Server Base or TP1/LiNK.

TXINTRO - COBOL data structures

Description

- Overview

  There is syntactic description in COBOL for each call of the TX interface.

  Each call is described by the following items:

  - Reference to the COBOL records in the `Working-Storage Section` needed by that call, by a `COPY` statement.
  - Synopsis of the call in the `Procedure Division`.
  - A description of the call.
  - List of the return codes.

- Data Structures Used by the COBOL TX Interface

  Two COBOL records: `TX-RETURN-STATUS` and `TX-INFO-AREA` are commonly used by the TX calls. They are expected to be defined in the `Working-Storage Section` by specification of `COPY` statements.

  <<These two records are stored in the `$DCDIR/include/COBOL`. When compiling the UAP, it is necessary to designate this directory as the location of the `COPY` file. For details about the specifications needed for compilation, see a manual for the COBOL language version you are using.>>

- `TX-RETURN-STATUS`

  Every function described in this chapter takes an instance of this record as a parameter. It is used to return a value to the caller. This record is expected to be used in the context:

  <<Since the value `TXSTATUS` of `TX-RETURN-STATUS` cannot be referenced by each call because FILTER is given as the data name. Redefine `TX-RETURN-STATUS` to have a different name so that each call can make reference using this new name. For details, see *6.4 Coding samples for X/Open-compliant UAPs*.>>

```
01 TX-RETURN-STATUS.
   COPY TXSTATUS.
```

`TXSTATUS` is a COBOL text library defining a signed integer that may be assigned one of the following values:

```
  05  FILLER PIC S9(9) COMP-5.
      88 TX-NOT-SUPPORTED      VALUE 1.
*        Normal execution
      88 TX-OK                 VALUE 0.
*        Normal execution
      88 TX-OUTSIDE            VALUE -1.
*        Application is in an RM local transaction
      88 TX-ROLLBACK           VALUE -2.
*        Transaction was rolled back
      88 TX-MIXED              VALUE -3.
*        Transaction was partially committed and partially *
rolled back
      88 TX-HAZARD             VALUE -4.
*        Transaction may have been partially committed and
*        partially rolled back
      88 TX-PROTOCOL-ERROR     VALUE -5.
*        Routine invoked in an improper context
      88 TX-ERROR              VALUE -6.
*        Transient error
      88 TX-FAIL               VALUE -7.
*        Fatal error
      88 TX-EINVAL             VALUE -8.
*        Invalid arguments were given
      88 TX-COMMITTED          VALUE -9.
*        The transaction was heuristically committed
      88 TX-NO-BEGIN           VALUE -100.
*        Transaction committed plus new transaction could *
not be started
      88 TX-ROLLBACK-NO-BEGIN  VALUE -102.
*        Transaction rollback plus new transaction could
*        not be started
      88 TX-MIXED-NO-BEGIN     VALUE -103.
*        Mixed plus new transaction could not be started
      88 TX-HAZARD-NO-BEGIN    VALUE -104.
*        Hazard plus new transaction could not be started
      88 TX-COMMITTED-NO-BEGIN VALUE -109.
*        Heuristically committed plus transaction could
*        not be started
```

- `TX-INFO-AREA`

  This record defines a data structure where the result of the `TXINFORM` call is stored. It is expected to be used in the context:

```
01 TX-INFO-AREA.
   COPY TXINFDEF.
```

`TXINFDEF` is a COBOL text library defining a record as follows:

```
*  XID record
   05 XID_REC.
      10 FORMAT-ID   PIC S9(9) COMP-5.
*  A value of -1 in FORMAT-ID means that the XID is null
      10 GTRID-LENGTH   PIC S9(9) COMP-5.
      10 BRANCH-LENGTH  PIC S9(9) COMP-5.
      10 XID-DATA   PIC X(128).
*  Transaction mode settings
   05 TRANSACTION-MODE  PIC S9(9) COMP-5.
      88 TX-NOT-IN-TRAN     VALUE 0.
      88 TX-IN-TRAN     VALUE 1.
*  Commit_return settings
   05  COMMIT-RETURN    PIC S9(9) COMP-5.
     88 TX-COMMIT-COMPLETED  VALUE 0.
     88 TX-COMMIT-DECISION-LOGGED  VALUE 1.
*  Transaction_control settings
   05  TRANSACTION-CONTROL  PIC S9(9) COMP-5.
     88 TX-UNCHAINED      VALUE 0.
     88 TX-CHAINED      VALUE 1.
*  Transaction_timeout value
   05  TRANSACTION-TIMEOUT  PIC S9(9) COMP-5.
     88 NO-TIMEOUT  VALUE 0.
*  Transaction_state information
   05  TRANSACTION-STATE  PIC S9(9) COMP-5.
     88 TX-ACTIVE  VALUE 0.
     88 TX-TIMEOUT-ROLLBACK-ONLY  VALUE 1.
     88 TX-ROLLBACK-ONLY  VALUE 2.
```

TXBEGIN - Begin a transaction

Format

```
  DATA DIVISION.
*   Include TX definitions.
  01 TX-RETURN-STATUS.
       COPY  TXSTATUS.

  PROCEDURE DIVISION.
  CALL "TXBEGIN" USING TX-RETURN-STATUS.
```

Description

TXBEGIN is used to place the calling thread of control in transaction mode. The calling thread must first ensure that its linked resource managers have been opened (by mean of TXOPEN) before it can start transactions. TXBEGIN fails (with a TX-RETURN-STATUS value of TX-PROTOCOL-ERROR) if the caller is already in transaction mode or TXOPEN has not been called.

Once in transaction mode, the calling thread must call TXCOMMIT or TXROLLBACK to complete its current transaction. There are certain cases related to transaction chaining where TXBEGIN does not need to be called explicitly to start a transaction. See TXCOMMIT and TXROLLBACK for details.

<<TXBEGIN cannot be called from MHPs.>>

Optional Set-up

- TXSETTIMEOUT

<<Data areas>>

- <<TX-RETURN-STATUS

  The results of TXBEGIN execution are returned to this area.>>

Return value

Upon successful completion, TXBEGIN sets TX-OK, a non-negative return value. <<0 is returned.>>

- Errors

  Under the following conditions, TXBEGIN fails and sets one of these negative values:

  TX-OUTSIDE

    The transaction manager is unable to start a global transaction because the calling thread of control is currently participating in work outside any global transaction with one or more resource managers. All such work must be completed before a global transaction can be started. The caller's state with respect to the local transaction is unchanged.

`TX-PROTOCOL-ERROR`

The function was called in an improper context (for example, the caller is already in transaction mode). The caller's state with respect to transaction mode is unchanged.

`TX-ERROR`

Either the transaction manager or one or more of the resource managers encountered a transient error trying to start a new transaction. When this error is returned, the caller is not in transaction mode. The exact nature of the error is determined in a product-specific manner.

`TX-FAIL`

Either the transaction manager or one or more of the resource managers encountered a fatal error. The nature of the error is such that the transaction manager and/or one or more of the resource managers can no longer perform work on behalf of the application. When this error is returned, the caller is not in transaction mode.

See also

`TXCOMMIT`, `TXOPEN`, `TXROLLBACK`, `TXSETTIMEOUT`.

Application usage

XA-compliant resource managers must be successfully opened to be included in the global transaction. (See `TXOPEN` for details.)

<<Notes on use with OpenTP1>>

1. <<Before the SPP can start transaction processing, `TXBEGIN` must be called. If the caller has called `TXBEGIN`, the SPP considers that transaction processing has begun.>>
2. <<Processes which create a transaction using `TXBEGIN` must have activated UAP executable files which are correctly linked according to the description in this manual.>>
3. <<`TXBEGIN` cannot be used along with OpenTP1 `CBLDCTRN`.>>

TXCLOSE - Close a set of resource managers

Format

```
DATA DIVISION.
*  Include TX definitions.
 01  TX-RETURN-STATUS.
     COPY  TXSTATUS.

 PROCEDURE DIVISION.
 CALL "TXCLOSE" USING TX-RETURN-STATUS.
```

Description

`TXCLOSE` closes a set of resource managers in a portable manner. It invokes a transaction manager to read information specific to the resource manager in a manner specific to the transaction manager and pass this information to the resource managers linked to the caller.

`TXCLOSE` closes all resource managers to which the caller is linked. This function is used in place of close calls specific to the resource manager and allows an application program to be free of calls, which may hinder portability. Since resource managers differ in their termination semantics, the specific information needed to close a particular resource manager must be published by each resource manager.

`TXCLOSE` should be called when an application thread of control no longer wishes to participate in global transactions. `TXCLOSE` fails (returning `TX-PROTOCOL-ERROR`) if the caller is in transaction mode. That is, no resource managers are closed even though some may not be participating in the current transaction.

When `TXCLOSE` sets success (`TX-OK`), all resource managers linked to the calling thread are closed.

<<Data areas>>

- <<`TX-RETURN-STATUS`

  The results of `TXCLOSE` execution are returned to this area.>>

Return value

Upon successful completion, `TXCLOSE` sets `TX-OK`, a non-negative value. <<`0` is returned.>> <<The set of resource managers linked to the `TXCLOSE` caller is closed.>>

- Errors

  Under the following conditions, `TXCLOSE` fails and sets one of these negative values:

  `TX-PROTOCOL-ERROR`

The function was called in an improper context (for example, the caller is in transaction mode). No resource managers are closed.

`TX-ERROR`

Either the transaction manager or one or more of the resource managers encountered a transient error. The exact nature of the error is determined in a product-specific manner. All resource managers that could be closed are closed.

`TX-FAIL`

Either the transaction manager or one or more of the resource managers encountered a fatal error. The nature of the error is such that the transaction manager and/or one or more of the resource managers can no longer perform work on behalf of the application.

See also

`TXOPEN`.

<<Notes on use with OpenTP1>>

1. <<`TXCLOSE` can close only resource managers complying with the X/Open XA interface.>>

TXCOMMIT - Commit a global transaction

Format

```
  DATA DIVISION.
*  Include TX definitions.
  01 TX-RETURN-STATUS.
       COPY  TXSTATUS.

  PROCEDURE DIVISION.
  CALL "TXCOMMIT" USING TX-RETURN-STATUS.
```

Description

`TXCOMMIT` is used to commit the work of the transaction active in the caller's thread of control.

If the `transaction_control` characteristic (see `TXSETTRANCTL`) is

`TX-UNCHAINED`, when `TXCOMMIT` returns, the caller is no longer in transaction mode. However, if the `transaction_control` characteristic is

`TX-CHAINED`, when `TXCOMMIT` returns, the caller remains in transaction mode on behalf of a new transaction (see the *Return value* and *Errors* sections below).

Optional set-up

- `TXSETCOMMITRET`
- `TXSETTIMEOUT`
- `TXSETTRANCTL`

<<Data areas>>

- <<`TX-RETURN-STATUS`

  The results of `TXCLOSE` execution are returned to this area.>>

Return value

Upon successful completion, `TXCOMMIT` sets `TX-OK`, a non-negative return value. <<`0` is returned.>> <<If the `transaction_control` characteristic is set to `TX-CHAINED`, a new transaction is started.>>

- Errors

  Under the following conditions, `TXCOMMIT` fails and sets one of these negative values:

  `TX-NO-BEGIN`

    The transaction committed successfully; however, a new transaction could not be started and the caller is no longer in transaction mode. This return value occurs only when the `transaction_control` characteristic is `TX-CHAINED`.

`TX-ROLLBACK`

The transaction could not commit and has been rolled back. In addition, if the `transaction_control` characteristic is `TX-CHAINED`, a new transaction is started.

`TX-ROLLBACK-NO-BEGIN`

The transaction could not commit and has been rolled back. In addition, a new transaction could not be started and the caller is no longer in transaction mode. This return value can occur only when the `transaction_control` characteristic is `TX-CHAINED`.

`TX-MIXED`

The transaction was partially committed and partially rolled back. In addition, if the `transaction_control` characteristic is `TX-CHAINED`, a new transaction is started.

`TX-MIXED-NO-BEGIN`

The transaction was partially committed and partially rolled back. In addition, a new transaction could not be started and the caller is no longer in transaction mode. This return value can occur only when the `transaction_control` characteristic is `TX-CHAINED`.

`TX-HAZARD`

Due to a failure, the transaction may have been partially committed and partially rolled back. In addition, if the `transaction_control` characteristic is `TX-CHAINED`, a new transaction is started.

`TX-HAZARD` will also be returned if `00000001` is assigned to the `trn_extend_function` operand in the transaction service definition and if the resource manager returns `XAER_NOTA` after a one-phase commit.

`TX-HAZARD-NO-BEGIN`

Due to a failure, the transaction may have been partially committed and partially rolled back. In addition, a new transaction could not be started and the caller is no longer in transaction mode. This return value can occur only when the `transaction_control` characteristic is `TX-CHAINED`.

`TX-HAZARD-NO-BEGIN` will also be returned if `00000001` is assigned to the `trn_extend_function` operand in the transaction service definition and if the resource manager returns `XAER_NOTA` after a one-phase commit.

`TX-PROTOCOL-ERROR`

The function was called in an improper context (for example, the caller is not in transaction mode). The caller's state with respect to transaction mode is not changed.

`TX-FAIL`

Either the transaction manager or one or more of the resource managers encountered a fatal error. The nature of the error is such that the transaction manager and/or one or more of the resource managers can no longer perform work on behalf of the application. The caller's state with respect to the transaction is unknown.

See also

`TXBEGIN`, `TXSETCOMMITRET`, `TXSETTRANCTL`, `TXSETTIMEOUT`.

<<Notes on use with OpenTP1>>

1. <<`TXCOMMIT` can be called only by processes of the UAP which has started the global transaction (the UAP which has issued `TXBEGIN`).>>
2. <<Processes which call `TXCOMMIT` must have activated UAP executable files which are correctly linked according to the description in this manual.>>
3. <<`TXCOMMIT` cannot be used along with OpenTP1 `CBLDCTRN`.>>

TXINFORM - Return global transaction information

Format

```
  DATA DIVISION.
*   Include TX definitions.
  01 TX-RETURN-STATUS.
     COPY  TXSTATUS.
*
  01 TX-INFO-AREA.
     COPY  TXINFDEF.

  PROCEDURE DIVISION.
  CALL "TXINFORM" USING TX-INFO-AREA TX-RETURN-STATUS.
```

Description

TXINFORM sets global transaction information in TX-INFO-AREA. In addition, this function sets a value indicating whether the caller is currently in transaction mode or not.

<<Data areas>>

- <<TX-INFO-AREA>>

  TXINFORM populates the TX-INFO-AREA record with global transaction information. The contents of the TX-INFO-AREA record are described under TXINTRO.

  If TXINFORM is called in transaction mode, TRANSACTION-MODE is set to TX-IN-TRAN, XID-REC is populated with a current transaction branch identifier and TRANSACTION-STATE contains the state of the current transaction. If the caller is not in transaction mode, TRANSACTION-MODE is set to TX-NOT-IN-TRAN and XID-REC is populated with the null XID (see TXINTRO for details). In addition, regardless of whether the caller is in transaction mode, COMMIT-RETURN, TRANSACTION-CONTROL, and TRANSACTION-TIMEOUT contain the current settings of the commit_return and transaction_control characteristics, and the transaction timeout value in seconds.

  The transaction timeout value returned reflects the setting to be used when the next transaction is started. Thus, it may not reflect the timeout value for the caller's current global transaction since calls made to TSXETTIMEOUT after the current transaction was begun may have changed its value.

- <<TX-RETURN-STATUS

  The results of TXINFORM execution are returned to this area.>>

Return value

If the TXINFORM caller is in transaction mode, 1 is returned. If the TXINFORM caller is not in transaction mode, `0` is returned.

- Errors

  Under the following conditions, `TXINFORM` fails and sets one of these negative values:

  `TX-PROTOCOL-ERROR`

  The function was called in an improper context (for example, the caller has not yet called `TXOPEN`).

  `TX-FAIL`

  The transaction manager encountered a fatal error. The nature of the error is such that the transaction manager can no longer perform work on behalf of the application.

Application usage

Within the same global transaction, subsequent calls to `TXINFORM` are guaranteed to provide an XID with the same `gtrid` component, but not necessarily the same `bqual` component.

See also

`TXOPEN`, `TXSETCOMMITRET`, `TXSETTRANCTL`, `TXSETTIMEOUT`.

TXOPEN - Open a set of resource managers

Format

```
DATA DIVISION.
*  Include TX definitions.
 01  TX-RETURN-STATUS.
     COPY  TXSTATUS.

 PROCEDURE DIVISION.
 CALL "TXOPEN" USING TX-RETURN-STATUS.
```

Description

TXOPEN opens a set of resource managers in a portable manner. It invokes a transaction manager to read information specific to the resource manager in a manner specific to the transaction manager and pass this information to the resource managers linked to the caller.

TXOPEN attempts to open all resource managers that have been linked with the application. This function is used in place of open calls specific to the resource manager and allows an application program to be free of calls, which may hinder portability. Since resource managers differ in their initialization semantics, the specific information needed to open a particular resource manager must be published by each resource manager.

If TXOPEN sets TX-ERROR, no resource managers are open. If TXOPEN sets TX-OK, some or all of the resource managers have been opened. Resource managers that are not open return errors specific to the resource manager when accessed by the application. TXOPEN must successfully return before a thread of control participates in global transactions.

Once TXOPEN sets success, subsequent calls to TXOPEN (before an intervening call to TXCLOSE) are allowed. However, such subsequent calls return success, and the TM does not attempt to reopen any RMs.

<<Data areas>>

- <<TX-RETURN-STATUS

  The results of TXOPEN execution are returned to this area.>>

Return value

Upon successful completion, TXOPEN sets TX-OK, a non-negative return value. <<0 is returned.>> <<The set of one or more resource managers linked to the TXOPEN caller is opened.>>

- Errors

Under the following conditions, `TXOPEN` fails and sets one of these negative values:

`TX-ERROR`

Either the transaction manager or one or more of the resource managers encountered a transient error. No resource managers are open.

`TX-FAIL`

Either the transaction manager or one or more of the resource managers encountered a fatal error. The nature of the error is such that the transaction manager and/or one or more of the resource managers can no longer perform work on behalf of the application.

Alternatively, an error occurred in the transaction manager because the execution environment was in non-journal operation mode.

See also

`TXCLOSE`.

<<Notes on use with OpenTP1>>

1. <<Processes which create a transaction using `TXBEGIN` must have activated UAP executable files which are correctly linked according to the description in this manual.>>

TXROLLBACK - Roll back a global transaction

Format

```
DATA DIVISION.
*  Include TX definitions.
 01  TX-RETURN-STATUS.
     COPY  TXSTATUS.

 PROCEDURE DIVISION.
 CALL "TXROLLBACK" USING TX-RETURN-STATUS.
```

Description

`TXROLLBACK` is used to roll back the work of the transaction active in the caller's thread of control.

If the `transaction_control` characteristic (see `TXSETTRANCTL`) is

`TX-UNCHAINED`, when `TXROLLBACK` returns, the caller is no longer in transaction mode. However, if the `transaction_control` characteristic is

`TX-CHAINED`, when `TXROLLBACK` returns, the caller remains in transaction mode on behalf of a new transaction (see the *Return value* and *Errors* sections below).

Optional set-up

- `TXSETTRANCTL`
- `TXSETTIMEOUT`

<<`TXROLLBACK` cannot be called from MHPs.>>

<<Data areas>>

- <<`TX-RETURN-STATUS`

  The results of `TXROLLBACK` execution are returned to this area.>>

Return value

Upon successful completion, `TXROLLBACK` sets `TX-OK`, a non-negative return value.

<<`0` is returned.>>

<<A new global transaction is started if the transaction_control characteristic is `TX-CHAINED`.>>

<<If the SPP which has issued `TXROLLBACK` is not the root transaction branch, actual rollback is not conducted, but only the fact that the transaction branch is in the rollback_only state is recorded. Transaction mode remains in effect until a rollback instruction is given during synchronization point processing for the root transaction

branch.>>

- Errors

  Under the following conditions, `TXROLLBACK` fails and sets one of these negative values.

  `TX-NO-BEGIN`

  The transaction rolled back; however, a new transaction could not be started and the caller is no longer in transaction mode. This return value occurs only when the `transaction_control` characteristic is `TX-CHAINED`.

  `TX-MIXED`

  The transaction was partially committed and partially rolled back. In addition, if the `transaction_control` characteristic is `TX-CHAINED`, a new transaction is started.

  `TX-MIXED-NO-BEGIN`

  The transaction was partially committed and partially rolled back. In addition, a new transaction could not be started and the caller is no longer in transaction mode. This return value can occur only when the `transaction_control` characteristic is `TX-CHAINED`.

  `TX-HAZARD`

  Due to a failure, the transaction may have been partially committed and partially rolled back. In addition, if the `transaction_control` characteristic is `TX-CHAINED`, a new transaction is started.

  `TX-HAZARD-NO-BEGIN`

  Due to a failure, the transaction may have been partially committed and partially rolled back. In addition, a new transaction could not be started and the caller is no longer in transaction mode. This return value can occur only when the `transaction_control` characteristic is `TX-CHAINED`.

  `TX-COMMITTED`

  The transaction was heuristically committed. In addition, if the `transaction_control` characteristic is `TX-CHAINED`, a new transaction is started.

  `TX-COMMITTED-NO-BEGIN`

  The transaction was heuristically committed. In addition, a new transaction could not be started and the caller is no longer in transaction mode. This return value can occur only when the `transaction_control` characteristic is `TX-CHAINED`.

  `TX-PROTOCOL-ERROR`

The function was called in an improper context (for example, the caller is not in transaction mode).

`TX-FAIL`

Either the transaction manager or one or more of the resource managers encountered a fatal error. The nature of the error is such that the transaction manager and/or one or more of the resource managers can no longer perform work on behalf of the application. The caller's state with respect to the transaction is unknown.

See also

`TXBEGIN`, `TXSETTRANCTL`, `TXSETTIMEOUT`.

<<Notes on use with OpenTP1>>

1. <<If the `transaction_control` characteristic is `TX-CHAINED`, only the root transaction branch (the UAP which has called `TXBEGIN`) can call `TXROLLBACK`.>>
2. <<If the `transaction_control` characteristic is `TX-UNCHAINED`, `TXROLLBACK` can be called by no-root transaction branches. However, processing varies depending on the caller is a root or non-root transaction branch. If the caller is the root branch, it requests non-root branches via RPC for rollback. If the `TXROLLBACK` caller is a non-root branch, the caller only records `rollback_only` and does not call a rollback request via RPC to the root branch. The caller non-root branch will initiate rollback after it is instructed by the root branch.>>
3. <<`TXROLLBACK` cannot be used along with OpenTP1 CBLDCTRN. >>

TXSETCOMMITRET - Set commit_return characteristic

Format

```
 DATA DIVISION.
*  Include TX definitions.
 01 TX-RETURN-STATUS.
    COPY  TXSTATUS.
*
 01 TX-INFO-AREA.
    COPY  TXINFDEF.

 PROCEDURE DIVISION.
 CALL "TXSETCOMMITRET" USING TX-INFO-AREA TX-RETURN-STATUS.
```

Description

TXSETCOMMITRET sets the commit_return characteristic to the value specified in COMMIT-RETURN. This characteristic affects the way TXCOMMIT behaves with respect to returning control to its caller. TXSETCOMMITRET may be called regardless of whether its caller is in transaction mode. This setting remains in effect until changed by a subsequent call to TXSETCOMMITRET.

The initial value of the commit_return characteristic depends on the product specification. <<For OpenTP1, the return value is TX-COMMIT-COMPLETED.>>

<<Data areas>>

- <<TX-INFO-AREA>>

The valid settings for COMMIT-RETURN are as follows:

{TX-COMMIT-DECISION-LOGGED|TX-COMMIT-COMPLETED}

TX-COMMIT-DECISION-LOGGED

<<This value is not supported under the relevant version of the OpenTP1. If TX_COMMIT-DECISION-LOGGED is assigned to TX-INFO-AREA, TXSETCOMMITRET returns with the return value of TX-NOT-SUPPORTED.>>

This flag indicates that TXCOMMIT should return after the commit decision has been logged by the first phase of the two-phase commit protocol but before the second phase has completed. This setting allows for faster response to the caller of TXCOMMIT. However, there is a risk that a transaction has a heuristic outcome, in which case the caller does not find out about this situation by means of return codes from TXCOMMIT. <<Note that the resource manager has made a heuristic decision using a method specific to it.>> Under normal conditions, participants that promise to commit during the first phase do so during the second phase. In certain unusual circumstances however (for example, long-lasting network or node failures) phase 2 completion may not be possible and heuristic results may occur. A transaction manager may choose not to select this flag and returns `TX-NOT-SUPPORTED` to indicate that this value is not set.

- `TX-COMMIT-COMPLETED`

  This flag indicates that `TXCOMMIT` should return after the two-phase commit protocol has ended completely. Setting this flag allows the caller of `TXCOMMIT` to see return codes even if heuristic results occur in phase 2 of commitment. A transaction manager may choose not to support this feature and can include notification indicating that the flag cannot be used in the reason for returning `TX-NOT-SUPPORTED`.

Return value

<<When return value is `0`>> upon successful completion, `TXSETCOMMITRET` returns `TX-OK`, a non-negative return value. <<In this case, the `commit_return` characteristic is changed to the value set in `TX-INFO-AREA`.>>

<<When return value is positive>> upon successful completion, `TXSETCOMMITRET` returns `TX-NOT-SUPPORTED`, a non-negative return value.

<<The transaction manager does not support the value set in `TX-INFO-AREA`.>> In this case, the `commit_return` characteristic remains set to its existing value. The transaction manager must make either `TX-COMMIT-COMPLETED` or `X-COMMIT-DECISION-LOGGED` available as the `COMMIT-RETURN` value. <<For OpenTP1, the return value is `TX-COMMIT-RETURN`.>>

- Errors

Under the following conditions, `TXSETCOMMITRET` does not change the setting of the `commit_return` characteristic and sets one of these negative values:

`TX-EINVAL`

`COMMIT-RETURN` is not one of `TX-COMMIT-DECISION-LOGGED` or `TX-COMMIT-COMPLETED`.

`TX-PROTOCOL-ERROR`

The function was called in an improper context (for example, the caller has not yet called `TXOPEN`).

`TX-FAIL`

The transaction manager encountered a fatal error. The nature of the error is such that the transaction manager can no longer perform work on behalf of the application.

See also

`TXCOMMIT`, `TXOPEN`, `TXINFORM`.

<<Notes on use with OpenTP1>>

1. <<`TXSETCOMMITRET` cannot be used along with OpenTP1 `CBLDCTRN`.>>

TXSETTIMEOUT - Set transaction_timeout characteristic

Format

```
 DATA DIVISION.
*  Include TX definitions.
 01 TX-RETURN-STATUS.
    COPY  TXSTATUS.
*
 01 TX-INFO-AREA.
    COPY  TXINFDEF.

 PROCEDURE DIVISION.
 CALL "TXSETTIMEOUT" USING TX-INFO-AREA TX-RETURN-STATUS.
```

Description

TXSETTIMEOUT sets the transaction_timeout characteristic to the value specified in TRANSACTION-TIMEOUT. This value specifies the time period in which the transaction must complete before becoming susceptible to transaction timeout; that is, the interval between the AP calling TXBEGIN and TXCOMMIT or TXROLLBACK. TXSETTIMEOUT may be called regardless of whether its caller is in transaction mode or not. If TXSETTIMEOUT is called in transaction mode, the new timeout value does not take effect until the next transaction.

The initial transaction_timeout value is 0 (no timeout).

<<When a value is specified for trn_expiration_time in system definition, it is treated as the initial value.>>

<<Data areas>>

- <<TX-INFO-AREA>>

  TRANSACTION-TIMEOUT specifies the number of seconds allowed before the transaction becomes susceptible to transaction timeout. It may be set to any value up to the maximum value for an S9(9) COMP 5 as defined by the system. A TRANSACTION-TIMEOUT value of zero disables the timeout feature.

- <<TX-RETURN-STATUS

  The results of TXSETTIMEOUT execution are returned to this area.>>

Return value

Upon successful completion, TXSETTIMEOUT sets TX-OK, a non-negative return value.

<<The transaction_timeout characteristic has become the value assigned to TX-INFO-AREA.>>

- Errors

  Under the following conditions, `TXSETTIMEOUT` does not change the setting of `transaction_timeout` characteristic and sets one of these negative values:

  `TX-EINVAL`

  The timeout value specified is invalid.

  `TX-PROTOCOL-ERROR`

  The function was called in an improper context. For example, the caller has not yet called `TXOPEN`.

  `TX-FAIL`

  The transaction manager encountered an error. The nature of the error is such that the transaction manager can no longer perform work on behalf of the application.

See also

`TXBEGIN`, `TXCOMMIT`, `TXOPEN`, `TXROLLBACK`, `TXINFORM`.

<<Notes on use with OpenTP1>>

1. <<`TXSETTIMEOUT` cannot be used along with OpenTP1 `CBLDCTRN`.>>

TXSETTRANCTL - Set transaction_control characteristic

Format

```
 DATA DIVISION.
*  Include TX definitions.
 01  TX-RETURN-STATUS.
     COPY  TXSTATUS.
*
 01  TX-INFO-AREA.
     COPY  TXINFDEF.

 PROCEDURE DIVISION.
 CALL "TXSETTRANCTL" USING TX-INFO-AREA TX-RETURN-STATUS.
```

Description

`TXSETTRANCTL` sets the `transaction_control` characteristic to the value specified in `TRANSACTION-CONTROL`. This characteristic determines whether `TXCOMMIT` and `TXROLLBACK` start a new transaction before returning to their caller. `TXSETTRANCTL` may be called regardless of whether the application program is in transaction mode. This setting remains in effect until changed by a subsequent call to `TXSETTRANCTL`.

The initial setting for this characteristic is `TX-UNCHAINED`.

<<Data areas>>

- <<`TRANSACTION-CONTROL`>>

  The valid settings for `TRANSACTION-CONTROL` are as follows:

  - `TX-UNCHAINED`

    This flag indicates that `TXCOMMIT` and `TXROLLBACK` should not start a new transaction before returning to their caller. The caller must issue `TXBEGIN` to start a new transaction.

  - `TX-CHAINED`

    This flag indicates that `TXCOMMIT` and `TXROLLBACK` should start a new transaction before returning to their caller.

Return value

Upon successful completion, `TXSETTRANCTL` sets `TX-OK`, a non-negative return value. <<`0` is returned.>> <<The `transaction_control` characteristic was changed to the value of `TRANSACTION-CONTROL`.>>

- Errors

  Under the following conditions, TXSETTRANCTL does not change the setting of the transaction_control characteristic and sets one of these negative values;

  TX-EINVAL

  TRANSACTION-CONTROL is not one of TX-UNCHAINED or TX-CHAINED.

  TX-PROTOCOL-ERROR

  The function was called in an improper context (for example, the caller has not yet called TXOPEN).

  TX-FAIL

  The transaction manager encountered a fatal error. The nature of the error is such that the transaction manager can no longer perform work on behalf of the application.

See also

TXBEGIN, TXCOMMIT, TXOPEN, TXROLLBACK, TXINFORM.

<<Notes on use with Open TP1>>

1. <<TXSETTRANCTL cannot be used along with OpenTP1 CBLDCTRN.>>

Chapter

5. Syntax of OpenTP1 COBOL-UAP Creation Programs (Association Status Notification)

Client/server communication using the OSI TP protocol requires communication event processing SPPs. This chapter explains the COBOL-UAP creation program to be used with the SPPs for a communication event and the format of received communication events.

This chapter contains the following sections:

Association operation (CBLDCXAT)

This section explains the following COBOL-UAP creation program that handles associations and is used with the SPPs for a communication event.

- `CBLDCXAT('CONNECT')` - Establish an association

The COBOL-UAP creation program (`CBLDCXAT`) for handling associations can be used only with the TP1/Server Base. It cannot be used with the TP1/LiNK.

The function for handling associations can be invoked only by SPPs for a communication event. It cannot be invoked by other OpenTP1 UAPs (SUPs, SPPs, and MHPs).

The `server_type` operand in the user service definition for the communication event processing SPP must be assigned `betran`.

If you set the `DATA DIVISION` of COBOL-UAP creation programs, you can use the COBOL language templates as samples. The COBOL templates for manipulating associations are held in `DCXAT.cbl` under the /`BeTRAN`/`examples`/`COBOL`/ directory.

CBLDCXAT('CONNECT') - Establish an association

Format

■ `PROCEDURE DIVISION` specification

```
CALL 'CBLDCXAT' USING unique-name-1
```

■ `DATA DIVISION` specification

```
01  unique-name-1.
    02  data-name-A   PIC X(8) VALUE 'CONNECT '.
    02  data-name-B   PIC X(5).
    02  FILLER        PIC X(3).
    02  data-name-Z   PIC S9(9)  COMP VALUE ZERO.
    02  data-name-C   PIC X(9).
    02  FILLER        PIC X(3).
    02  data-name-D   PIC X(9).
```

Description

`CBLDCXAT('CONNECT')` requests the XATMI communication service specified in *data-name-C* to establish the association specified in *data-name-D*.

`CBLDCXAT('CONNECT')` returns after sending an association establishment request to the remote system. It cannot be used for receiving the notification of association establishment.

`CBLDCXAT('CONNECT')` can be used only for OSI TP communication using the TP1/NET/OSI-TP-Extended.

`CBLDCXAT('CONNECT')` can be invoked either inside or outside the transaction range.

Data areas whose values are set in the UAP

■ *data-name-A*

Specify `VALUE 'CONNECT∆'` as a request code for association establishment.

■ *data-name-C*

Specify the name of the XATMI communication service that will be asked to establish an association. The specified name must be the name of the XATMI communication service definition file assigned to the `xat_invoke_server` operand in the XATMI communication service definition. Add a space after the service name.

■ *data-name-D*

Specify the name of the association to be established. The specified name must be the connection name assigned to the `-c` option of the `nettalccn` operand in the protocol-specific definition contained in the TP1/NET/OSI-TP-Extended definition. Add a space after the association name.

- *data-name-Z*

  Specify 0.

Data area whose value is returned from OpenTP1

- *data-name-B*

  A status code of 5 digits is returned.

Status codes

Status code	Explanation
`00000`	Normal termination.
`04570`	The value specified for a data name is invalid.
`04571`	The memory became insufficient.
`04572`	`CBLDCRPC('OPEN    ')` has not been invoked.
`04575`	Acquisition of address information of the XATMI communication service failed.
`04576`	The XATMI communication service is being terminated.
`04577`	The service request failed during sending to the XATMI communication service.
`04578`	The service request failed during receiving from the XATMI communication service. The probable cause is that a connection establishment request is already being executed by the XATMI communication service.
`04580`	The specified association name is not defined.
`04581`	The association has already been established.
`04582`	The association is being established.
`04583`	The association is being released.
`04584`	The association cannot be established because it is in recipient mode.

Format of received communication events

This section explains the format of the communication events that indicate the association status. Before a communication event can be received, the service group name and service name of the SPP for a communication event must be specified in the XATMI communication service definition. What type of communication event can be received depends on which operand is assigned the service group name and service name.

`xat_aso_con_event_svcname` operand:

Communication event for association establishment notification

`xat_aso_discon_event_svcname` operand:

Communication event for normal release of association

`xat_aso_failure_event_svcname` operand:

Communication event for abnormal release of association

One SPP for a communication event can handle multiple communication events if you specify the same service group name and service name to multiple operands.

Contents of communication event indicating association status

The contents of communication event indicating association status are shown below.

```
01  unique-name-1.
    02  data-name-A   PIC S9(9)  COMP.
    02  data-name-B   PIC X(9).
    02  FILLER        PIC X(3).
    02  data-name-C   PIC S9(9)  COMP.
    02  data-name-D   PIC 9(9)  COMP.
    02  data-name-E   PIC X(9).
    02  FILLER        PIC X(63).
```

■ *data-name-A*

Will be assigned the identifier of the communication event. The identifier is one of the following codes. The number enclosed in parentheses is the numerical representation (decimal) of the code.

`DCXAT_ASO_CONNECT (00000001):` Association establishment

`DCXAT_ASO_DISCONNECT (00000002):` Normal release of association

`DCXAT_ASO_FAILURE (00000003):` Abnormal release of association

- *data-name-B*

  Will be assigned the name of the association whose status is to be reported by the communication event.

- *data-name-C*

  Will be assigned the value that indicates whether the local system is the initiating or recipient for the established connection. The value assigned is one of the following codes. The number enclosed in parentheses is the numerical representation (decimal) of the code.

  `DCXAT_ASO_INIT (00000001):` The local system is the initiating.

  `DCXAT_ASO_RESP (00000002):` The local system is the recipient.

- *data-name-D*

  Will be assigned the reason code which will be returned when the association is released. The value assigned is one of the following codes. The number enclosed in parentheses is the numerical representation (decimal) of the code.

  For the normal releasing of an association, one of the following values is assigned:

  `00000001:` Releasing of an association by executing a command

  `00000005:` Releasing of an association by the XATMI

  `00000007:` Normal releasing of an association from the remote system

  `00000008:` Normal releasing of an association by the TP layer

  For the abnormal releasing of an association, one of the following values is assigned:

  `00000001:` Forced releasing of an association by executing a command

  `00000003:` Failure in a lower layer (such as a line failure and communication management failure)

  `00000005:` Forced releasing of an association by an XATMI communication service

  `00000006:` Failure in association establishment

  `00000007:` Forced releasing of an association from the remote system

- *data-name-E*

  Will be assigned the XATMI communication service name.

Chapter

6. Coding Samples

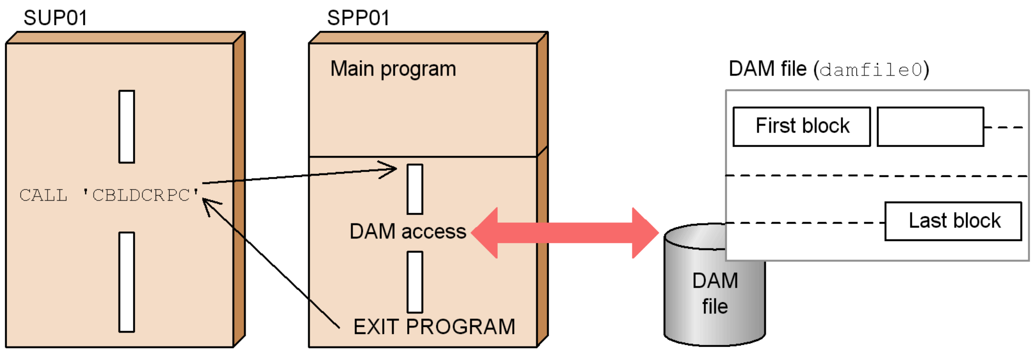
This chapter presents and explains application program (UAP) coding samples.

This chapter contains the following sections:

6.1 Coding samples for client/server UAPs (SUP, SPP DAM access)

The figure below shows an example of a client/server configuration UAP.

Figure 6-1: Client/server UAP configuration sample (DAM access)



Explanation

DAM file `damfile0` contains a control section in its first block and data records in the second and subsequent blocks. During service processing, the first block is read (`CBLDCDAM('READ')`) and is updated (`CBLDCDAM('REWT')`), then the second and subsequent blocks are directly updated using `CBLDCDAM('WRIT')`.

This section presents a coding example based on the configuration sample shown in the figure.

(1) SUP sample

The following shows a coding example for SUP.

```
      *
      **********************************************
      * SUP01                                      *
      **********************************************
      *
       IDENTIFICATION DIVISION.
      *
       PROGRAM-ID. MAIN.
      *
      **********************************************
      *  Set the data area                         *
      **********************************************
      *
       DATA DIVISION.
```

```
 WORKING-STORAGE SECTION.
 01  RPC-ARG1.
     02 REQUEST      PIC X(8) VALUE SPACE.
     02 STATUS-CODE  PIC X(5) VALUE SPACE.
     02 FILLER       PIC X(3).
     02 FLAGS        PIC S9(9) COMP VALUE ZERO.
*
 01  RPC-ARG2.
     02 REQUEST      PIC X(8) VALUE SPACE.
     02 STATUS-CODE  PIC X(5) VALUE SPACE.
     02 FILLER       PIC X(3).
     02 FLAGS        PIC S9(9) COMP VALUE ZERO.
     02 DESCRIPTOR   PIC S9(9) COMP VALUE ZERO.
     02 S-NAME       PIC X(32) VALUE SPACE.
     02 G-NAME       PIC X(32) VALUE SPACE.
*
 01  RPC-ARG3.
     02  SEND-DATA-LENG  PIC S9(9) COMP VALUE ZERO.
     02  SEND-DATA       PIC X(32) VALUE SPACE.
*
 01  RPC-ARG4.
    02  RECEIVE-DATA-LENG PIC S9(9) COMP VALUE ZERO.
     02  RECEIVE-DATA      PIC X(32) VALUE SPACE.
*
 01  ADM-ARG1.
     02  REQUEST      PIC X(8) VALUE SPACE.
     02  STATUS-CODE  PIC X(5) VALUE SPACE.
     02  FILLER       PIC X(3).
     02  FLAGS        PIC S9(9) COMP VALUE ZERO.
     02  FILLER       PIC X(3).
*
 01  TRN-ARG1.
     02  REQUEST      PIC X(8) VALUE SPACE.
     02  STATUS-CODE  PIC X(5) VALUE SPACE.
*
 PROCEDURE DIVISION.
*
***********************************************
* RPC-OPEN (start the UAP)                    *
***********************************************
*
 MOVE 'OPEN' TO REQUEST OF RPC-ARG1.
  MOVE ZERO   TO FLAGS   OF RPC-ARG1.
  CALL 'CBLDCRPC' USING RPC-ARG1.
      IF STATUS-CODE OF RPC-ARG1 NOT = '00000' THEN
        DISPLAY 'SUP01:RPC-OPEN FAILED. CODE = '
        STATUS-CODE OF RPC-ARG1
        GO TO PROG-END
 630             END-IF.
 640        *
 650        *********************************************
 660        * ADM-COMPLETE (report completion of        *
 665        * user server start processing)             *
 670        *********************************************
 680        *
 690         MOVE 'COMPLETE' TO REQUEST OF ADM-ARG1.
 700         CALL 'CBLDCADM' USING ADM-ARG1.
 710             IF STATUS-CODE OF ADM-ARG1 NOT = '00000' THEN
 720               DISPLAY 'SUP01:RPC-COMPLETE FAILED. CODE = '
 730               STATUS-CODE OF ADM-ARG1
 740               GO TO PROG-END
 750             END-IF.
 760        *
 770        *********************************************
 780        * TRN_BEGIN (start the transaction)         *
 790        *********************************************
 800        *
 810         MOVE 'BEGIN' TO REQUEST OF TRN-ARG1.
 820         CALL 'CBLDCTRN' USING TRN-ARG1.
 830             IF STATUS-CODE OF TRN-ARG1 NOT = '00000' THEN
 840               DISPLAY 'SUP01:TRN-BEGIN FAILED. CODE = '
 850               STATUS-CODE OF TRN-ARG1
 860               GO TO TRAN-END
 870             END-IF.
 880        *
 890        *********************************************
 900        *  RPC-CALL (request a remote service)      *
 910        *********************************************
 920        *
 930         MOVE 'CALL'  TO REQUEST OF RPC-ARG2.
 940         MOVE 'SPP01' TO G-NAME  OF RPC-ARG2.
 950         MOVE 'SVR01' TO S-NAME  OF RPC-ARG2.
 960        MOVE 'SUP01:DATA OpenTP1' TO SEND-DATA OF RPC-ARG3.
 970         MOVE 32 TO SEND-DATA-LENG OF RPC-ARG3.
 980         MOVE 32 TO RECEIVE-DATA-LENG  OF RPC-ARG4.
 990         CALL 'CBLDCRPC' USING RPC-ARG2 RPC-ARG3 RPC-ARG4.
1000             IF STATUS-CODE OF RPC-ARG2 NOT = '00000' THEN
1010               DISPLAY 'SUP01:RPC-CALL RETURN CODE = '
1020               STATUS-CODE OF RPC-ARG2
1030               GO TO TRAN-END
1040             END-IF.
1050         DISPLAY 'SERVICE FUNCTION RETURN = ' RECEIVE-DATA.
1060         TRAN-END.
1070        *
1080        *********************************************
1090        * TRN-UNCHAINED-COMMIT                      *
```

```
      * (commit in unchained mode)                *
      *********************************************
      *
       MOVE 'U-COMMIT' TO REQUEST OF TRN-ARG1.
       CALL 'CBLDCTRN' USING TRN-ARG1.
           IF STATUS-CODE OF TRN-ARG1 NOT = '00000' THEN
             DISPLAY 'SUP01:TRN-UNCHAINED-COMMIT FAILED.
CODE = '
             STATUS-CODE OF TRN-ARG1
           END-IF.
       PROG-END.
      *
      *********************************************
      * RPC-CLOSE (terminate the UAP)             *
      *********************************************
      *
       MOVE 'CLOSE' TO REQUEST OF RPC-ARG1.
       MOVE ZERO    TO FLAGS   OF RPC-ARG1.
       CALL 'CBLDCRPC' USING RPC-ARG1.
       DISPLAY 'SUP01:SUP PROCESS ENDED'.
       STOP RUN.
```

(2) SPP sample (main program)

The following shows a coding example for the SPP main program.

```
     *
     **********************************************
     * SPP01 main program                         *
     **********************************************
     *
      IDENTIFICATION DIVISION.
     *
      PROGRAM-ID. MAIN.
     *

********************************************************
     * Set the data area                          *

********************************************************
     *
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01  FD-ID EXTERNAL.
          10  FD-SAVE      PIC S9(9) COMP.
      01  RPC-ARG1.
          02  REQ-CODE     PIC X(8) VALUE SPACE.
          02  STATUS-CODE  PIC X(5) VALUE SPACE.
          02  FILLER       PIC X(3).
          02  FLAGS        PIC S9(9) COMP.
```

```
 01  DAM-ARG1.
     02  REQUEST      PIC X(8) VALUE SPACE.
     02  STATUS-CODE  PIC X(5) VALUE SPACE.
     02  FILLER       PIC X(3).
     02  FILE-NAME    PIC X(8).
     02  FILLER       PIC S9(9) COMP.
     02  FILLER       PIC S9(9) COMP.
     02  FILDES       PIC S9(9) COMP VALUE ZERO.
     02  FILLER       PIC X(28).
 01  DAM-ARG2.
     02  ACCESS-CODE  PIC X(4).
     02  FLAG1        PIC X(1).
     02  FILLER       PIC X(1).
     02  FILLER       PIC X(1).
     02  FILLER       PIC X(1).
     02  FLAGS        PIC S9(9) COMP VALUE ZERO.
*
 PROCEDURE DIVISION.
*
***********************************************
* RPC-OPEN (start the UAP)                    *
***********************************************
*
 MOVE 'OPEN' TO REQ-CODE OF RPC-ARG1.
 MOVE ZERO   TO FLAGS    OF RPC-ARG1.
 CALL 'CBLDCRPC' USING RPC-ARG1.
     IF STATUS-CODE OF RPC-ARG1 NOT = '00000' THEN
       DISPLAY 'SPP01:RPC-OPEN FAILED. CODE = '
       STATUS-CODE OF RPC-ARG1
       GO TO PROG-END
     END-IF.
*
***********************************************
* DAM-OPEN (open a logical file)              *
***********************************************
*
 MOVE 'DCDAMSVC' TO REQUEST     OF DAM-ARG1.
 MOVE 'damfile0' TO FILE-NAME   OF DAM-ARG1.
 MOVE 'OPEN'     TO ACCESS-CODE OF DAM-ARG2.
 MOVE 'B'        TO FLAG1       OF DAM-ARG2.
 CALL 'CBLDCDAM' USING DAM-ARG1 DAM-ARG2.
     IF STATUS-CODE OF DAM-ARG1 NOT = '00000' THEN
       DISPLAY 'SPP01:DAM-OPEN FAILED. CODE = '
       STATUS-CODE OF DAM-ARG1
       GO TO DAM-END
     END-IF.
 MOVE FILDES TO FD-SAVE.
*
```

```
**********************************************
* RPC-MAINLOOP (start the SPP service)       *
**********************************************
*
 DISPLAY 'SPP01: MAINLOOP START.'
 MOVE 'MAINLOOP' TO REQ-CODE OF RPC-ARG1.
 MOVE ZERO       TO FLAGS    OF RPC-ARG1.
 CALL 'CBLDCRSV' USING RPC-ARG1.
     IF STATUS-CODE OF RPC-ARG1 NOT = '00000' THEN
       DISPLAY 'SPP01:RPC-MAINLOOP FAILED. CODE ='
       STATUS-CODE OF RPC-ARG1
     END-IF.
 DAM-END.
*
**********************************************
* DAM-CLOSE (close the logical file)         *
**********************************************
*
 MOVE 'damfile0' TO FILE-NAME   OF DAM-ARG1.
 MOVE FD-SAVE    TO FILDES      OF DAM-ARG1.
 MOVE 'CLOS'     TO ACCESS-CODE OF DAM-ARG2.
 CALL 'CBLDCDAM' USING DAM-ARG1 DAM-ARG2.
     IF STATUS-CODE OF DAM-ARG1 NOT = '00000' THEN
       DISPLAY 'SPP01:DAM-CLOSE FAILED. CODE = '
       STATUS-CODE OF DAM-ARG1
     END-IF.
 PROG-END.
*
**********************************************
* RPC-CLOSE (terminate the UAP)               *
**********************************************
*
 MOVE 'CLOSE' TO REQ-CODE OF RPC-ARG1.
 MOVE ZERO    TO FLAGS    OF RPC-ARG1.
 CALL 'CBLDCRPC' USING RPC-ARG1.
*
**********************************************
* Terminate processing                       *
**********************************************
*
 DISPLAY 'SPP01:Good-by!'
 STOP RUN.
```

(3) SPP sample (service program)

The following shows a coding example for the SPP service program.

```
 *
 **********************************************
* SPP service program  SVR01                        *
```

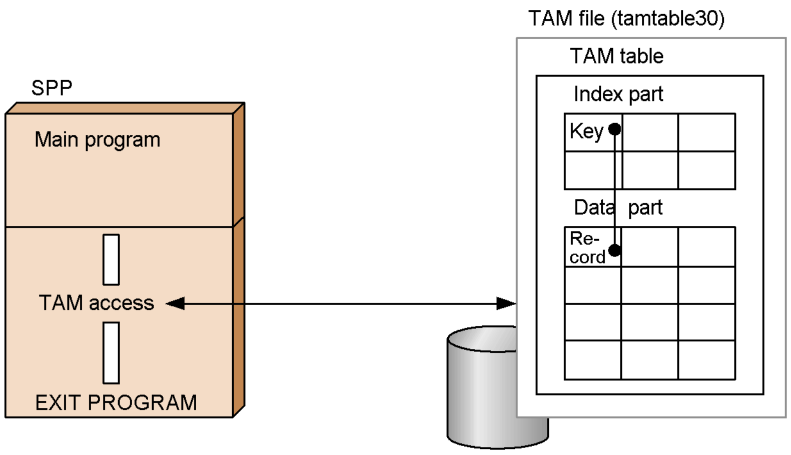
```
**********************************************
*
 IDENTIFICATION DIVISION.
*
 PROGRAM-ID. SVR01.
*
**********************************************
* Set the data area                          *
**********************************************
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  FD-ID EXTERNAL.
     10  FD-SAVE      PIC S9(9) COMP.
 01  DAM-ARG1.
     02  REQUEST      PIC X(8) VALUE SPACE.
     02  STATUS-CODE  PIC X(5) VALUE SPACE.
     02  FILLER       PIC X(3).
     02  FILE-NAME    PIC X(8).
     02  KEY-NO       PIC S9(9) COMP VALUE ZERO.
     02  BUFFER-LEN   PIC S9(9) COMP VALUE ZERO.
     02  FILDES       PIC S9(9) COMP VALUE ZERO.
     02  FILLER       PIC X(28).
 01  DAM-ARG2.
     02  ACCESS-CODE  PIC X(4).
     02  FLAG1        PIC X(1).
     02  FLAG2        PIC X(1).
     02  FLAG3        PIC X(1).
     02  FLAG4        PIC X(1).
     02  FLAGS        PIC S9(9) COMP VALUE ZERO.
     02  DAMKEY.
         03  FIRST-BLOCK-NO PIC S9(9) COMP.
         03  LAST-BLOCK-NO  PIC S9(9) COMP.
*
 01  CNTL-BUFFER.
     02  W-COUNT      PIC S9(9) COMP.
     02  RWT-DATA     PIC X(18) VALUE SPACE.
     02  FILLER       PIC X(483) VALUE SPACE.
*
 01  W-BUFFER.
     02  FILLER       PIC X(504).
*
  LINKAGE SECTION.
  77  IN-DATA      PIC X(32).
  77  IN-LENG      PIC S9(9) COMP.
  77  OUT-DATA     PIC X(32).
  77  OUT-LENG     PIC S9(9) COMP.
*
```

```
  PROCEDURE DIVISION USING IN-DATA IN-LENG OUT-DATA OUT-LENG.
  SVR01 SECTION.
  DISPLAY  'SVR01:PROCEDURE START' .
 *
 *********************************************
 * DAM_READ(read logical file blocks)        *
 *********************************************
 *
  MOVE 'DCDAMSVC' TO REQUEST     OF DAM-ARG1.
  MOVE 'damfile0' TO FILE-NAME   OF DAM-ARG1.
  MOVE 1          TO KEY-NO      OF DAM-ARG1.
  MOVE 504        TO BUFFER-LEN  OF DAM-ARG1.
  MOVE FD-SAVE    TO FILDES      OF DAM-ARG1.
  MOVE 'READ'     TO ACCESS-CODE OF DAM-ARG2.
  MOVE 'M'        TO FLAG1       OF DAM-ARG2.
  MOVE SPACE      TO FLAG2       OF DAM-ARG2.
  MOVE 0          TO FIRST-BLOCK-NO OF DAMKEY.
  MOVE 0          TO LAST-BLOCK-NO  OF DAMKEY.
CALL 'CBLDCDAM' USING DAM-ARG1 DAM-ARG2 CNTL-BUFFER.
      IF STATUS-CODE OF DAM-ARG1 NOT = '00000' THEN
         DISPLAY 'SVR01:DAM-READ FAILED. CODE ='
         STATUS-CODE OF DAM-ARG1
         MOVE 'SVR01: DAM READ FAILED' TO OUT-DATA
         MOVE 25 TO OUT-LENG
         GO TO PROG-END
      END-IF.
 *
 *********************************************
 * DAM_WRITE (write to logical file blocks)  *
 *********************************************
 *
  DAM-WRITE.
  ADD 1 TO W-COUNT OF CNTL-BUFFER.
  MOVE 'DCDAMSVC' TO REQUEST     OF DAM-ARG1.
  MOVE 'damfile0' TO FILE-NAME   OF DAM-ARG1.
  MOVE 1          TO KEY-NO      OF DAM-ARG1.
  MOVE 504        TO BUFFER-LEN  OF DAM-ARG1.
  MOVE FD-SAVE    TO FILDES      OF DAM-ARG1.
  MOVE 'WRIT'     TO ACCESS-CODE OF DAM-ARG2.
  MOVE W-COUNT OF CNTL-BUFFER TO FIRST-BLOCK-NO OF DAMKEY.
  MOVE 0          TO LAST-BLOCK-NO OF DAMKEY.
  MOVE IN-DATA TO W-BUFFER.
  CALL 'CBLDCDAM' USING DAM-ARG1 DAM-ARG2 W-BUFFER.
      IF STATUS-CODE OF DAM-ARG1 NOT = '00000' THEN
        IF STATUS-CODE OF DAM-ARG1 = '01606' THEN
          MOVE 0 TO W-COUNT OF CNTL-BUFFER
          GO TO DAM-WRITE
        END-IF
        DISPLAY 'SVR01:DAM-WRITE FAILED. CODE = '
        STATUS-CODE OF DAM-ARG1
        MOVE 'SVR01:DAM WRITE FAILED' TO OUT-DATA
        MOVE 26 TO OUT-LENG
        GO TO PROG-END
      END-IF.
 *
 *********************************************
 * DAM_REWRITE (update logical file blocks)  *
 *********************************************
 *
  MOVE 'DCDAMSVC' TO REQUEST     OF DAM-ARG1.
  MOVE 'damfile0' TO FILE-NAME   OF DAM-ARG1.
  MOVE 1          TO KEY-NO      OF DAM-ARG1.
  MOVE 504        TO BUFFER-LEN  OF DAM-ARG1.
  MOVE FD-SAVE    TO FILDES      OF DAM-ARG1.
  MOVE 'REWT'     TO ACCESS-CODE OF DAM-ARG2.
  MOVE 'U'        TO FLAG1       OF DAM-ARG2.
  MOVE 0         TO FIRST-BLOCK-NO OF DAMKEY.
  MOVE 0         TO LAST-BLOCK-NO  OF DAMKEY.
 MOVE 'REWRITE COMPLETE' TO RWT-DATA OF CNTL-BUFFER.
CALL 'CBLDCDAM' USING DAM-ARG1 DAM-ARG2 CNTL-BUFFER.
      IF STATUS-CODE OF DAM-ARG1 NOT = '00000' THEN
        DISPLAY 'SVR01:DAM-REWRITE FAILED. CODE = '
        STATUS-CODE OF DAM-ARG1
        MOVE 'SVR01:DAM REWRITE FAILED' TO OUT-DATA
        MOVE 28 TO OUT-LENG
        GO TO PROG-END
      END-IF.
  MOVE 'SVR01:PROCESS COMPLETE' TO OUT-DATA.
  MOVE 26 TO OUT-LENG.
  PROG-END.
  DISPLAY 'SVR01:Good-By!!'.
  END PROGRAM SVR01.
```

6.2 Coding samples for client/server UAPs (SPP TAM access)

The figure below shows an example of a client/server configuration UAP. This section presents only an SPP coding sample. This example assumes that the same SUP as in *6.1 Coding samples for client/server configuration UAPs (SUP, SPP DAM access)* requests this SPP for service.

Figure 6-2: Client/server UAP configuration sample (TAM access)



This section presents a coding example based on the configuration sample shown in the figure.

(1) SPP sample (main program)

The following shows a coding example for the SPP main program.

```
      *
      *********************************************
      *  SPP01 main program                       *
      *********************************************
      *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. MAIN.
      *
      *********************************************
      *  Set the data area                        *
      *********************************************
      *
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 RPC-ARG.
```

```
    02 REQ-CODE       PIC  X(8)    VALUE SPACE.
    02 STATUS-CODE    PIC  X(5)    VALUE SPACE.
    02 FILLER          PIC  X(3).
    02 FLAGS          PIC S9(9)   COMP.
 PROCEDURE DIVISION.
*
*********************************************
*  RPC-OPEN (start the UAP)                 *
*********************************************
*
     MOVE 'OPEN' TO REQ-CODE OF RPC-ARG.
     MOVE ZERO   TO FLAGS  OF RPC-ARG.
     CALL 'CBLDCRPC' USING RPC-ARG.
     IF STATUS-CODE OF RPC-ARG NOT = '00000' THEN
         DISPLAY 'SPP01 : RPC-OPEN FAILED. CODE = '
                 STATUS-CODE OF RPC-ARG
         GO TO PROG-END
     END-IF.
*
*********************************************
*  RPC-MAINLOOP (start the SPP service)     *
*********************************************
*
     MOVE 'MAINLOOP' TO REQ-CODE OF RPC-ARG.
     MOVE ZERO     TO FLAGS    OF RPC-ARG.
     CALL 'CBLDCRSV' USING RPC-ARG.
     IF STATUS-CODE OF RPC-ARG NOT = '00000' THEN
     DISPLAY ' SPP01 : RPC-MAINLOOP FAILED. CODE = '
                 STATUS-CODE OF RPC-ARG
     END-IF.
*
*********************************************
*  RPC-CLOSE (terminate the UAP)            *
*********************************************
*
     MOVE 'CLOSE   ' TO REQ-CODE OF RPC-ARG.
     MOVE ZERO       TO FLAGS    OF RPC-ARG.
     CALL 'CBLDCRPC' USING RPC-ARG.
 PROG-END.
*
*********************************************
*  Terminate processing                     *
*********************************************
*
     DISPLAY ' SPP01 : GooD-by!' .
     STOP RUN.
```

(2) SPP sample (service program)

The following shows a coding example for the SPP service program.

```
*
***************************************************
*  SPP service program  SVR01                     *
***************************************************
*
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SVR01.
*
***************************************************
*  Set the data area                              *
***************************************************
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 TAM-ARG1.
    02 REQ-CODE       PIC  X(4)    VALUE SPACE.
    02 STATUS-CODE    PIC  X(5)    VALUE SPACE.
    02 FILLER          PIC  X(3).
    02 TABLE-NAME     PIC  X(32)   VALUE SPACE.
    02 FILLER         PIC  X(68).
    02 BUF-SIZE       PIC S9(4)   COMP VALUE ZERO.
    02 FILLER         PIC  X(398).
 01 READ-ARG1.
    02 DML-KIND       PIC  X(4)    VALUE SPACE.
    02 LOK-KIND       PIC  X(1)    VALUE SPACE.
    02 FILLER         PIC  X(3).
 01 WRITE-ARG1.
    02 DML-KIND       PIC  X(4)    VALUE SPACE.
    02 FILLER         PIC  X(4).
 01 KEY-DATA1         PIC X(10)   VALUE
X'00000000000000000001'.
 01 KEY-DATA2         PIC X(10)   VALUE
X'00000000000000000002'.
 01 KEY-DATA4         PIC X(10)   VALUE
X'00000000000000000004'.
 01 KEY-ARG.
    02 KEYNAME        PIC X(10)   VALUE SPACE.
 01 W-BUFFER.
    02 KEYNAME        PIC X(10)   VALUE SPACE.
    02 DATAREA        PIC  X(118)  VALUE SPACE.
 LINKAGE SECTION.
    77 IN-DATA        PIC X(118).
    77 IN-LENG        PIC S9(9)   COMP.
    77 OUT-DATA       PIC X(32).
    77 OUT-LENG       PIC S9(9)   COMP.
```

```
        PROCEDURE DIVISION USING IN-DATA IN-LENG OUT-DATA
OUT-LENG.
            DISPLAY ' SVR01:PROCEDURE START' .
       *
       ***************************************************
       *  TAM_READ (read the first record from the      *
       *  TAM table)                                     *
       ***************************************************
       *
            MOVE 'tamtable30' TO TABLE-NAME  OF TAM-ARG1.
            MOVE 128         TO BUF-SIZE     OF TAM-ARG1.
            MOVE 'FCHU'      TO DML-KIND     OF READ-ARG1.
            MOVE KEY-DATA1   TO KEY-ARG.
             CALL 'CBLDCTAM' USING TAM-ARG1 READ-ARG1 KEY-ARG
W-BUFFER.
             IF STATUS-CODE OF TAM-ARG1 NOT = '00000' THEN
                 DISPLAY 'SVR01:TAM-READ FAILED. CODE = '
                         STATUS-CODE OF TAM-ARG1
                 MOVE 'SVR01: TAM READ FAILED' TO OUT-DATA
                 MOVE 22 TO OUT-LENG
                 GO TO PROG-END
             END-IF.
        *
        *************************************************
        * TAM_REWRITE (update the first record of       *
        * TAM table on the assumption of entry)        *
        *************************************************
        *
             MOVE 'MFY '       TO DML-KIND    OF WRITE-ARG1.
             MOVE IN-DATA      TO DATAREA     OF W-BUFFER.
             CALL 'CBLDCTAM' USING TAM-ARG1 WRITE-ARG1 KEY-ARG
W-BUFFER.
             IF STATUS-CODE OF TAM-ARG1 NOT = '00000' THEN
                 DISPLAY 'SVR01:TAM-REWRITE FAILED. CODE = '
                         STATUS-CODE OF TAM-ARG1
                 MOVE 'SVR01: TAM REWRITE FAILED' TO OUT-DATA
                 MOVE 25 TO OUT-LENG
                 GO TO PROG-END
             END-IF.
        *
        *************************************************
        *  TAM_WRITE (update the second record of       *
        * TAM table)                                     *
        *************************************************
        *
             MOVE 'MFY '      TO DML-KIND OF WRITE-ARG1.
             MOVE KEY-DATA2   TO KEY-ARG.
             MOVE KEY-DATA2   TO KEYNAME OF W-BUFFER.
```

```
     MOVE IN-DATA     TO DATAREA OF W-BUFFER.
     CALL 'CBLDCTAM' USING TAM-ARG1 WRITE-ARG1 KEY-ARG
W-BUFFER.
     IF STATUS-CODE OF TAM-ARG1 NOT = '00000' THEN
         DISPLAY 'SVR01:TAM-WRITE FAILED. CODE = '
                 STATUS-CODE OF TAM-ARG1
         MOVE 'SVR01: TAM WRITE FAILED' TO OUT-DATA
         MOVE 23 TO OUT-LENG
         GO TO PROG-END
     END-IF.
*
*************************************************
*  TAM-DELETE(delete the fourth record of       *
* the TAM table)                                *
*************************************************
*
     MOVE 'ERS '    TO DML-KIND OF WRITE-ARG1.
     MOVE KEY-DATA4 TO KEY-ARG.
     CALL 'CBLDCTAM' USING TAM-ARG1 WRITE-ARG1 KEY-ARG
W-BUFFER.
     IF STATUS-CODE OF TAM-ARG1 NOT = '00000' THEN
         DISPLAY 'SVR01:TAM-DELETE FAILED. CODE = '
                 STATUS-CODE OF TAM-ARG1
         MOVE 'SVR01: TAM DELETE FAILED' TO OUT-DATA
         MOVE 24 TO OUT-LENG
     END-IF.
 PROG-END.
*
*************************************************
*  Terminate processing                         *
*************************************************
*
     DISPLAY 'SVR01:GooD-by!'.
     EXIT PROGRAM.
```

6.3 Coding samples for message exchange UAPs (MHP)

The figure below shows an example of a message exchange UAP.

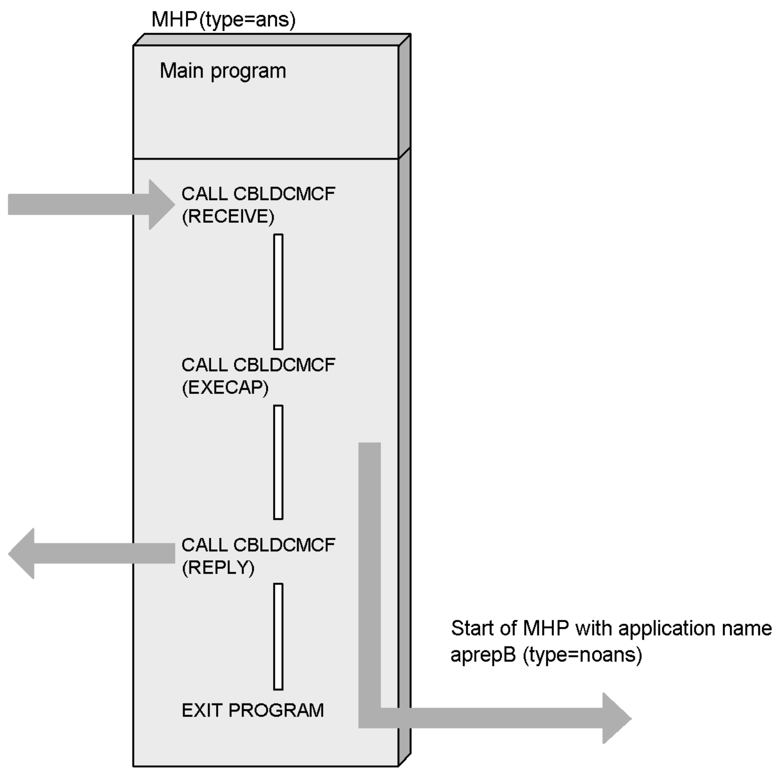


Figure 6-3: Message exchange UAP configuration sample (MHP)

This section presents a coding example based on the configuration sample shown in the figure.

(1) MHP sample (main program)

The following shows a coding example for the MHP main program.

```
  10       *
  20       ***********************************************
  30       * MHP main program                            *
  40       ***********************************************
  50       *
  60        IDENTIFICATION DIVISION.
  70
  80        PROGRAM-ID. CBMAIN.
```

```

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
**********************************************
*     Work variable                     *
**********************************************
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
**********************************************
*  RPC-OPEN  data area             *
**********************************************
*
 01  ROPEN-PARM1.
   02  ROPEN-NAME         PIC X(8) VALUE 'OPEN    '.
   02  ROPEN-STATUS       PIC X(5).
   02  FILLER             PIC X(3).
   02  RO-FLG             PIC S9(9) COMP VALUE ZERO.
*
**********************************************
*  MCF-OPEN  data area                 *
**********************************************
*
 01  MOPEN-PARM1.
   02  MOPEN-NAME         PIC X(8) VALUE 'OPEN    '.
   02  MOPEN-STATUS       PIC X(5).
   02  FILLER             PIC X(3).
   02  MO-FLG1            PIC S9(9) COMP VALUE ZERO.
   02  MO-RSV             PIC X(12) VALUE LOW-VALUE.
*
**********************************************
*  MCF-MAINLOOP data area        *
**********************************************
*
 01  MAIN-PARM1.
   02  MAIN-NAME         PIC X(8) VALUE 'MAINLOOP'.
   02  MAIN-STATUS       PIC X(5).
   02  FILLER            PIC X(3).
   02  M-RSV             PIC X(16) VALUE LOW-VALUE.
*
**********************************************
*  MCF-CLOSE data area                *
**********************************************
 *
  01  MCLSE-PARM1.
   02  MCLSE-NAME         PIC X(8) VALUE 'CLOSE   '.
   02  MCLSE-STATUS       PIC X(5).
   02  MFILLER            PIC X(3).
  02  MC-FLG1            PIC S9(9) COMP VALUE ZERO.
  02  MC-RSV             PIC X(12) VALUE LOW-VALUE.
*
*********************************************
*  RPC-CLOSE data area              *
*********************************************
*
 01  RCLSE-PARM1.
  02  RCLSE-NAME         PIC X(8) VALUE 'CLOSE   '.
   02  RCLSE-STATUS       PIC X(5).
   02  FILLER             PIC X(3).
  02  RC-FLG             PIC S9(9) COMP VALUE ZERO.
*
 PROCEDURE DIVISION.
*
*********************************************
*  RPC-OPEN (start the UAP)                 *
*********************************************
*
   CALL 'CBLDCRPC' USING ROPEN-PARM1.
   IF ROPEN-STATUS IS NOT EQUAL TO '00000'
      GO TO RCLOS.
*
*********************************************
*  MCF-OPEN (open the MCF environment)      *
*********************************************
*
   CALL 'CBLDCMCF' USING MOPEN-PARM1.
   IF MOPEN-STATUS IS NOT EQUAL TO '00000'
      GO TO RCLOS.
*
*********************************************
*  MCF-MAINLOOP (start the MHP service)    *
*********************************************
*
   CALL 'CBLDCMCF' USING MAIN-PARM1.
*
*********************************************
*  MCF-CLOSE (close the MCF environment)    *
*********************************************
*
   CALL 'CBLDCMCF' USING MCLSE-PARM1.
*
*********************************************
*  RPC-CLOSE (terminate the UAP)            *
*********************************************
```

```
*
 RCLOS.
   CALL 'CBLDCRPC' USING RCLSE-PARM1.
*
***********************************************
*      Terminate processing                   *
***********************************************
*
   STOP RUN.
```

(2) MHP sample (service program)

The following shows a coding example for the MHP service program.

```
*
**********************************************
*  MHP service program                       *
**********************************************
*
 IDENTIFICATION DIVISION.

 PROGRAM-ID. SVRA.

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
**********************************************
*  Work variable                              *
**********************************************
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
**********************************************
*  MCF-RECEIVE data area             *
**********************************************
*
 01  RECV-PARM1.
   02   RECV-NAME         PIC X(8) VALUE 'RECEIVE '.
   02   RECV-STATUS       PIC X(5).
   02   FILLER            PIC X(3).
   02   FRST-ID           PIC X(4) VALUE 'FRST'
   02   RE-RSV1           PIC X(4) VALUE SPACE.
   02   DATE-ID           PIC 9(8).
   02   TIME-ID           PIC 9(8).
   02   RE-LENG           PIC 9(9) COMP VALUE 1024.
   02   RE-RSV2           PIC X(4) VALUE SPACE.
   02   RE-RSV3           PIC X(4) VALUE SPACE.
   02   RE-RSV4           PIC X(4) VALUE SPACE.
   02   RE-RSV5           PIC X(4) VALUE SPACE.
```

```
            02   RE-RSV6           PIC X(8) VALUE SPACE.
            02   RE-RSV7           PIC X(4) VALUE SPACE.
            02   RE-RSV8           PIC X(8) VALUE SPACE.
            02   RE-RSV9           PIC X(4) VALUE SPACE.
            02   RE-RSV10          PIC 9(9) COMP VALUE ZERO.
            02   RE-RSV11          PIC 9(9) COMP VALUE ZERO.
            02   RE-RSV12          PIC X(1) VALUE SPACE.
            02   RE-RSV13          PIC X(1) VALUE '1'.
            02   RE-RSV14          PIC X(14) VALUE LOW-VALUE.
        01  RECV-PARM2.
            02   RE-RSV15          PIC X(4) VALUE SPACE.
            02   TERM-ID           PIC X(8).
            02   RE-RSV16          PIC X(8) VALUE SPACE.
            02   RE-RSV17          PIC X(8) VALUE SPACE.
            02   RE-RSV18          PIC X(28) VALUE LOW-VALUE.
        01  RECV-PARM3.
            02   RE-DATALENG       PIC 9(9) COMP.
            02   RE-RSV19          PIC X(8).
            02   RE-DATA           PIC X(1024).
       *
       **********************************************
       *  MCF-EXECAP data area             *
       **********************************************
       *
        01   EXEC-PARM1.
            02   EXEC-NAME         PIC X(8) VALUE 'EXECAP  '.
            02   EXEC-STATUS       PIC X(5).
            02   FILLER            PIC X(3).
            02   EX-RSV1           PIC X(4) VALUE SPACE.
            02   EX-RSV2           PIC X(4) VALUE SPACE.
            02   EX-RSV3           PIC 9(8).
            02   EX-RSV4           PIC 9(8).
            02   EX-RSV5           PIC 9(9) COMP VALUE ZERO.
            02   EX-EMI            PIC X(4) VALUE 'EMI '.
            02   EX-RSV6           PIC X(4) VALUE SPACE.
            02   EX-RSV7           PIC X(4) VALUE SPACE.
            02   EX-RSV8           PIC X(4) VALUE SPACE.
            02   EX-TIME           PIC X(8) VALUE '00000000'.
            02   EX-RSV9           PIC X(4) VALUE SPACE.
            02   EX-RSV10          PIC X(8) VALUE 'aprepB  '.
            02   EX-EXEC           PIC X(4) VALUE 'JUST'.
            02   EX-RSV11          PIC 9(9) COMP VALUE ZERO.
            02   EX-RSV12          PIC 9(9) COMP VALUE ZERO.
            02   EX-RSV13          PIC X(1) VALUE SPACE.
            02   EX-RSV14          PIC X(1) VALUE '1'.
            02   EX-RSV15          PIC X(14) VALUE LOW-VALUE.
        01   EXEC-PARM2.
            02   EX-RSV16          PIC X(4) VALUE SPACE.
```

```
             02   EX-RSV17          PIC X(8) VALUE SPACE.
             02   EX-RSV18          PIC X(8) VALUE SPACE.
             02   EX-RSV19          PIC X(8) VALUE SPACE.
             02   EX-RSV20          PIC X(28) VALUE LOW-VALUE.
           01   EXEC-PARM3.
             02   EX-DATALENG       PIC 9(9) COMP VALUE 16.
             02   EX-RSV21          PIC X(8).
             02   EX-DATA           PIC X(16) VALUE 'SVRA EXECAP DATA'.
          *
          **********************************************
          *  MCF-REPLY data area                       *
          **********************************************
          *
           01  RPLY-PARM1.
             02   RPLY-NAME         PIC X(8) VALUE 'REPLY  '.
             02   RPLY-STATUS       PIC X(5).
             02   FILLER            PIC X(3).
             02   RP-RSV1           PIC X(4) VALUE SPACE.
             02   RP-RSV2           PIC X(4) VALUE SPACE.
             02   RP-RSV3           PIC 9(8).
             02   RP-RSV4           PIC 9(8).
             02   RP-RSV5           PIC 9(9) COMP VALUE ZERO.
             02   RP-EMI            PIC X(4) VALUE'EMI '.
             02   RP-RSV6           PIC X(4) VALUE SPACE.
             02   RP-RSV7           PIC X(4) VALUE SPACE.
             02   RP-RSV8           PIC X(4) VALUE SPACE.
             02   RP-RSV9           PIC X(8) VALUE SPACE.
             02   RP-RSV10          PIC X(4) VALUE SPACE.
             02   RP-RSV11          PIC X(8) VALUE SPACE.
             02   RP-RSV12          PIC X(4) VALUE SPACE.
             02   RP-RSV13          PIC 9(9) COMP VALUE ZERO.
             02   RP-RSV14          PIC 9(9) COMP VALUE ZERO.
             02   RP-RSV15          PIC X(1) VALUE SPACE.
             02   RP-RSV16          PIC X(1) VALUE '1'.
             02   RP-RSV17          PIC X(14) VALUE LOW-VALUE.
           01  RPLY-PARM2.
             02   RP-RSV18          PIC X(4) VALUE SPACE.
             02   RP-RSV19          PIC X(8) VALUE SPACE.
             02   RP-RSV20          PIC X(8) VALUE SPACE.
             02   RP-RSV21          PIC X(8) VALUE SPACE.
             02   RP-RSV22          PIC X(28) VALUE LOW-VALUE.
           01  RPLY-PARM3.
             02   RP-DATALENG       PIC 9(9) COMP VALUE 16.
             02   RP-RSV23          PIC X(8).
             02   RP-DATA           PIC X(16) VALUE 'SVRA REPLY DATA1'.
          *
```

```
**********************************************
*  MCF-ROLLBACK data area             *
**********************************************
*
 01   RBK-PARM1.
   02   RBK-NAME          PIC X(8) VALUE 'ROLLBACK'.
   02   RBK-STATUS        PIC X(5).
   02   FILLER            PIC X(3).
   02   RBK-ACTION        PIC X(4) VALUE 'NRTN'.
   02   RBK-RSV1          PIC X(12) VALUE LOW-VALUE.

 PROCEDURE DIVISION.
*
**********************************************
*  MCF-RECEIVE (receive messages)            *
**********************************************
*
 CALL 'CBLDCMCF' USING RECV-PARM1 RECV-PARM2 RECV-PARM3.
 IF RECV-STATUS IS NOT EQUAL TO '00000'
*
**********************************************
*  MCF-ROLLBACK (error processing)           *
**********************************************
*
 CALL 'CBLDCMCF' USING RBK-PARM1.
*
**********************************************
*  MCF-EXECAP (start the application program)*
**********************************************
*
 CALL 'CBLDCMCF' USING EXEC-PARM1 EXEC-PARM2 EXEC-PARM3.
 IF EXEC-STATUS IS NOT EQUAL TO '00000'
*
**********************************************
*  MCF-ROLLBACK (error processing)           *
**********************************************
*
 CALL 'CBLDCMCF' USING RBK-PARM1.
*
**********************************************
*  MCF-REPLY (send a response message)       *
**********************************************
*
 CALL 'CBLDCMCF' USING RPLY-PARM1 RPLY-PARM2 RPLY-PARM3.
 IF RPLY-STATUS IS NOT EQUAL TO '00000'
```

```
*
**********************************************
*  MCF-ROLLBACK (error processing)           *
**********************************************
*
 CALL 'CBLDCMCF' USING RBK-PARM1.
*
**********************************************
*  Terminate processing                      *
**********************************************
*
 EXIT PROGRAM.
```

(3) MHP sample (service program in DML)

The following shows a coding example for the MHP service program written in the data manipulation language (DML).

```
*
**********************************************
*    MHP service program            *
**********************************************
*
 IDENTIFICATION DIVISION.

 PROGRAM-ID. SVRA.

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
**********************************************
*    Work variable                        *
**********************************************
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
**********************************************
*    Area for receiving messages             *
**********************************************
*
  01  RECV-AREA.
    02  RE-DATALENG       PIC 9(4) COMP VALUE 1028.
    02  RE-RSV1           PIC X(2).
    02  RE-DATA           PIC X(1024).
*
**********************************************
*    Application start message area *
**********************************************
```

```
*
  01  SEND-PRO-AREA.
    02  PRO-DATALENG       PIC 9(4) COMP VALUE 20.
    02  PRO-RSV1           PIC X(2).
  02  PRO-DATA         PIC X(16) VALUE 'SVRA EXECAP DATA'.
*
**********************************************
*    Response message transmission area          *
**********************************************
*
  01  SEND-IO-AREA.
    02  IO-DATALENG       PIC 9(4) COMP VALUE 20.
    02  IO-RSV1           PIC X(2).
   02  IO-DATA          PIC X(16) VALUE 'SVRA REPLY DATA1'.
*
**********************************************
*    Communication description entry              *
**********************************************
*
 COMMUNICATION SECTION.
*
**********************************************
*    Receive messages                        *
**********************************************
 *
  CD RECV-INF
    FOR INPUT
    STATUS KEY IS        RE-STATUS
    SYMBOLIC TERMINAL IS RE-TERMNAM
    MESSAGE DATE IS      RE-DATE
    MESSAGE TIME IS      RE-TIME.
 *
 *********************************************
 *    Start the application program          *
 *********************************************
 *
  CD SEND-PRO
    FOR OUTPUT PROGRAM
    STATUS KEY IS        SE-STATUS-PRO
    SYMBOLIC TERMINAL IS SE-TERMNAM-PRO.
 *
 *********************************************
 *    Send response messages                 *
 *********************************************
 *
  CD SEND-IO
```

```
   FOR I-O
   STATUS KEY IS        SE-STATUS-IO
   SYNCHRONOUS MODE IS  ASYNC.

 PROCEDURE DIVISION.

*
**********************************************
*    Receive messages                        *
**********************************************
*
   RECEIVE RECV-INF
           FIRST SEGMENT
           INTO RECV-AREA.
   IF RE-STATUS IS NOT EQUAL '00000'
*
**********************************************
*    Partial recovery                        *
**********************************************
*
   ROLLBACK WITH STOPPING.
*
**********************************************
*    Start the application program           *
**********************************************
*
   MOVE 'aprepB  ' TO SE-TERMNAM-PRO
   SEND SEND-PRO
        FROM SEND-PRO-AREA
        WITH EMI.
   IF SE-STATUS-PRO IS NOT EQUAL '00000'
*
**********************************************
*    Partial recovery                        *
**********************************************
*
   ROLLBACK WITH STOPPING.
*
**********************************************
*    Send response messages                  *
**********************************************
*
   SEND SEND-IO
        FROM SEND-IO-AREA
        WITH EMI.
   IF SE-STATUS-IO IS NOT EQUAL '00000'
*
**********************************************
```

```
*    Partial recovery                       *
*********************************************
*
   ROLLBACK WITH STOPPING.
*
*********************************************
*    Terminate processing                   *
*********************************************
*
   EXIT PROGRAM.
```

6.4 Coding samples for X/Open-compliant UAPs

6.4.1 XATMI interface samples

(1) Request/response service paradigm sample

(a) Outline of processing

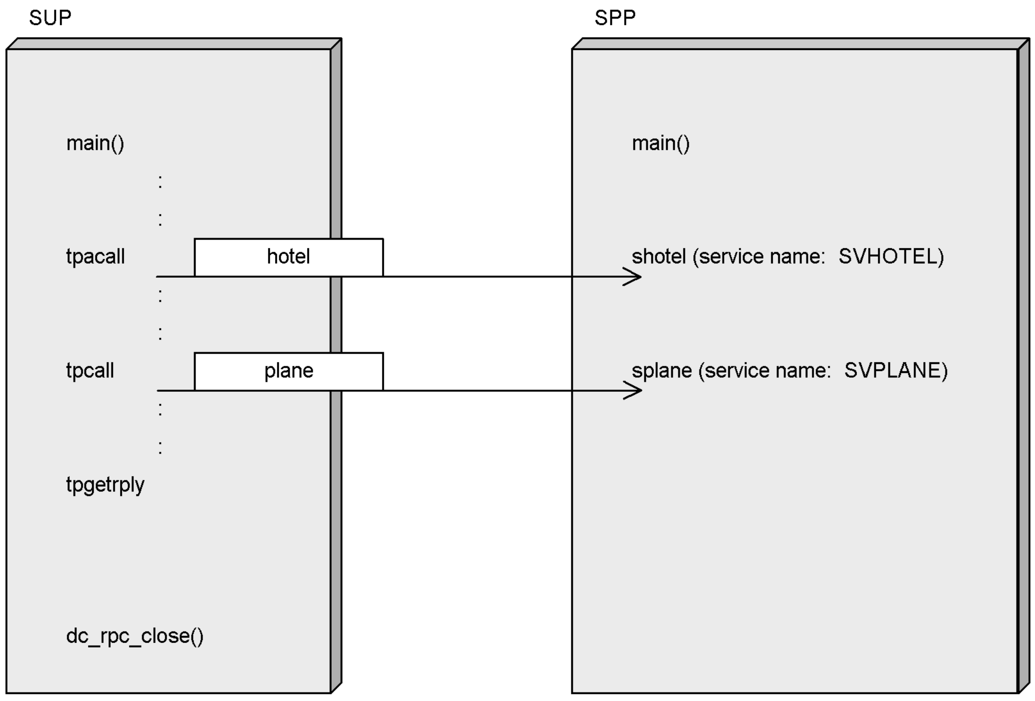
The processing of the sample here is outlined below.

A service for checking hotel room availability and a service for checking airplane seat availability are called from the SUP. The first service receives responses asynchronously, whereas the second service receives responses synchronously.

(b) UAP configuration

The following figure shows the configuration of the sample UAP.

Figure 6-4: Communication of request/response services receiving responses synchronously



(c) Typed records used

The following shows the structure of typed buffers used for communication.

```
HOTEL.cbl
           05 RDATE            PIC S9(9) COMP-5.
           05 PLACE            PIC X(128).
           05 HNAME            PIC X(128).
           05 RSTATUS          PIC S9(9) COMP-5.

PLANE.cbl
           05 RDATE            PIC S9(9) COMP-5.
           05 DEST             PIC X(128).
           05 DEPARTURE        PIC S9(9) COMP-5.
           05 RSTATUS          PIC S9(9) COMP-5.
```

(d) SUP sample

- XATMI interface definition sample

  The following shows the XATMI interface definition of the SUP used in the example of request/response service.

```
/* Example of XATMI interface definition of SUP  *
 * (cvsupcb.def file)                            */
called_servers = { "cvsppcb.def" };
```

- SUP coding sample

  The following shows a coding example for the SUP used in the example of request/response service.

```
      *
      ************************************************
      * Example of SUP (rrsup.cbl file)              *
      ************************************************
      *
       IDENTIFICATION DIVISION.
      *
       PROGRAM-ID.  MAIN.
      *
      ************************************************
      * Set the data area
      ************************************************
      *
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      ************************************************
      * Declare variables
      ************************************************
      ***** typed record for SVHOTEL *****
       01  HOTEL-REQ.
           COPY HOTEL.
      ***** type information for SVHOTEL *****
       01  HOTELTYPE-REC.
```

```
           COPY TPTYPE.
      ***** typed record for SVPLANE *****
       01  PLANE-REQ.
           COPY PLANE.
      ***** type information for SVPLANE *****
       01  PLANETYPE-REC.
           COPY TPTYPE.
      ***** WERRMSG *****
       01  WERRMSG-REC.
           COPY ERRMSG.
      ***** service definition for SVHOTEL *****
       01  HOTELDEF-REC.
           COPY TPSVCDEF.
      ***** service definition for SVPLANE *****
       01  PLANEDEF-REC.
           COPY TPSVCDEF.
      ***** return record *****
       01  STATUS-REC.
           COPY TPSTATUS.
      ***** working area is used for replies *****
       01  WK-AREA    PIC X(264).
      ***** redefine working area 1 *****
       01  HOTEL-REP REDEFINES WK-AREA.
           COPY HOTEL.
      ***** redefine working area 2 *****
       01  PLANE-REP REDEFINES WK-AREA.
           COPY PLANE.
      ***** redefine working area 3 *****
       01  ERRMSG-REP REDEFINES WK-AREA.
           COPY ERRMSG.
      ***** typed information  *****
       01  TYPE-REC.
           COPY TPTYPE.
      ***** others *****
       01  WSTATUS PIC S9(9) COMP-5.
      ***** dc_rpc_open *****
       01  RPC-OP-ARG.
           02  REQEST        PIC X(8) VALUE 'OPEN    '.
           02  STATUS-CODE   PIC X(5) VALUE SPACE.
           02  FILLER        PIC X(3).
           02  FLAGS         PIC S9(9) COMP VALUE ZERO.
      ***** dc_rpc_close *****
       01  RPC-CL-ARG.
           02  REQEST        PIC X(8) VALUE 'CLOSE   '.
           02  STATUS-CODE   PIC X(5) VALUE SPACE.
           02  FILLER        PIC X(3).
           02  FLAGS         PIC S9(9) COMP VALUE ZERO.
      *
```

```
 01  ADM-ARG.
     02    REQUEST     PIC X(8) VALUE 'COMPLETE'.
     02    STATUS-CODE PIC X(5) VALUE SPACE.
     02    FILLER      PIC X(3).
     02    FLAGS       PIC S9(9) COMP VALUE ZERO.
     02    FILLER      PIC X(3).
*
 01  FLAG              PIC S9(9) COMP VALUE ZERO.
*
 PROCEDURE DIVISION.
*
***********************************************
* RPC-OPEN (start the UAP)
***********************************************
*
 CALL 'CBLDCRPC' USING RPC-OP-ARG.
     IF STATUS-CODE OF RPC-OP-ARG NOT = '00000' THEN
       DISPLAY 'CLIENT: RPC-OPEN FAILED. CODE = '
                STATUS-CODE OF RPC-OP-ARG
       GO TO PROG-END
     END-IF.
*
***********************************************
* ADM-COMPLETE (report completion of user     *
* server start processing)                    *
***********************************************
*
 CALL 'CBLDCADM' USING ADM-ARG.
     IF STATUS-CODE OF ADM-ARG NOT = '00000' THEN
       DISPLAY 'CLIENT: ADM-COMPLETE FAILED. CODE = '
                STATUS-CODE OF ADM-ARG
       GO TO PROG-END
     END-IF.
*
***********************************************
* TPACALL (service request (SVHOTEL))
***********************************************
*
***** set parameters *****
*
***** set up HOTELDEF-REC *****
*
 MOVE LOW-VALUES            TO HOTELDEF-REC.
 MOVE "SVHOTEL"             TO SERVICE-NAME OF
HOTELDEF-REC.
*
***** set up HOTELTYPE-REC *****
*
```

```
 MOVE "X_COMMON"            TO REC-TYPE    OF HOTELTYPE-REC.
 MOVE "hotel"               TO SUB-TYPE    OF HOTELTYPE-REC.
 COMPUTE LEN OF HOTELTYPE-REC = FUNCTION LENGTH(HOTEL-REQ).
*
***** set up HOTEL-REQ *****
*
MOVE 940415              TO RDATE       OF HOTEL-REQ.
MOVE "SAPPRO"            TO PLACE       OF HOTEL-REQ.
MOVE "PRINCE"            TO HNAME       OF HOTEL-REQ.
MOVE 0                   TO RSTATUS     OF HOTEL-REQ.
*
***** CALL TPACALL *****
 CALL "TPACALL" USING
     HOTELDEF-REC HOTELTYPE-REC HOTEL-REQ STATUS-REC.
     IF NOT TPOK OF STATUS-REC THEN
       DISPLAY 'CLIENT: SVHOTEL SERVICE REQ WAS FAIL:ERROR = '
                TP-STATUS OF STATUS-REC
       GO TO PROG-END
     END-IF.
*
 DISPLAY 'CLIENT: SVHOTEL SERVICE REQ WAS SUCCESS '.
*
*
***********************************************
* TPCALL (service request (SVPLANE))
***********************************************
*
***** set parameters *****
*
***** set up PLANEDEF-REC *****
*
 MOVE LOW-VALUES            TO PLANEDEF-REC.
 MOVE "SVPLANE"             TO SERVICE-NAME OF PLANEDEF-REC.
*
***** set up PLANETYPE-REC *****
*
 MOVE "X_COMMON"            TO REC-TYPE    OF PLANETYPE-REC.
 MOVE "plane"               TO SUB-TYPE    OF PLANETYPE-REC.
 COMPUTE LEN OF PLANETYPE-REC = FUNCTION LENGTH(PLANE-REQ).
*
```

```
***** set up PLANE-REQ *****
*
MOVE 940415              TO RDATE          OF PLANE-REQ.
MOVE "CHITOSE"           TO DEST           OF PLANE-REQ.
MOVE 1540                TO DEPARTURE      OF PLANE-REQ.
MOVE 0                   TO RSTATUS        OF PLANE-REQ.
*
*
***** set up TYPE-REC *****
*
MOVE "X_COMMON"           TO REC-TYPE      OF TYPE-REC.
MOVE "plane"              TO SUB-TYPE      OF TYPE-REC.
 COMPUTE LEN OF TYPE-REC = FUNCTION LENGTH(WK-AREA).
*
***** CALL TPCALL *****
 CALL "TPCALL" USING PLANEDEF-REC PLANETYPE-REC
PLANE-REQ
      TYPE-REC WK-AREA STATUS-REC.
*
* FAILURE CASE
*
     IF NOT TPOK OF STATUS-REC THEN
     DISPLAY 'CLIENT: SVPLANE SERVICE REQ WAS FAILED'
       DISPLAY 'CLIENT: TPCALL WAS FAILED:ERROR='
                TP-STATUS OF STATUS-REC
       IF TPESVCFAIL OF STATUS-REC THEN
         MOVE ERRMESSAGE IN ERRMSG-REP
              TO ERRMESSAGE OF WERRMSG-REC
         DISPLAY 'CLIENT: USER  CODE = '
                  ERRMESSAGE OF WERRMSG-REC
         GO TO PROG-END
       END-IF
       GO TO PROG-END
     END-IF.
*
* SUCCESS CASE
*
 DISPLAY 'CLIENT: SVPLANE SERVICE REQ WAS SUCCESS '.
     MOVE RSTATUS IN PLANE-REP TO WSTATUS.
     IF WSTATUS = 1 THEN
       DISPLAY 'CLIENT: NO BORDING TICKET'
     ELSE
       DISPLAY 'CLIENT: A BORDING TICKET WAS FOUND'
     END-IF.
*
***********************************************
* TPGETRPLY (receive response messages)
***********************************************
```

```
*
***** set parameters *****
*
***** set up TYPE-REC *****
*
MOVE "X_COMMON"           TO REC-TYPE    OF TYPE-REC.
MOVE "hotel"              TO SUB-TYPE    OF TYPE-REC.
 COMPUTE LEN OF TYPE-REC = FUNCTION LENGTH(WK-AREA).
*
***** CALL TPGETRPLY *****
 CALL "TPGETRPLY" USING HOTELDEF-REC TYPE-REC WK-AREA
      STATUS-REC.
*
* FAILURE CASE
*
     IF NOT TPOK OF STATUS-REC THEN
     DISPLAY 'CLIENT: SVHOTEL SERVICE RSP WAS FAILED '
       DISPLAY 'CLIENT: TPGETRPLY WAS FAILED:ERROR='
                TP-STATUS OF STATUS-REC
       IF TPESVCFAIL OF STATUS-REC THEN
         MOVE ERRMESSAGE IN ERRMSG-REP
              TO ERRMESSAGE OF WERRMSG-REC
         DISPLAY 'CLIENT: USER  CODE = '
                  ERRMESSAGE OF WERRMSG-REC
         GO TO PROG-END
       END-IF
       GO TO PROG-END
     END-IF.
*
* SUCCESS CASE
*
 DISPLAY 'CLIENT: SVHOTEL SERVICE RSP WAS SUCCESS '.
     MOVE RSTATUS IN HOTEL-REP TO WSTATUS.
     IF WSTATUS = 1 THEN
       DISPLAY 'CLIENT: NO ROOM'
     ELSE
       DISPLAY 'CLIENT: A ROOM WAS FOUND'
     END-IF.
*
***********************************************
* Terminate processing
***********************************************
*
 PROG-END.
*
 DISPLAY 'CLIENT: SEE YOU LATER'
*
***********************************************
```

```
      * RPC-CLOSE (terminate the UAP)
      ***********************************************
      *
       CALL 'CBLDCRPC' USING RPC-CL-ARG.
      *
       STOP RUN.
```

- User service definition sample

  The following shows an example of user service definition of the SUP that was presented in the example of the request/response service.

```
# Example of user service definition (rrsup file)
set  module             = "rrsup"
set  receive_from       = none
set  trn_expiration_time = 180
set  trn_expiration_time_suspend = Y
```

(e) SPP sample

- XATMI interface definition sample

  The following shows an example of XATMI interface definition of the SPP that was presented in the example of the request/response service.

```
/* Example of XATMI interface definition         *
 * (rrsppcb.def file)                             */
  X_COMMON hotel {
      long   rdate;
      char   place[128];
      char   hname[128];
      long   rstatus;
  };
  X_COMMON plane {
      long   rdate;
      char   dest[128];
      long   departure;
      long   rstatus;
  };
  X_COMMON errmsg {
      char   errmessage[128];
  };
  service SHOTEL(X_COMMON hotel) ;
  service SPLANE(X_COMMON plane) ;
```

- SPP coding sample (main program)

  The following shows a coding example (main program) of the SPP that was presented in the example of the request/response service.

```
      *
      ***********************************************
```

```
 30      * Example of SPP (rrspp.cbl file)
 40      ***********************************************
 50      *
 60       IDENTIFICATION DIVISION.
 70      *
 80       PROGRAM-ID.     MAIN.
 90      *
100      ***********************************************
110      * Set the data area
120      ***********************************************
130      *
140       DATA DIVISION.
150       WORKING-STORAGE SECTION.
160       01  RPC-OP-ARG.
170           02  REQEST       PIC X(8) VALUE 'OPEN    '.
180           02  STATUS-CODE  PIC X(5) VALUE SPACE.
190           02  FILLER       PIC X(3).
200           02  FLAGS        PIC S9(9) COMP VALUE ZERO.
210      *
220       01  RPC-CL-ARG.
230           02  REQEST       PIC X(8) VALUE 'CLOSE   '.
240           02  STATUS-CODE  PIC X(5) VALUE SPACE.
250           02  FILLER       PIC X(3).
260           02  FLAGS        PIC S9(9) COMP VALUE ZERO.
270      *
280       01  RSV-ARG.
290           02  REQUEST      PIC X(8) VALUE 'MAINLOOP'.
300           02  STATUS-CODE  PIC X(5) VALUE SPACE.
310           02  FILLER       PIC X(3).
320           02  FLAGS        PIC S9(9) COMP VALUE ZERO.
330      *
340       PROCEDURE DIVISION.
350      *
360      ***********************************************
370      * RPC-OPEN (start the UAP)
380      ***********************************************
390      *
400       CALL 'CBLDCRPC' USING RPC-OP-ARG.
410           IF STATUS-CODE OF RPC-OP-ARG NOT = '00000' THEN
420             DISPLAY 'SERVER: RPC-OPEN FAILED. CODE = '
430                      STATUS-CODE OF RPC-OP-ARG
440             GO TO PROG-END
450           END-IF.
460      *
470      ***********************************************
480      * RPC-MAINLOOP (start the SPP service)
490      ***********************************************
500      *
```

```
 DISPLAY 'SERVER: ENTERING MAINLOOP...'
 CALL 'CBLDCRSV' USING RSV-ARG.
     IF STATUS-CODE OF RSV-ARG NOT = '00000' THEN
       DISPLAY 'SERVER: RPC-MAINLOOP FAILED. CODE = '
                STATUS-CODE OF RSV-ARG
     END-IF.
*
***********************************************
* End of program
***********************************************
 PROG-END.
*
***********************************************
* RPC-CLOSE (terminate the UAP)
***********************************************
*
 CALL 'CBLDCRPC' USING RPC-CL-ARG.
*
***********************************************
* Terminate the processing
***********************************************
*
 STOP RUN.
```

(f) SPP coding sample (service program)

The following shows a coding example (service program) of the SPP that was presented in the example of the request/response service.

- Coding example for the `SVHOTEL` service

```
*
***********************************************
* Example of SPP service functions (shotel.cbl file)
***********************************************
*
 IDENTIFICATION DIVISION.
*
 PROGRAM-ID.     SHOTEL.
*
***********************************************
* Set the data area
***********************************************
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
***********************************************
* Declare variables
***********************************************
```

```
*
* TPSVCDEF record
*
 01  SVCDEF-REC.
     COPY TPSVCDEF.
*
* TPTYPE record
*
 01  TYPE-REC.
     COPY TPTYPE.
*
* TPSTATUS record
*
 01  STATUS-REC.
     COPY TPSTATUS.
*
* TPSVCRET record
*
 01  SVCRET-REC.
     COPY TPSVCRET.
*
* WK-AREA is where service requests are read into
*
 01  WK-AREA PIC X(264).
*
 01  HOTEL-REC REDEFINES WK-AREA.
     COPY HOTEL.
*
***********************************************
* shotel processing
***********************************************
*
 PROCEDURE DIVISION.
*
***** set length *****
 COMPUTE LEN OF TYPE-REC = FUNCTION LENGTH(WK-AREA).
*
***********************************************
* TPSVCSTART
***********************************************
*
 CALL "TPSVCSTART" USING
      SVCDEF-REC TYPE-REC WK-AREA STATUS-REC.
*
* Shotel returns status=0 if the specified hotel
* can makea reservation. Shotel returns status=1
* if there are no rooms in the specified hotel.
* In this case, shotel return status=1 because
* there are no rooms.
*
  MOVE 1 TO RSTATUS IN HOTEL-REC.
*
***********************************************
* TPRETURN
***********************************************
*
 SET TPSUCCESS OF SVCRET-REC TO TRUE.
 MOVE 1 TO APPL-CODE OF SVCRET-REC.
*
 COPY TPRETURN
      REPLACING TPSVCRET-REC BY SVCRET-REC
                TPTYPE-REC BY TYPE-REC
                DATA-REC BY WK-AREA.
*
***********************************************
* Terminate processing
***********************************************
*
*
 END PROGRAM SHOTEL.
```

- The following shows a coding example for the SVPLANE service.

```
*
***********************************************
* Example of SPP service functions (splane.cbl file)
***********************************************
*
 IDENTIFICATION DIVISION.
*
 PROGRAM-ID.     SPLANE.
*
***********************************************
* Set the data area
***********************************************
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
***********************************************
* Declare variables
***********************************************
*
* TPSVCDEF record
*
 01  SVCDEF-REC.
```

```
           COPY TPSVCDEF.
      *
      * TPTYPE record
      *
       01  TYPE-REC.
           COPY TPTYPE.
      *
      * TPSTATUS record
      *
       01  STATUS-REC.
           COPY TPSTATUS.
      *
      * TPSVCRET record
      *
       01  SVCRET-REC.
           COPY TPSVCRET.
      *
      * WK-AREA is where service requests are read into
      *
       01  WK-AREA PIC X(264).
      *
       01  PLANE-REC REDEFINES WK-AREA.
           COPY PLANE.
      *
      ***********************************************
      * splane processing
      ***********************************************
      *
       PROCEDURE DIVISION.
      *
      ***** set length *****
       COMPUTE LEN OF TYPE-REC = FUNCTION LENGTH(WK-AREA).
      *
      ***********************************************
      * TPSVCSTART
      ***********************************************
      *
       CALL "TPSVCSTART" USING
            SVCDEF-REC TYPE-REC WK-AREA STATUS-REC.
      *
      * Splane returns status=0 if a seat on the specified flight
      * can be ticketed. Splane returns status=1 if there aren't any
      * seats on the specified flight.
     *   In this case, splane returns status=1 because there are no
      * seats.
```

```
*
  MOVE 1 TO RSTATUS IN PLANE-REC.
*
************************************************
* TPRETURN
************************************************
*
 SET TPSUCCESS OF SVCRET-REC TO TRUE.
 MOVE 0 TO APPL-CODE OF SVCRET-REC.
*
 COPY TPRETURN
      REPLACING TPSVCRET-REC BY SVCRET-REC
                TPTYPE-REC BY TYPE-REC
                DATA-REC BY WK-AREA.
*
************************************************
* Terminate processing
************************************************
*
*
 END PROGRAM SPLANE.
```

- User service definition sample

  The following shows an example of user service definition of the SPP that was presented in the example of request/response service.

```
# Example of user service definition (rrspp file)
set    service_group     = "rrspp_svg"
set    module            = "rrspp"
set    service          = "SVHOTEL=shotel","SVPLANE=splane"
set    trn_expiration_time = 180
set    trn_expiration_time_suspend = Y
set    server_type = "xatmi"
```

(2) Conversational service paradigm sample

(a) Outline of processing

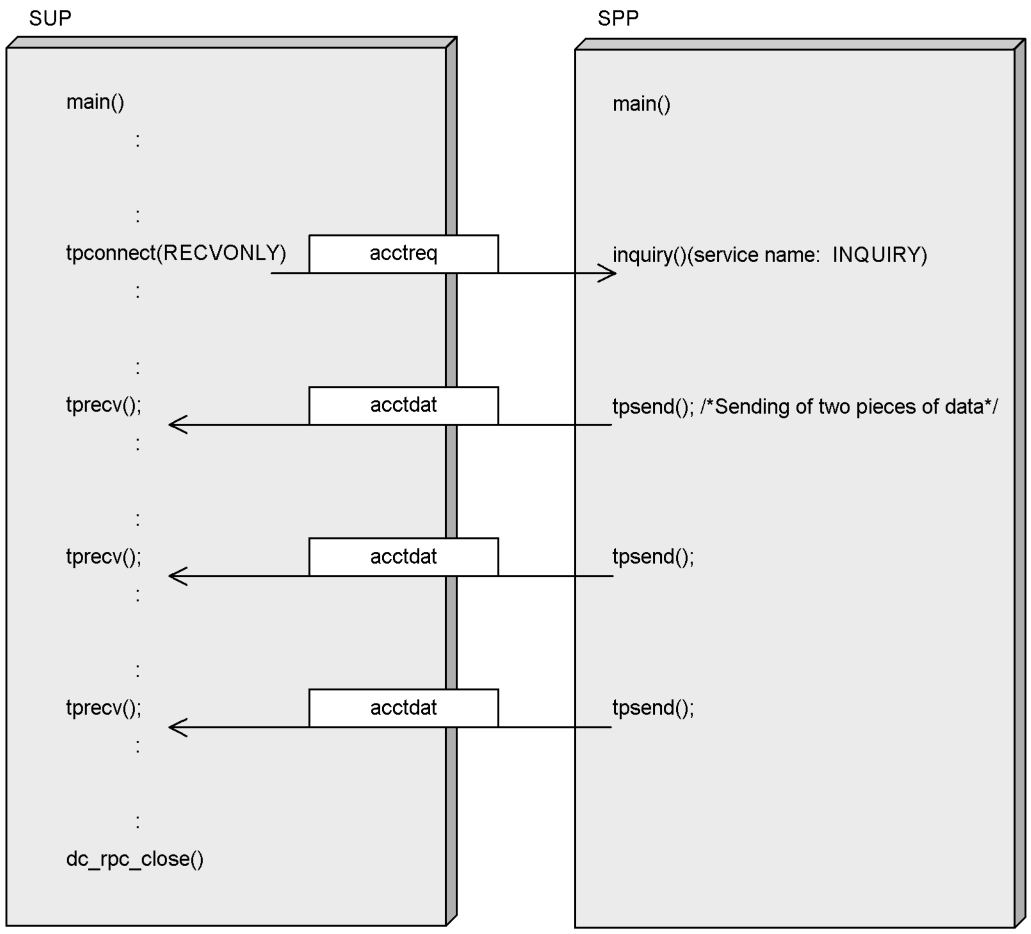
The processing of the sample here is outlined below.

The service program is activated through a typed record having data of `ACCTREQ`. The members of `ACCTREQ` indicate the upper and lower limits of the account numbers. The service program sets account data in this range in the typed record having data of `ACCTDAT` and sends the data to the originator of the conversation.

(b) UAP configuration

The following figure shows the configuration of the sample UAP.

Figure 6-5: Communication of conversational service



(c) Typed records used

The structures of typed records used are shown below.

Data for activating the service program

```
    ACCTREQ.cbl
              05 UPPERNO          PIC S9(9) COMP-5.
              05 LOWERNO          PIC S9(9) COMP-5.
```

Data for communication with the conversational service

```
             05 ACCTNO           PIC S9(9) COMP-5.
             05 NAME             PIC X(128).
             05 AMOUNT           PIC S9(4) COMP-5.
             05 FILLER           PIC X(2).
```

(d) SUP sample

- XATMI interface definition sample

  The following shows the XATMI interface definition of the SUP for the sample conversational service.

```
/*  Example of XATMI interface definition (rrsupcb.def file) */
called_servers = { "rrsppcb.def" } ;
```

- SUP coding sample

  The following shows a coding example for the SUP used in the example of the conversational service.

```
      *
      ***********************************************
      * Example of SUP (convsup.cbl file)
      ***********************************************
      *
       IDENTIFICATION DIVISION.
      *
       PROGRAM-ID. MAIN.
      *
      ***********************************************
      * Set the data area
      ***********************************************
      *
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      ***********************************************
      * Declare constants
      ***********************************************
      ***********************************************
      * Declare variables
      ***********************************************
      ***** typed record for INQUIRY when inquiry service *****
       01  ACCTREQ-REC.
           COPY ACCTREQ.
      ***** type information for INQUIRY *****
       01  ACCTREQTYPE-REC.
           COPY TPTYPE.
      ***** service definition for INQUIRY *****
       01  ACCTREQDEF-REC.
```

```
     COPY TPSVCDEF.
***** return record *****
 01  STATUS-REC.
     COPY TPSTATUS.
***** received record between INQUIRY and CONVSUP
 01  ACCTDATA-REQ.
     COPY ACCTDATA.
**** type information received record between INQUIRY and CONVSUP
 01  ACCTDATATYPE-REC.
     COPY TPTYPE.
***** service definition for INQUIRY *****
 01  ACCTDATADEF-REC.
     COPY TPSVCDEF.
***** working area is used for replies *****
 01  WK-AREA    PIC X(136).
***** redefine working area 1 *****
 01  ACCTREQ-REP REDEFINES WK-AREA.
     COPY ACCTREQ.
***** redefine working area 2 *****
 01  ACCTDATA-REP REDEFINES WK-AREA.
     COPY ACCTDATA.
***** typed information  *****
 01  TYPE-REC.
     COPY TPTYPE.
***** others *****
 01  WSTATUS PIC S9(9) COMP-5.
***** dc_rpc_open *****
 01  RPC-OP-ARG.
     02  REQEST        PIC X(8) VALUE 'OPEN    '.
     02  STATUS-CODE   PIC X(5) VALUE SPACE.
     02  FILLER        PIC X(3).
     02  FLAGS         PIC S9(9) COMP VALUE ZERO.
***** dc_rpc_close *****
 01  RPC-CL-ARG.
     02  REQEST        PIC X(8) VALUE 'CLOSE   '.
     02  STATUS-CODE   PIC X(5) VALUE SPACE.
     02  FILLER        PIC X(3).
     02  FLAGS         PIC S9(9) COMP VALUE ZERO.
***** dc_adm_complete ****
 01  ADM-ARG.
     02    REQUEST     PIC X(8) VALUE 'COMPLETE'.
     02    STATUS-CODE PIC X(5) VALUE SPACE.
     02    FILLER      PIC X(3).
     02    FLAGS       PIC S9(9) COMP VALUE ZERO.
     02    FILLER      PIC X(3).
*
 01  FLAG              PIC S9(9) COMP VALUE ZERO.
```

```
*
****** for TX interface *****
*
 01  TX-RETURN-STATUS.
     COPY TXSTATUS.
*
 01  RS REDEFINES TX-RETURN-STATUS.
     05 RSVAL         PIC S9(9) COMP-5.
*
 01 TX-INFO-AREA.
     COPY TXINFDEF.
*
 PROCEDURE DIVISION.
*
***********************************************
* RPC-OPEN (start the UAP)
***********************************************
*
 CALL 'CBLDCRPC' USING RPC-OP-ARG.
     IF STATUS-CODE OF RPC-OP-ARG NOT = '00000' THEN
       DISPLAY 'CLIENT: RPC-OPEN FAILED. CODE = '
                STATUS-CODE OF RPC-OP-ARG
       GO TO PROG-END
     END-IF.
*
***********************************************
* TXOPEN (open the resource manager)
***********************************************
*
 CALL "TXOPEN" USING TX-RETURN-STATUS.
     IF RSVAL OF RS NOT = 0 THEN
       DISPLAY 'CLIENT:TX-OPEN FAILED. CODE = '
       RSVAL OF RS
       GO TO PROG-END
     END-IF.
*
***********************************************
* TX-SET-TRANSACTION-TIMEOUT                  *
* (set the transaction monitoring interval)   *
***********************************************
*
 MOVE 180 TO TRANSACTION-TIMEOUT OF TX-INFO-AREA.
 CALL "TXSETTIMEOUT" USING TX-INFO-AREA TX-RETURN-STATUS.
     IF RSVAL OF RS NOT = 0 THEN
       DISPLAY 'CLIENT:TX-SET-TRANSACTION-TIMEOUT FAILED. CODE = '
       RSVAL OF RS
```

```
              GO TO PROG-END
            END-IF.
      *
      ***********************************************
      * ADM-COMPLETE (report completion of          *
      * user server start processing)               *
      ***********************************************
      *
       CALL 'CBLDCADM' USING ADM-ARG.
            IF STATUS-CODE OF ADM-ARG NOT = '00000' THEN
              DISPLAY 'CLIENT: ADM-COMPLETE FAILED. CODE = '
                       STATUS-CODE OF ADM-ARG
              GO TO PROG-END
            END-IF.
      *
      ***********************************************
      * TX-SET-TRANSACTION-CONTROL
      * (unchained mode settings)
      ***********************************************
      *
       MOVE 0 TO TRANSACTION-CONTROL OF TX-INFO-AREA.
       CALL "TXSETTRANCTL" USING TX-INFO-AREA
TX-RETURN-STATUS.
            IF RSVAL OF RS NOT = 0 THEN
              DISPLAY 'CLIENT:TX-SET-TRANSACTION-CONTROL
FAILED. CODE ='
              RSVAL OF RS
            END-IF.
      *
      ***********************************************
      * TPXBEGIN (start transaction)
      ***********************************************
      *
       CALL "TXBEGIN" USING TX-RETURN-STATUS.
            IF RSVAL OF RS NOT = 0 THEN
              DISPLAY 'CLIENT:TX-BEGIN FAILED. CODE ='
              RSVAL OF RS
              GO TO PROG-END
            END-IF.
      *
      ***********************************************
      * TPCONNECT (request service (INQUIRY))
      ***********************************************
      *
      ***** set parameters *****
      *
      ***** set up ACCTREQDEF-REC *****
      *
```

```
           MOVE LOW-VALUES            TO ACCTREQDEF-REC.
           MOVE 1                     TO TPSENDRECV-FLAG OF ACCTREQDEF-REC.
           MOVE "INQUIRY"             TO SERVICE-NAME    OF ACCTREQDEF-REC.
          *
          ***** set up ACCTREQTYPE-REC *****
          *
           MOVE "X_COMMON"            TO REC-TYPE    OF ACCTREQTYPE-REC.
           MOVE "acctreq"             TO SUB-TYPE    OF ACCTREQTYPE-REC.
           COMPUTE LEN OF ACCTREQTYPE-REC = FUNCTION LENGTH(ACCTREQ-REC).
          *
          ***** set up ACCTREQ-REC *****
          *
           MOVE "100000000" TO LOWERNO OF ACCTREQ-REC.
           MOVE "200000000" TO UPPERNO OF ACCTREQ-REC.
          *
          ***** CALL TPCONNECT *****
           CALL "TPCONNECT" USING
               ACCTREQDEF-REC ACCTREQTYPE-REC ACCTREQ-REC STATUS-REC.
               IF NOT TPOK OF STATUS-REC THEN
                DISPLAY 'CLIENT: INQUARY SERVICE REQ WAS FAIL. CODE = '
                          TP-STATUS OF STATUS-REC
                 DISPLAY  'CLIENT:TX-ROLLBACK STARTED'
                 CALL "TXROLLBACK" USING TX-RETURN-STATUS
                 DISPLAY  'CLIENT:TX-ROLLBACK ENDED'
                 IF RSVAL OF RS NOT = 0 THEN
                   DISPLAY 'CLIENT:TX-ROLLBACK FAILED. CODE ='
                   RSVAL OF RS
                 END-IF
                 GO TO PROG-END
               END-IF.
          *
           DISPLAY 'CLIENT: INQUIRY SERVICE REQ WAS SUCCESS '.
          *
          ***** set up ACCTDATA-REC *****
          *
           MOVE 0 TO TP-STATUS OF STATUS-REC.
           MOVE LOW-VALUES  TO ACCTDATADEF-REC.
           MOVE COMM-HANDLE OF ACCTREQDEF-REC TO
                COMM-HANDLE OF ACCTDATADEF-REC.
           MOVE "X_COMMON"            TO REC-TYPE    OF ACCTDATATYPE-REC.
```

```
 MOVE "acctdata"            TO SUB-TYPE    OF ACCTDATATYPE-REC.
 COMPUTE LEN OF ACCTDATATYPE-REC = FUNCTION LENGTH(ACCTDATA-REQ).
*
 PERFORM WITH TEST AFTER UNTIL NOT TPOK OF STATUS-REC
*
***********************************************
* TPRECV (receive data)
***********************************************
   MOVE 0 TO TP-STATUS OF STATUS-REC
   CALL "TPRECV" USING
       ACCTDATADEF-REC ACCTDATATYPE-REC WK-AREA STATUS-REC
       IF TPOK OF STATUS-REC THEN
       DISPLAY 'CLIENT: RECEIVED ACOUNT INFORMATION '
         DISPLAY 'CLIENT: ACCOUNT NUMBER ='
                 ACCTNO IN ACCTDATA-REP
       DISPLAY 'CLIENT: NAME =' ANAME IN ACCTDATA-REP
         DISPLAY 'CLIENT: AMOUNT =' AMOUNT IN ACCTDATA-REP
       END-IF
 END-PERFORM.
*
     IF TPEEVENT OF STATUS-REC THEN
       IF TPEV-SVCSUCC OF STATUS-REC THEN
         DISPLAY 'CLIENT:INQUARY SERVICE SUCCESS'
*
***********************************************
*   TX-COMMIT (commit transaction)
***********************************************
*
         DISPLAY  'CLIENT:TX-COMMIT STARTED'
         CALL "TXCOMMIT" USING TX-RETURN-STATUS
         DISPLAY  'CLIENT:TX-COMMIT ENDED'
           IF RSVAL OF RS NOT = 0 THEN
            DISPLAY 'CLIENT:TX-COMMIT FAILED. CODE ='
             RSVAL OF RS
           END-IF
       ELSE
         DISPLAY 'CLIENT:EVENT OCCURRED IN INQUIRY SERVICE'
         DISPLAY 'CODE =' TPEVENT OF STATUS-REC
*
***********************************************
*   TX-ROLLBAK (roll back transaction)
***********************************************
*
```

```
         DISPLAY  'CLIENT:TX-ROLLBACK STARTED'
         CALL "TXROLLBACK" USING TX-RETURN-STATUS
         DISPLAY  'CLIENT:TX-ROLLBACK ENDED'
         IF RSVAL OF RS NOT = 0 THEN
          DISPLAY 'CLIENT:TX-ROLLBACK FAILED. CODE ='
           RSVAL OF RS
         END-IF
       END-IF
     ELSE
    DISPLAY 'CLIENT:EVENT OCCURED IN INQUARY SERVICE'
       DISPLAY 'CODE =' TPEVENT OF STATUS-REC
*
***********************************************
*   TX-ROLLBAK (roll back transaction)
***********************************************
*
       DISPLAY  'CLIENT:TX-ROLLBACK STARTED'
       CALL "TXROLLBACK" USING TX-RETURN-STATUS
       DISPLAY  'CLIENT:TX-ROLLBACK ENDED'
       IF RSVAL OF RS NOT = 0 THEN
         DISPLAY 'CLIENT:TX-ROLLBACK FAILED. CODE ='
         RSVAL OF RS
       END-IF
     END-IF.
*
***********************************************
* Terminate processing
***********************************************
*
 PROG-END.
*
 DISPLAY 'CLIENT: SEE YOU LATER'
*
***********************************************
* RPC-CLOSE (terminate the UAP)
***********************************************
*
 CALL 'CBLDCRPC' USING RPC-CL-ARG.
*
 STOP RUN.
```

- User service definition sample

  The following shows an example of a user service definition of the SPP that was presented in the example of the conversational service.

```
# Example of user service definition (convsup file)
set module     "convsup" # Name of executable file
set watch_time = 180     # Maximum time to wait
```

```
                         # for a response
set receive_from = none   # Receiving method
set trn_expiration_time = 180
          # Expiry time in transaction branch
set trn_expiration_time_suspend = Y # Always specify Y
```

(e) SPP sample

- XATMI interface definition sample

  The following shows an example of XATMI interface definition of the SPP that was presented in the example of the conversational service.

```
/* Example of XATMI interface definition of      *
 * SPP (cvsppcb.def file)                        */
X_COMMON acctreq {
    long   upperno;
    long   lowerno;
};
X_COMMON acctdata {
    long   acctno;
    char   name[128];
    short  amount;
};
service INQUIRY(X_COMMON acctreq) ;
```

- SPP coding sample (main program)

  The following shows a coding example (main program) of the SPP that was presented in the example of the conversational service.

```
    *
    ***********************************************
    * Example of SPP (convspp.cbl file)
    ***********************************************
    *
     IDENTIFICATION DIVISION.
    *
     PROGRAM-ID.     MAIN.
    *
    ***********************************************
    * Set the data area
    ***********************************************
    *
     DATA DIVISION.
     WORKING-STORAGE SECTION.
     01  RPC-OP-ARG.
         02  REQEST       PIC X(8) VALUE 'OPEN    '.
         02  STATUS-CODE  PIC X(5) VALUE SPACE.
         02  FILLER       PIC X(3).
         02  FLAGS        PIC S9(9) COMP VALUE ZERO.
```

```
*
 01  RPC-CL-ARG.
     02  REQEST       PIC X(8) VALUE 'CLOSE   '.
     02  STATUS-CODE  PIC X(5) VALUE SPACE.
     02  FILLER       PIC X(3).
     02  FLAGS        PIC S9(9) COMP VALUE ZERO.
*
 01  RSV-ARG.
     02  REQUEST      PIC X(8) VALUE 'MAINLOOP'.
     02  STATUS-CODE  PIC X(5) VALUE SPACE.
     02  FILLER       PIC X(3).
     02  FLAGS        PIC S9(9) COMP VALUE ZERO.
*
 PROCEDURE DIVISION.
*
***********************************************
* RPC-OPEN (start the UAP)
***********************************************
*
 CALL 'CBLDCRPC' USING RPC-OP-ARG.
     IF STATUS-CODE OF RPC-OP-ARG NOT = '00000' THEN
       DISPLAY 'SERVER: RPC-OPEN FAILED. CODE = '
                STATUS-CODE OF RPC-OP-ARG
       GO TO PROG-END
     END-IF.
*
***********************************************
* RPC-MAINLOOP (start the SPP service)
***********************************************
*
 DISPLAY 'SERVER: ENTERING MAINLOOP...'
 CALL 'CBLDCRSV' USING RSV-ARG.
     IF STATUS-CODE OF RSV-ARG NOT = '00000' THEN
       DISPLAY 'SERVER: RPC-MAINLOOP FAILED. CODE = '
                STATUS-CODE OF RSV-ARG
     END-IF.
*
***********************************************
* Terminate the program
***********************************************
 PROG-END.
*
***********************************************
* RPC-CLOSE (terminate the UAP)
***********************************************
*
 CALL 'CBLDCRPC' USING RPC-CL-ARG.
*
```

```
**********************************************
* Terminate processing
**********************************************
*
 STOP RUN.
```

- SPP coding sample (service program)

  The following shows a coding example (service program) of the SPP that was presented in the example of the conversational service.

```
*
**********************************************
* INQUIRY    service program (convsvc.cbl file)
**********************************************
*
 IDENTIFICATION DIVISION.
*
 PROGRAM-ID.     INQUIRY.
*
**********************************************
* Set the data area
**********************************************
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
**********************************************
* Declare variables
**********************************************
*
* TPSVCDEF record
*
 01  SVCDEF-REC.
     COPY TPSVCDEF.
*
* TPTYPE record
*
 01  TYPE-REC.
     COPY TPTYPE.
*
* TPSTATUS record
*
 01  STATUS-REC.
     COPY TPSTATUS.
*
* TPSVCRET record
*
 01  SVCRET-REC.
```

```
     COPY TPSVCRET.
*
* WK-AREA is where service requests are read into
*
 01  WK-AREA PIC X(136).
*
 01  ACCTREQ-REC REDEFINES WK-AREA.
     COPY ACCTREQ.
*
 01  ACCTDATA-REC.
     COPY ACCTDATA.
*
* TPSVCDEF record for TPSEND
*
 01  ACCTDATADEF-REC.
     COPY TPSVCDEF.
*
* TPTYPE record for TPSEND
*
 01  ACCTDATATYPE-REC.
     COPY TPTYPE.
*
*
***********************************************
* Inquiry processing
***********************************************
*
 PROCEDURE DIVISION.
*
***** set length *****
 COMPUTE LEN OF TYPE-REC = FUNCTION LENGTH(WK-AREA).
*
***********************************************
* TPSVCSTART
***********************************************
*
 CALL "TPSVCSTART" USING
      SVCDEF-REC TYPE-REC WK-AREA STATUS-REC.
*
* find user data files between lower and upper account number.
* In this case 2 data was found, and was replied.
*
***********************************************
* Set the send data
***********************************************
 MOVE LOW-VALUES        TO ACCTDATADEF-REC.
 MOVE COMM-HANDLE OF SVCDEF-REC TO COMM-HANDLE OF
```

```
ACCTDATADEF-REC.
MOVE "X_COMMON"      TO REC-TYPE OF ACCTDATATYPE-REC.
MOVE "acctdata"      TO SUB-TYPE OF ACCTDATATYPE-REC.
 COMPUTE LEN OF ACCTDATATYPE-REC =
         FUNCTION LENGTH(ACCTDATA-REC).
*
************************************************
* TPSEND (send the first data)
************************************************
*
 MOVE "10000001"       TO ACCTNO  OF ACCTDATA-REC.
 MOVE "HITACHI HANAKO" TO ANAME   OF ACCTDATA-REC.
 MOVE "2000"           TO AMOUNT  OF ACCTDATA-REC.
 CALL "TPSEND" USING ACCTDATADEF-REC ACCTDATATYPE-REC
                     ACCTDATA-REC STATUS-REC.
     IF TPOK OF STATUS-REC THEN
       MOVE 0 TO TP-RETURN-VAL OF SVCRET-REC
     ELSE
       MOVE 1 TO TP-RETURN-VAL OF SVCRET-REC
       GO TO PROG-END
     END-IF
*
************************************************
* TPSEND (send the second data)
************************************************
*
 MOVE "10000002"       TO ACCTNO  OF ACCTDATA-REC.
 MOVE "HITACHI TAROU"  TO ANAME   OF ACCTDATA-REC.
 MOVE "1000"           TO AMOUNT  OF ACCTDATA-REC.
 CALL "TPSEND" USING ACCTDATADEF-REC ACCTDATATYPE-REC
                     ACCTDATA-REC STATUS-REC.
     IF TPOK OF STATUS-REC THEN
       MOVE 0 TO TP-RETURN-VAL OF SVCRET-REC
     ELSE
       MOVE 1 TO TP-RETURN-VAL OF SVCRET-REC
       GO TO PROG-END
     END-IF
*
************************************************
* TPRETURN (terminate the receive program)
************************************************
*
 SET TPSUCCESS OF SVCRET-REC TO TRUE.
 MOVE 1 TO APPL-CODE OF SVCRET-REC.
*
 PROG-END.
*
 MOVE "        "         TO REC-TYPE OF TYPE-REC.
```

```
1330       MOVE "                " TO SUB-TYPE OF TYPE-REC.
1340       MOVE 0 TO  LEN OF TYPE-REC.
1350       COPY TPRETURN
1360            REPLACING TPSVCRET-REC BY SVCRET-REC
1370                      TPTYPE-REC BY TYPE-REC
1380                      DATA-REC BY WK-AREA.
1390      *
1400      ***********************************************
1410      * Terminate processing
1420      ***********************************************
1430      *
1440      *
1450       END PROGRAM INQUIRY.
```

- User service definition sample

  The following shows an example of a user service definition of the SPP that was presented in the example of the conversational service.

```
 10  # Example of user service definition (convspp file)
 20    set  service_group = "convspp_svg"
 25                          # Service group name
 30    set  module = "convspp" # Name of executable file
 40    set  service = "INQUIRY=inquiry"
 50               # Service name = entry point name
 60    set  watch_time = 180  # Maximum time to wait
 65                           # for a response
 70    set  trn_expiration_time =  240
 80             # Expiry time in transaction branch
 90    set  trn_expiration_time_suspend = Y # Always
 95                                         # specify Y
100    set  server_type = "xatmi" # Server type
110    set  receive_from = "socket"  # Receiving method
```

6.4.2 TX interface sample

This subsection shows a coding example for an SUP that uses the X/Open TX interface. This SUP uses TX-interfaced transaction control for processing that was described in *6.1 Coding samples for client/server UAPs (SUP, SPP DAM access)*. See *6.1 Coding samples for client/server UAPs (SUP, SPP DAM access)* for the process configuration and details of the SPP to which the service request is addressed. However, `TX-RETURN-STATUS` at line numbers 460 and 470 are redefined as `RS REDEFINES TX-RETURN-STATUS` because it cannot be correctly referenced if it is directly invoked from the process.

```
 10  *
 20  ***************************************************
 30  * SUP01                                           *
 40  ***************************************************
```

```
*
 IDENTIFICATION DIVISION.
*
 PROGRAM-ID. MAIN.
*
***************************************************
* Set the data area                           *
***************************************************
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  RPC-ARG1.
     02 REQUEST      PIC X(8) VALUE SPACE.
     02 STATUS-CODE  PIC X(5) VALUE SPACE.
     02 FILLER       PIC X(3).
     02 FLAGS        PIC S9(9) COMP VALUE ZERO.
*
 01  RPC-ARG2.
     02 REQUEST      PIC X(8) VALUE SPACE.
     02 STATUS-CODE  PIC X(5) VALUE SPACE.
     02 FILLER       PIC X(3).
     02 FLAGS        PIC S9(9) COMP VALUE ZERO.
     02 DESCRIPTOR   PIC S9(9) COMP VALUE ZERO.
     02 S-NAME       PIC X(32) VALUE SPACE.
     02 G-NAME       PIC X(32) VALUE SPACE.
*
 01  RPC-ARG3.
     02  SEND-DATA-LENG  PIC S9(9) COMP VALUE ZERO.
     02  SEND-DATA       PIC X(32) VALUE SPACE.
*
 01  RPC-ARG4.
     02  RECEIVE-DATA-LENG PIC S9(9) COMP VALUE ZERO.
     02  RECEIVE-DATA      PIC X(32) VALUE SPACE.
*
 01  ADM-ARG1.
     02  REQUEST      PIC X(8) VALUE SPACE.
     02  STATUS-CODE  PIC X(5) VALUE SPACE.
     02  FILLER       PIC X(3).
     02  FLAGS        PIC S9(9) COMP VALUE ZERO.
     02  FILLER       PIC X(3).
*
 01  TX-RETURN-STATUS.
     COPY TXSTATUS.
*
 01  RS REDEFINES TX-RETURN-STATUS.
     05 RSVAL         PIC S9(9) COMP-5.
*
 01  TX-INFO-AREA.
 530       COPY TXINFDEF.
 540  *
 550   PROCEDURE DIVISION.
 560  *
 570  ***************************************************
 580  * RPC-OPEN (start the UAP)                        *
 590  ***************************************************
 600  *
 610   MOVE 'OPEN' TO REQUEST OF RPC-ARG1.
 620   MOVE ZERO   TO FLAGS  OF RPC-ARG1.
 630   CALL 'CBLDCRPC' USING RPC-ARG1.
 640       IF STATUS-CODE OF RPC-ARG1 NOT = '00000' THEN
 650         DISPLAY 'SUP01:RPC-OPEN FAILED. CODE = '
 660         STATUS-CODE OF RPC-ARG1
 670         GO TO PROG-END
 680       END-IF.
 690  *
 700  ***************************************************
 710  * TX-OPEN (open the resource manager)             *
 720  ***************************************************
 730  *
 740   CALL 'TXOPEN' USING TX-RETURN-STATUS.
 750       IF RSVAL OF RS NOT = 0 THEN
 760         DISPLAY 'SUP01:TX-OPEN FAILED. CODE = '
 770         RSVAL OF RS
 780         GO TO PROG-END
 790       END-IF.
 800  *
 810  ***************************************************
 820  * TX-SET-TRANSACTION-TIMEOUT                      *
      * (set the transaction monitoring interval)       *
 830  ***************************************************
 840  *
 850   MOVE 180 TO TRANSACTION-TIMEOUT OF TX-INFO-AREA.
 860   CALL 'TXSETTIMEOUT' USING TX-INFO-AREA TX-RETURN-STATUS.
 870       IF RSVAL OF RS NOT = 0 THEN
 880         DISPLAY 'SUP01:TX-SET-TRANSACTION-TIMEOUT FAILED.
CODE = '
 890         RSVAL OF RS
 900         GO TO PROG-END
 910       END-IF.
 920  *
 930  ***************************************************
 940  * ADM-COMPLETE (report completion of user         *
      * server start processing)                        *
 950  ***************************************************
 960  *
 970   MOVE 'COMPLETE' TO REQUEST OF ADM-ARG1.
 CALL 'CBLDCADM' USING ADM-ARG1.
     IF STATUS-CODE OF ADM-ARG1 NOT = '00000' THEN
       DISPLAY 'SUP01:ADM-COMPLETE FAILED. CODE = '
       STATUS-CODE OF ADM-ARG1
       GO TO PROG-END
     END-IF.
*
***************************************************
* TX-BEGIN (start the transaction)                *
***************************************************
*
 CALL 'TXBEGIN' USING TX-RETURN-STATUS.
     IF RSVAL OF RS NOT = 0 THEN
       DISPLAY 'SUP01:TX-BEGIN FAILED. CODE = '
       RSVAL OF RS
       GO TO TRAN-END
     END-IF.
*
***************************************************
* TX-INFO (acquire transaction information)       *
***************************************************
*
 CALL 'TXINFORM' USING TX-INFO-AREA TX-RETURN-STATUS.
     IF RSVAL OF RS <= 0 THEN
       DISPLAY 'SUP01:NOT IN TRANSACTION. CODE = '
       RSVAL OF RS
       GO TO PROG-END
     ELSE
       IF RSVAL OF RS = 1 THEN
         DISPLAY 'SUP01:RETURN = ' COMMIT-RETURN
         DISPLAY 'SUP01:CONTROL = ' TRANSACTION-CONTROL
         DISPLAY 'SUP01:TIMEOUT = ' TRANSACTION-TIMEOUT
         DISPLAY 'SUP01:STATE = ' TRANSACTION-STATE
       END-IF
     END-IF.
***************************************************
* RPC-CALL (request a remote service)             *
***************************************************
*
 MOVE 'CALL' TO REQUEST OF RPC-ARG2.
 MOVE 'SVR01' TO G-NAME OF RPC-ARG2.
 MOVE 'SVR01' TO S-NAME OF RPC-ARG2.
 MOVE 'SUP01:DATA OpenTP1' TO SEND-DATA OF RPC-ARG3.
 MOVE 32 TO SEND-DATA-LENG OF RPC-ARG3.
 MOVE 32 TO RECEIVE-DATA-LENG OF RPC-ARG4.
 CALL 'CBLDCRPC' USING RPC-ARG2 RPC-ARG3 RPC-ARG4.
     IF STATUS-CODE OF RPC-ARG2 NOT = '00000' THEN
       DISPLAY 'SUP01:RPC-CALL RETURN CODE = '
       STATUS-CODE OF RPC-ARG2
*       GO TO TRAN-END
     END-IF.
 DISPLAY 'SERVICE FUNCTION RETURN = ' RECEIVE-DATA.
 TRAN-END.
*
***************************************************
* TX-SET-TRANSACTION-CONTROL (set the unchained    *
* mode)                                           *
***************************************************
*
 MOVE 0 TO TRANSACTION-CONTROL OF TX-INFO-AREA.
 CALL 'TXSETTRANCTL' USING TX-INFO-AREA TX-RETURN-STATUS.
     IF RSVAL OF RS NOT = 0 THEN
       DISPLAY 'SUP01:TX-SET-TRANSACTION-CONTROL FAILED. CODE = '
       RSVAL OF RS
     END-IF.
*
***************************************************
* TX-COMMIT (commit in unchained mode)            *
***************************************************
*
 CALL 'TXCOMMIT' USING TX-RETURN-STATUS.
     IF RSVAL OF RS NOT = 0 THEN
       DISPLAY 'SUP01:TX-COMMIT FAILED. CODE = '
       RSVAL OF RS
     END-IF.
 PROG-END.
*
***************************************************
* RPC-CLOSE (terminate the UAP)                   *
***************************************************
*
 MOVE 'CLOSE' TO REQUEST OF RPC-ARG1.
 MOVE ZERO    TO FLAGS  OF RPC-ARG1.
 CALL 'CBLDCRPC' USING RPC-ARG1.
 DISPLAY 'SUP01:SUP PROCESS ENDED'.
 STOP RUN.
```

Chapter

7. Reference for Application Activation

This chapter explains user exit routines (written in C language) and MCF event reference information which are related to the facility for activating application programs in an environment where TP1/Message Control is used.

This chapter contains the following sections:

Function format of user exit routine that determines the inheriting timer-start message

The user exit routine that determines the inheriting timer-start message is called in the following format:

Format

■ ANSI C , C++

```
#include <dcmpsv.h>
long uoc_func (dcmpsv_uoc_rtime *parm)
```

■ K&R C

```
#include <dcmpsv.h>
long uoc_func(parm)

dcmpsv_uoc_rtime *parm ;
```

Description

If the use of the timer-started application program activate (`CBLDCMCF('EXECAP '))` is followed by an error which raises the need for rerunning OpenTP1, this user exit routine can change the timer-start environment. It can perform the following:

- Inherit or cancel the current timer-start
- Make inherited timer-start immediate start
- Change the name of the application to be timer-started

When installing in the MCF the user exit routine that determines the inheriting timer-start message, specify the address of the user exit routine function in the MCF main function for the application startup service. The MCF main function for the application startup service does not depend on the communication protocol.

For details on how to create the MCF main function for the application startup service, see the *OpenTP1 Operation* manual.

When `uoc_func` (user exit routine that determines the inheriting time-start message) is called, the following parameters are passed from the MCF to parm.

Parameters

- dcmpsv_uoc_rtime

```
typedef struct {
     char le_name[9] ;      ... Input source logical terminal name
     char reserve1[7] ;     ... Reserved
     char ap_name[9] ;      ... Application name
     char reserve2[7] ;     ... Reserved
     long exec_time ;       ... Timer start time
     char ap_type ;         ... Application type
                                'a': ans type; 'n': noans type
     char time_type;        ... Timer-start type
                                'i': Interval specification
                                     for timer start
                                't': Time point specification
                                       for timer start
     char reserve3[26] ;    ... Reserved
} dcmpsv_uoc_rtime;
```

Arguments whose value is passed from MCF to user exit routine

- `le_name`

  The input source logical terminal name is set here. If the application program activate (`CBLDCMCF('EXECAP ')`) is called from the SPP, '*' is set here.

- `ap_name`

  The application name specified by the UAP in the timer-started application program activate (`CBLDCMCF('EXECAP ')`) is set here.

- `exec_time`

  The MHP start time specified by the UAP in the timer-started application program activate (`CBLDCMCF('EXECAP ')`) is set here, as the number of seconds counted from 00:00:00 on January 1, 1970.

- `ap_type`

  The application type of the UAP which issued the timer-started application program activate (`CBLDCMCF('EXECAP ')`) is set here:

  `'a'`: `ans` type

  `'n'`: `noans` type

- `time_type`

  The timer-start type specified by the UAP in the timer-started application program activate (`CBLDCMCF('EXECAP ')`) is set here:

  `'i'`: Interval specification for timer start

  `'t'`: Time point specification for timer start

Arguments whose value is set in the user exit routine

- `ap_name`

  To change the application to be timer-started, specify the new application name here. The name specified here has effect when `DCMPSV_UOC_TIME_JUST` is specified for the return value.

Return values

`uoc_func()` must return the following values:

Return value	Explanation
`DCMPSV_UOC_TIME_CONTINUE`	Timer-start is inherited
`DCMPSV_UOC_TIME_JUST`	Immediate start will be in effect
`DCMPSV_UOC_TIME_DEQ`	Timer-start is canceled

The subsequent MCF processing varies depending on the return value from `uoc_func()` as follows:

- `DCMPSV_UOC_TIME_CONTINUE`

  If this value is returned from the user exit routine, the MCF counts the seconds from 00:00:00 on January 1, 1970 to the present time and compares it with the time specified in the application program activate (`CBLDCMCF('EXECAP ')`). If the present time is later than the time specified in the function, the MCF immediately starts the pertinent MHP. Otherwise, the application will be timer-started.

- `DCMPSV_UOC_TIME_JUST`

  If this value is returned from the user exit routine, the MCF immediately starts the pertinent MHP. If this value is to be returned, the application to be immediately started can be changed in the user exit routine. However, change to an MHP for MCF event processing is not allowed. If the specified new application name is not defined, ERREVT4 is reported.

  If the application name of the UAP to be immediately started by the user exit routine is changed and the application types of the old and new MHPs to be started are different, the segments to be timer-started are deleted from the output queue, with the output of a warning message (`KFCA10711-W`).

- `DCMPSV_UOC_TIME_DEQ`

  If this value is returned from the user exit routine, the MCF cancels timer-start. The segments to be timer-started are deleted from the output queue, with the output of an information message (`KFCA10700-I`).

If another value is returned from the user exit routine, the segments to be timer-started are deleted from the output queue, with the output of a warning message (`KFCA10710-W`).

Notes on creating user exit routines

- Functions available to user exit routines

  When creating a user exit routine, keep in mind that only the functions listed below can be used. If a function that is not listed below is used for a user exit routine, operation is unpredictable.

  - Memory manipulation functions

    Data area management (example: `malloc`, `free`)

    Shared memory management (example: `shmctl`, `shmget`, `shmop`)

    Memory manipulation (example: `memcpy`)

    Character string manipulation (example: `strcpy`)

  - Time acquisition functions

- User exit routine errors

  If an error is detected by a user exit routine, it must be reported to the MCF using the MCF-specified return code. If a process terminating signal or `abort()` is issued in a user exit routine, the MCF will terminate abnormally.

- User exit routine execution timing

  Execution timing of a user exit routine started by the MCF may not be executed in synchronization with the OpenTP1 system and UAP startup and termination sequences. Design the user exit routine to ensure that no problems will occur even if the routine is executed before the UAP, or if it is called after all UAPs have terminated.

- Local variable size of user exit routines

  The sum of the sizes of the local variables used in a single user exit routine must be within 1 kilobyte. Do not call a function recursively within a user exit routine.

Data format of MCF event that reports discarding of a timer-start message (ERREVT4)

The format of the data passed as the first segment of the event that reports discarding of a timer-start message (ERREVT4) is shown blow. For the format of MCF event information other than ERREVT4, see the applicable *OpenTP1 Protocol* manual.

MCF event ERREVT4 information

Table 7-1 gives MCF event ERREVT4 information. Table 7-2 gives the reason codes for ERREVT4. Formats 1 and 2 indicate buffer types 1 and 2, respectively.

Table 7-1: MCF event ERREVT4 information

Item	**Position (byte)**		**Length (bytes)**	**Attribute**	**Explanation**
	Format 1	**Format 2**			
Reserved (only for format 1)	0	--	2	--	--
Reserved (only for format 1)	2	--	2	--	--
Error event code	4	0	3	Alphanumeric	ERR is set here.
	7	3	3	--	--
	10	6	2	Alphanumeric	A 4 Δ ' indicating ERREVT4 is set here.
Input source logical terminal name	12	8	8	Alphanumeric	Name of the logical terminal from which the message was input. A * is set here in the following cases: 1. An error occurred in the MHP which was started as an application by the SPP. 2. In addition to the above error, another error occurred in an MHP which was started as an application by the MHP that was started as an MCF event processing MHP.
Reserved	20	16	20	--	--

Item	Position (byte)		Length (bytes)	Attribute	Explanation
	Format 1	Format 2			
Application name	40	36	8	Alphanumeric	Name of the timer-started application which encountered the error.
Reserved	48	44	8	--	--
Reserved	56	52	8	--	--
Reserved	64	60	8	--	--
Connection name	72	68	8	Alphanumeric	Name of the connection. A * is set here in the following cases: 1. An error occurred in the MHP which was started as an application by the SPP. 2. In addition to the above error, another error occurred in an MHP which was started as an application by the MHP that was started as an MCF event processing MHP.
Reserved	80	76	16	--	--
Message input date	96	92	8	External decimal	Date the message was input on the terminal, in the format of *yyyymmdd*: *yyyy*: Year *mm*: Month *dd*: Day
Message input time	104	100	8	External decimal	Time the message was input on the terminal, in the format of *hhmmss*`00`: *hh*: Hour *mm*: Minute *ss*: Second `00`: Fixed
Reason code	112	108	4	External decimal	Reason code is set here
Reserved	116	112	12	--	--

Legend:

--: Not applicable.

Table 7-2: Reason codes for ERREVT4

Reason codes in COBOL language (external decimal)	Reason why ERREVT4 is reported
`0020`	The MHP or SPP could not be activated because of an RPC error or inactive server.
`0030`	Writing to the input queue failed due to insufficient memory.
`0031`	Writing to the input queue failed because the queue file became full.
`0032`	Writing to the input queue stopped because the number of input messages exceeded the specified definition value for the maximum number of input messages to be stored.
`0033`	An error occurred during writing to the input queue.
`0040`	An MHP application is in shutdown status.
`0041`	An MHP application is in secure status.
`0042`	An MHP service or service group is in shutdown status.
`0043`	An MHP service group is in secure status.

Appendix

A. Using OpenTP1 Remote Procedure Calls and XATMI-interfaced API Functions in Combination

This appendix explains how to use `CBLDCRPC('CALL ')`, an OpenTP1-specific interface function, together with the XATMI interface.

Only the OpenTP1-specific interface functions can be used together with the XATMI interface. Note that the TxRPC and XATMI interfaces cannot be used in combination.

A.1 Modes of combined use

There are the following modes of combined use:

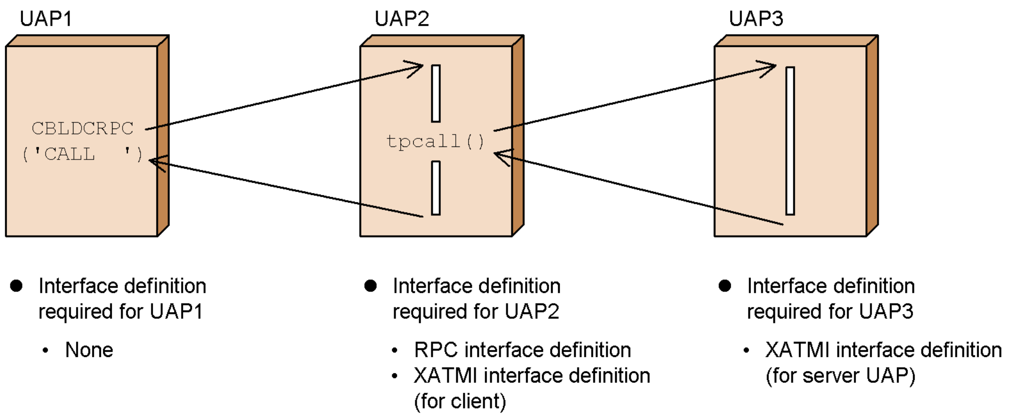
1. When the machine is an OpenTP1 RPC server and an XATMI interface communication client
2. When the machine is an XATMI interface communication server and an OpenTP1 RPC client

In mode 1., specify RPC and XATMI interface definitions for one file when creating a stub, and execute the `stbmake` command or `tpstbmk` command.

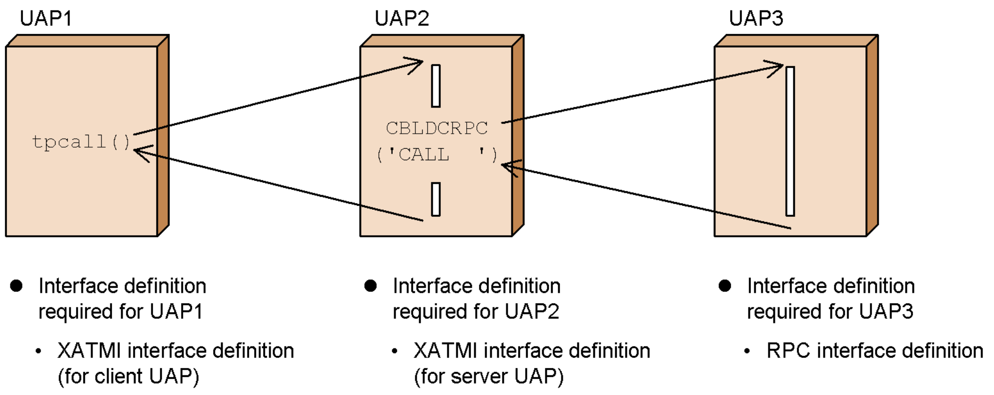
The figure below shows the modes of combined use of inter-process communication and the stubs required.

Figure A-1: Modes of combined use of inter-process communication and the stubs required

1. When the computer is an OpenTP1 RPC server and an XATMI interface communication client


UAP1
UAP2
UAP3
CBLDCRPC ('CALL ')
tpcall()
Interface definition required for UAP1
None
Interface definition required for UAP2
RPC interface definition
XATMI interface definition (for client)
Interface definition required for UAP3
XATMI interface definition (for server UAP)

2. When the computer is an XATMI interface communication server and an OpenTP1RPC client


UAP1
UAP2
UAP3
tpcall()
CBLDCRPC ('CALL ')
Interface definition required for UAP1
XATMI interface definition (for client UAP)
Interface definition required for UAP2
XATMI interface definition (for server UAP)
Interface definition required for UAP3
RPC interface definition

A.2 How to create stubs of application programs that use both OpenTP1 remote procedure calls and XATMI-interfaced API functions

This section explains how to create the stubs of UAPs that are called from `CBLDCRPC('CALL  ')` and call XATMI interface functions (TPCALL, etc). To create the UAP:

1. Create an interface definition file.

   For the file to be created, specify the RPC and XATMI interface definitions (for the client). Suffix the file name with `.def`.

2. Execute the `stbmake` command or `tpstbmk` command.

   Specify the required arguments for the `stbmake` command, and execute the command. Execution of the command creates the declaration files listed below. `xxxxx` indicates a character string of an interface definition file name from which `.def` is excluded.

   - OpenTP1 RPC stub source file (default file name: *xxxxx*`_sstb.c`)
   - XATMI stub source file (default file name: *xxxxx*`_stbx.c`)
   - XATMI stub header file (default file name: *xxxxx*`_stbx.h`)
   - XATMI stub copy file (the file name consists of the subtype name followed by `.cbl`)

   If the RPC interface definition and XATMI interface definition coexist, the XATMI stub source file, XATMI stub header file and XATMI stub copy file are created.

3. Compile the stub source files and link them with a UAP.

   Compile the source files created in step 2. with the C compiler, and link them with a UAP.

A.3 Callable XATMI-interfaced API functions

Table A-1 lists XATMI-interfaced API functions that can be used by an SPP called by `CBLDCRPC('CALL  ')`. The stubs explained in Appendix A.2 must have been linked with the SPP that called these functions.

Table A-1: XATMI-interfaced API functions that can be used by an SPP called by the function dc_rpc_call()

XATMI interface APIs	Call
`TPACALL`	Y
`TPADVERTISE`	--
`TPCALL`	Y
`TPCANCEL`	Y
`TPCONNECT`	Y
`TPDISCON`	Y
`TPGETRPLY`	Y

XATMI interface APIs	Call
TPRECV	Y
TPRETURN	
TPSEND	Y
TPSVCSTART	--
TPUNADVERTISE	--

Legend:

Y: Can be called.

--: Cannot be called.

Index

Reader's Comment Form

We would appreciate your comments and suggestions on this manual. We will use these comments to improve our manuals. When you send a comment or suggestion, please include the manual name and manual number. You can send your comments by any of the following methods:

- Send email to your local Hitachi representative.
- Send email to the following address:
  WWW-mk@itg.hitachi.co.jp
- If you do not have access to email, please fill out the following information and submit this form to your Hitachi representative:

Manual name:	
Manual number:	
Your name:	
Company or organization:	
Street address:	
Comment:	

(For Hitachi use)