z/OS
Version 2 Release 4

MVS Programming: Writing Transaction Schedulers for APPC/MVS

SA23-1398-40

Note

Before using this information and the product it supports, read the information in "Notices" on page 97.

This edition applies to Version 2 Release 4 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2019-07-10

Contents

Figures

Tables

About This Book

This book contains two parts. Part 1 gives a brief introduction to the APPC/MVS system services and their use by transaction schedulers. Part 2 gives details about each service, including function, requirements, syntax, linkage information, parameters, and related exit routines.

Who Should Use This Book

This book is for system programmers who write transaction schedulers to use in addition to or instead of the transaction scheduler that APPC/MVS provides. The book assumes the user understands the concepts of APPC/MVS, and can code in one or more high-level languages (HLLs) that APPC/MVS supports. Using this book also requires you to be familiar with the operating system and the services that programs running under it can invoke.

How to Use This Book

This book is one of the set of programming books for MVS. This set describes how to write programs in assembler language or high-level languages, such as C, FORTRAN, and COBOL. For more information about the content of this set of books, see *z/OS Information Roadmap*.

Where to Find More Information

Before using this book, you should be familiar with APPC/MVS application programming and administration information from *z/OS MVS Programming: Writing Transaction Programs for APPC/MVS* and *z/OS MVS Planning: APPC/MVS Management*.

Where necessary, this book references information in other books, using the shortened version of the book title. For complete titles and order numbers of the books for all products that are part of z/OS, see *z/OS Information Roadmap*.

How to send your comments to IBM

We invite you to submit comments about the z/OS® product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead "If you have a technical problem" on page xi.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM® Knowledge Center function
If your comment or question is about the IBM Knowledge Center functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Knowledge Center Support at ibmkc@us.ibm.com.

Feedback on the z/OS product documentation and content
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS MVS Writing Transaction Schedulers for APPC/MVS, SA23-1398-40
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive authority to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Summary of changes for z/OS MVS Programming: Writing Transaction Schedulers for APPC/MVS for Version 2 Release 4 (V2R4)

This information contains no technical changes for this release.

Summary of changes for z/OS MVS Programming: Writing Transaction Schedulers for APPC/MVS for Version 2 Release 3 (V2R3)

This information contains no technical changes for this release.

Part 1. An Introduction to APPC/MVS System Services

APPC/MVS is an implementation of IBM's Advanced Program-to-Program Communication (APPC) in the MVS operating SYSTEM. APPC/MVS allows MVS application programs to communicate on a peer-to-peer basis with other application programs on the same MVS SYSTEM, different MVS systems, or different operating SYSTEMs (including OS/2, OS/400 and VM) in an SNA network. These communicating programs, known as **transaction programs**, together form cooperative processing applications that can exploit the strengths of different computer architectures.

Transaction programs can be scheduled on MVS by the APPC/MVS transaction scheduler or by an alternative transaction scheduler. This book documents the services that an alternative transaction scheduler must issue to interact with APPC/MVS. These services are callable from high-level or assembler language programs that are running in supervisor state or with PSW key 0-7.

Chapter 1. Transaction Scheduler Services in APPC/MVS

APPC/MVS provides a transaction scheduler that initiates and schedules APPC/MVS transaction programs (TPs) in response to inbound requests from other TPs in an SNA network. APPC/MVS also provides system services that let installations use alternative transaction schedulers and assign TPs to run under them. Those system services are applicable to MVS subsystems and other applications that provide their own work schedulers and want to receive work requests from APPC/MVS.

Transaction schedulers must be defined to a logical unit (LU) that represents the point of entry for inbound requests from an SNA network. That definition must be made in an APPCPMxx parmlib member on MVS. The transaction scheduler can then use system services to receive inbound requests that are directed to the LU, and can schedule the appropriate TPs to handle the requests. A transaction scheduler can obtain TP-specific scheduling information from TP profiles that are maintained by system administrators.

A transaction scheduler commonly has direct control over a number of address spaces and schedules its applications into these subordinate address spaces; the use of subordinate address spaces allows a transaction scheduler to access APPC from its own environment for additional performance and function. Each transaction scheduler may have its own term for these subordinate address spaces; for example, the APPC/MVS transaction scheduler refers to them as transaction initiators.

See Figure 1 on page 3 for an overview of how multiple transaction schedulers and their subordinate address spaces operate under APPC/MVS. When APPC/MVS receives an inbound allocate request for a particular LU, it sends a message describing the request to the associated transaction scheduler. That scheduler can then schedule the appropriate transaction program into a subordinate address space to process the request. A transaction scheduler can also process inbound allocate requests within its own address space.

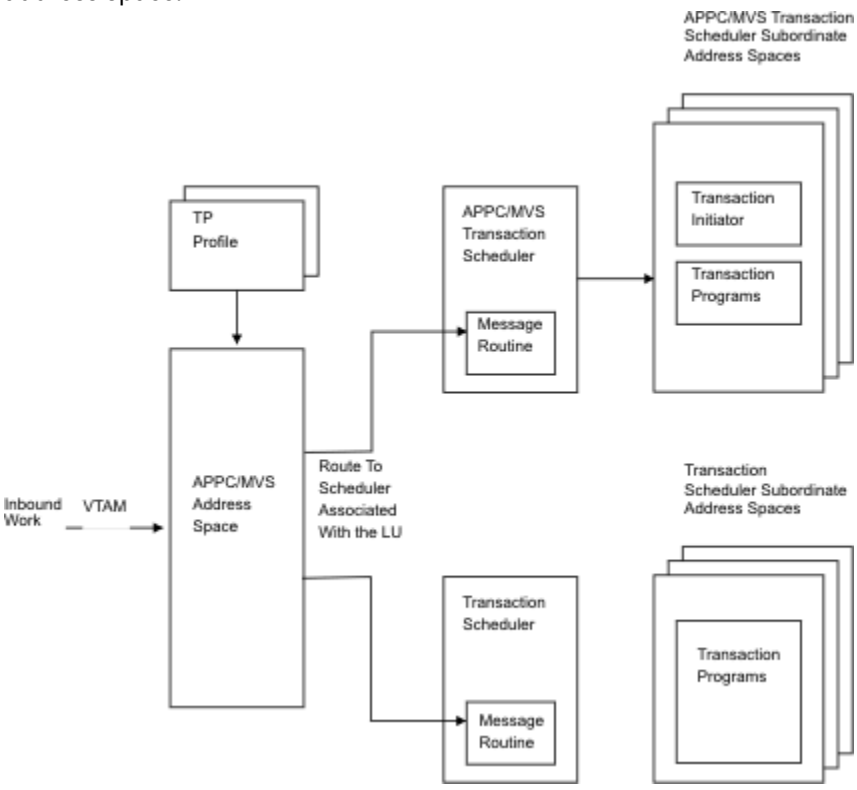


Figure 1. Transaction Program Routing

The following transaction scheduler services are provided by APPC/MVS. See Figure 2 on page 7 for an example of the sequence in which a transaction scheduler calls these services.

Join_Sysappc_Group
A callable service that transaction schedulers and other system applications can use to join the XCF group used by APPC/MVS. Each transaction scheduler must be a member of the APPC XCF group. APPC/MVS notifies all group members of "general interest" events, such as APPC initialization and termination, and notifies individual transaction schedulers of inbound allocate requests for TPs under their control. Transaction schedulers must call the Join_Sysappc_Group service before calling the Identify service.

Identify
A callable service that a transaction scheduler can use to make itself known to APPC/MVS. A transaction scheduler issues Identify after it has initialized itself and is ready to receive or schedule requests from APPC/MVS. The transaction scheduler must supply an XCF member token on Identify to allow APPC/MVS to communicate with it. A transaction scheduler must identify itself to APPC/MVS before its subordinate address spaces can connect to APPC/MVS.

Connect
A callable service that a transaction scheduler can use to inform APPC/MVS that an address space is a subordinate address space of a particular transaction scheduler. The subordinate address space is said to be connected to that transaction scheduler. Connect is required only for transaction schedulers managing one or more subordinate address spaces.

Set_AS_Attributes
A callable service that a transaction scheduler can use to prevent conversations allocated by a subordinate address space from being associated with the system default LU. This service is important in situations where a subordinate address space could allocate an APPC conversation before the transaction scheduler connects the subordinate address space to itself.

Associate
A callable service that a transaction scheduler or subordinate address space can use to associate or relate a particular transaction program instance and its conversations with either the transaction scheduler address space or one of the transaction scheduler's subordinate address spaces. Any previous association established between this TP and another address space is broken. Associate can also be used to provide or change a unit_of_work_id for the transaction program.

Cleanup_TP
A callable service that may be used to request APPC/MVS to clean up all conversation resources associated with a transaction program instance. Conversation resources refers to network resources such as control structures and buffers that are used to manage the transaction program instance and its conversations. This service can be called asynchronously.

Cleanup_Address_Space
A callable service that can be used to request APPC/MVS to clean up all APPC/MVS resources for an address space. APPC/MVS will clean up all conversation resources for all transaction programs associated with the address space at the time Cleanup_Address_Space was issued. This service can be called asynchronously.

Control
A callable service that can be used by a transaction scheduler to control the operational characteristics of a specified LU. Control allows a transaction scheduler to temporarily halt or resume processing of inbound allocate requests received for the LU.

Define_Local_TP
A callable service that can be used by a transaction scheduler to create a new local transaction program instance to be associated with the transaction scheduler address space. A transaction scheduler may wish to create a new transaction program instance so that it can allocate outbound conversations under a transaction program distinct from any inbound transactions it has received. The Define_Local_TP service returns the transaction program identifier (TP_ID), assigned by APPC/MVS, that represents the new transaction program instance just created. This TP_ID can then be passed on the Allocate call or returned by the transaction scheduler extract exit described below.

Disconnect
A callable service that can be used by a transaction scheduler to inform APPC/MVS that an address space is no longer a subordinate address space of a transaction scheduler.

Unidentify
A callable service that can be used by a transaction scheduler to reverse the effect of invocation of the Identify service. Unidentify terminates all APPC services for the specified transaction scheduler and its subordinate address spaces.

After performing Unidentify, a transaction scheduler can issue the IXCLEAVE macro to undo the effects of its invocation of Join_Sysappc_Group.

Chapter 2. General Transaction Scheduler Function: From Start-up to Termination

The following figure is a general example of how a transaction scheduler uses APPC/MVS services. Each number in Figure 2 on page 7 corresponds to a possible step, and to an explanation in the text immediately following the diagram.

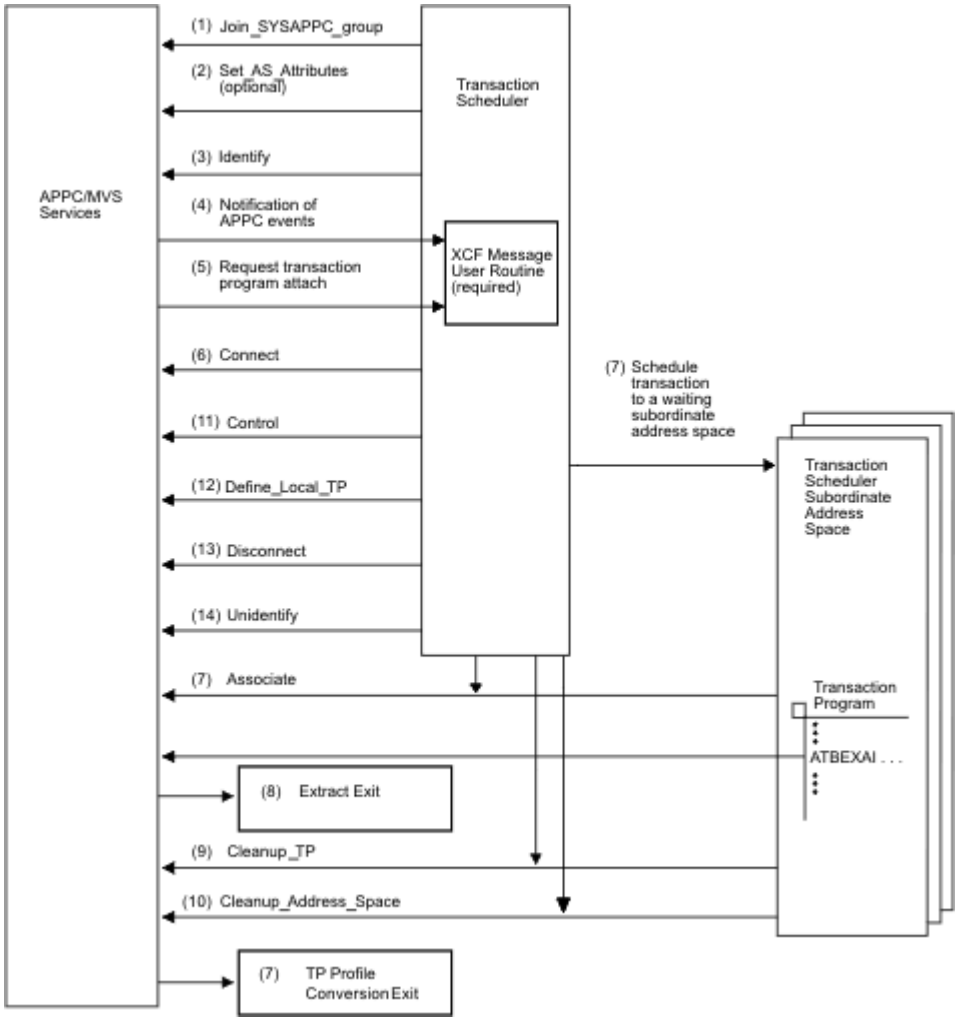


Figure 2. Transaction Schedulers in APPC/MVS

Explanations:

1. Each transaction scheduler must join the APPC XCF group. The transaction scheduler must supply the address of its XCF message user routine, which will receive messages from APPC/MVS. Join_Sysappc_Group returns a value to be used as the member token parameter of Identify. A transaction scheduler must invoke Join_Sysappc_Group before invoking Identify.
2. The transaction scheduler optionally calls the Set_AS_Attributes service to prevent conversations allocated by a subordinate address space from being associated with the system default LU. This prevention takes effect in cases when the subordinate address space is not connected to a transaction scheduler.
3. The transaction scheduler issues the Identify service to make itself known to APPC/MVS. The Identify service indicates to APPC/MVS that the transaction scheduler is fully operational and is ready to

receive and schedule requests from APPC/MVS. The Identify service also identifies possible exit routines for TP profile conversion and information extraction.

4. APPC/MVS notifies all members of the APPC group of significant events, such as APPC initialization and termination.
5. When an inbound allocate request is received, APPC/MVS performs checking, obtains related data, and sends an XCF message requesting the transaction scheduler to attach the TP. The transaction scheduler's XCF message user routine must recognize that the message describes the TP to be attached, and process it.
6. Any time a new subordinate address space is created, the transaction scheduler must issue Connect on behalf of the subordinate address space to inform APPC/MVS that the newly created subordinate address space is owned by the transaction scheduler. Connect must be issued for the subordinate address space before any other APPC/MVS services are used in that address space. Connect is only required for a transaction scheduler managing one or more subordinate address spaces.
7. When a transaction scheduler receives a transaction request from APPC/MVS, the transaction scheduler has the option of passing that work on to a subordinate address space, or of processing the work itself. If the transaction scheduler decides to pass the transaction on to a subordinate address space to process, the transaction scheduler or subordinate address space must invoke the Associate service. If the scheduler specifies a TP profile conversion exit on the Identify service, and the requested transaction has a TP profile entry that requires conversion, the exit is invoked to convert the entry. APPC/MVS then saves the converted copy for future requests, avoiding repeated conversion.
8. After the TP starts running, it might request information about the environment in which it was scheduled. The transaction program can invoke the APPC/MVS Extract_Information service to obtain information about its environment. The Extract_Information service will use the transaction scheduler extract exit to obtain the information to return to the transaction program.
9. Cleanup_TP can be called to cleanup a TP after it is processed, or reject an inbound TP that cannot be processed (for example, because the TP was not available). If the TP is rejected, you can use Cleanup_TP to send error log data to the partner system or TP that submitted the request.
10. Cleanup_Address_Space can be issued to clean up all transaction programs in an address space. The transaction scheduler will probably issue Cleanup_Address_Space after each transaction program completes in a subordinate address space. Cleanup_Address_Space can be used to clean up conversations normally in error situations. In error situations, all conversations are terminated, and the partner TPs are notified of termination with a Deallocate_ABEND_SVC sense code.
11. The transaction scheduler can use the Control request to temporarily halt or resume processing of inbound allocate requests received for a specific LU or all LUs for that transaction scheduler.
12. The transaction scheduler can issue Define_Local_TP to create a TP instance in its address space.
13. Disconnect is the opposite of Connect and is issued by the transaction scheduler to inform APPC/MVS that the specified address space is no longer under the control of the transaction scheduler. Address space termination is an automatic Disconnect.
14. The transaction scheduler issues an Unidentify request to terminate use of APPC/MVS services. Unidentify is the opposite of Identify.

Additional Considerations:

- The transaction scheduler can also issue IXCLEAVE to undo the effects of Join_Sysappc_Group. IXCLEAVE is an XCF macro that disassociates a member from its XCF group (in this case, the APPC XCF group). The input to the IXCLEAVE macro is the member token that was passed back from Join_Sysappc_Group. For more information about the IXCLEAVE macro, including syntax, see *z/OS MVS Programming: Sysplex Services Reference*.
- The transaction scheduler may process protected conversations (conversations with a synchronization level of syncpt). To do so, you must:
  - Define it to an LU that is capable of handling conversations with a synchronization level of syncpt. See the session management section of *z/OS MVS Planning: APPC/MVS Management* for further information about enabling LUs for protected conversations support.

  - Be aware of changes for the following APPC/MVS system services:
    - Cleanup_Address_Space (all versions)
    - Cleanup_TP (all versions)
    - Control
    - Unidentify
- Additionally, the transaction scheduler may register as a resource manager of protected conversations. Doing so allows the scheduler to obtain a privately managed context that it can use to associate with the inbound Allocate request.

  Note that designing and coding a scheduler to act as a resource manager is relatively difficult. If you want to code an alternate transaction scheduler to manage the contexts for protected conversations, you need to understand the concepts and requirements for resource recovery in *z/OS MVS Programming: Resource Recovery*. Design the scheduler to use APPC/MVS system services, along with registration and context callable services, in the following sequence:

  1. Join the APPC XCF group by calling the Join_Sysappc_Group service. Optionally, call the Set_AS_Attributes service.
  2. Register through the Register_Resource_Manager service, supplying a resource manager name. The service returns a resource manager token that the scheduler uses on subsequent calls to registration and context services.
  3. Use the Identify service to identify itself to APPC/MVS; on this service call, provide the resource manager name. After receiving an inbound Allocate request for the LU associated with this scheduler, APPC/MVS creates a privately managed context, and passes the context token to the scheduler through an XCF message.
  4. Change to the correct state to call context services. To do this, call the Set_Exit_Information service to cause the server resource manager state to change to SET state. The server is now in the correct state with context services to issue context callable services.
  5. Switch to the context passed through the XCF message, by issuing a call to the Switch_Context service. After the service returns, the privately managed context is the current context.
  6. Receive the inbound protected conversation by issuing the Get_Conversation service. As part of processing this service, APPC/MVS expresses interest in the unit of recovery under the privately managed context, and sets the logical unit of work identifier (LUWID) for the current context.

  Depending on the design of the scheduler routines, either the scheduler or its subordinate address space invoke the Associate service, and perform steps “4” on page 9 through “6” on page 9. If the scheduler uses subordinate address spaces, it must pass to them the context token for the privately managed context.

Part 2. APPC/MVS System Services Reference

This section describes the features and usage requirements of the APPC/MVS SYSTEM services. System programmers coding authorized programs in high-level languages or assembler can use these callable services to obtain the SYSTEM services they need. This section includes detailed information—such as the function, syntax, linkage information, and parameters— needed to use the SYSTEM services.

Chapter 3. Invocation Details for APPC/MVS System Services

APPC/MVS system services provide access to system services not normally used by transaction programs, but used by other MVS components, management subsystems, and transaction schedulers. These system services have a standard set of syntax and linkage requirements as well as parameter specification details necessary for successful invocation.

Syntax Conventions for the System Services

All APPC/MVS system services have a general calling syntax as follows:

```
CALL routine_name        (parameters,return_code)
```

The specific format for invoking APPC/MVS callable services through the assembler CALL macro is:

```
CALL routine_name,(parm1,parm2,..return_code),VL
```

Linkage Conventions for the System Services

Callers must also use the following linkage conventions for all APPC/MVS system services:

- Register 1 must contain the address of a parameter list, which is a list of consecutive words, each containing the address of a parameter to be passed. The last word in this list must have a 1 in the high-order (sign) bit.
- Register 13 must contain the address of an 18-word save area.
- Register 14 must contain the return address.
- Register 15 must contain the entry-point address of the service being called.
- If the caller is running in AR ASC mode, access registers 1, 13, 14, and 15 must all be set to zero.

On return from the service, general and access registers 2 through 14 are restored (registers 0, 1, and 15 are not restored).

Any high-level language that generates this type of interface can be used to invoke APPC/MVS callable services.

Two methods can be used to access the APPC/MVS system services.

- The ATBCSS module from SYS1.CSSLIB can be link-edited with any program that issues APPC/MVS system services.
- A program can issue the MVS LOAD macro for the APPC/MVS system service to obtain its entry-point address, and then use that address to call the APPC/MVS system service.

Parameter Description for Callable Services

All the parameters of the APPC/MVS callable system services are required positional parameters. When you invoke a service, you must specify all the parameters in the order listed. APPC/MVS checks all parameters for valid values, regardless of whether the parameters are used in call processing. Even though a language may allow parameters to be omitted, APPC/MVS services do not.

Note: Some parameters do not require values and allow you to substitute zeros or a string of blanks for the parameter. The descriptions of the parameters identify those that can be replaced by blanks or zeros, and when to do so.

In the descriptions of services in this document, each parameter is described as supplied or returned:

Supplied
You supply a value for the parameter in the call.

Returned
The service returns a value in the named parameter when the call is finished (for example, **return_code**).

Each parameter is also described in terms of its data type, character set, and length:

Data type
Either address, character string, integer, pointer, or structure.

Character set
Applies only to parameters whose values are character strings and governs the values allowed for that parameter. Possible character sets are:

- No restriction

  There is no restriction on the byte values contained in the character string.

- Type A EBCDIC

  The string can contain only uppercase alphabetics, numerics, and national characters (@, $, #), and must begin with an alphabetic or national character. Use of @, $, and # is discouraged, because those characters display differently on different national code pages.

- 01134

  The string can contain uppercase alphabetics or numerics, with no restriction on the first character.

- 00640

  The string can contain upper- or lowercase alphabetics, numerics, or any of 19 special characters with no restriction on the first character.

  Note: APPC/MVS does not allow blanks in 00640 character strings.

For more detailed information about the characters in each character set, see Appendix A, “Character sets,” on page 65.

Length
Depends on the data type of the parameter:

- For an address, integer, or pointer, the length indicates the size of the field in bits.
- For a character-string parameter, the length value indicates the number of characters that can be contained in a character type parameter. The length can specify a single number or a minimum and maximum number.
- For a structure parameter, the length value indicates the size of the structure in bytes, or a minimum and maximum size if the size of the structure is variable.

Versions of Callable Services

Some APPC/MVS calls have a version number as the last character of the call name (for example, ATBIDN1). That number corresponds to the version of APPC/MVS in which the call was introduced.

To determine which calls are valid on a system, you can obtain the current APPC/MVS version number from the APPC/MVS Version service. On any system, valid APPC/MVS calls include those with no version number in the call name or a version number less than or equal to the current APPC/MVS version number. For example, calls to ATBIDEN and ATBIDN1 are both valid when the current APPC/MVS version number

is 1 or higher. Likewise, a call named ATBxxx2 would be valid only when the current APPC/MVS version number is 2 or higher.

For more information about APPC/MVS version numbers, including how to obtain the version number that is current on your system, see the Version service in *z/OS MVS Programming: Writing Transaction Programs for APPC/MVS*.

Interface Definition File (IDF)

APPC/MVS provides an IDF (also called a pseudonym file) that defines variables and values for parameters of APPC/MVS system services. The IDF can be included or copied from a central library into programs that invoke APPC/MVS callable services.

For APPC/MVS system services, the IDF for assembler language programs is the ATBCSASM member of SYS1.MACLIB.

Chapter 4. APPC/MVS System Services Summary

This chapter describes the specific system services available in APPC/MVS. The function, invocation requirements, parameters, and other detailed information are explained separately for each system service.

Callers of these system services must be in supervisor state or have a PSW key of 0-7. Callers that are not in supervisor state or do not have PSW key 0-7 end with system completion (abend) code 0C2, with the exceptions of the Join_Sysappc_Group and Set_Address_Space_Attributes services, which provide a return code.

The ATBCSS module from SYS1.CSSLIB must be link-edited with any program that issues these services.

The following table lists the system services that have more than one associated call name. This chapter describes the current versions of the calls, which are the preferred programming interfaces for these services. The previous versions are described in Appendix B, “Previous Versions of APPC/MVS System Services,” on page 69.

Table 1. APPC/MVS System Callable Services

Service Name	Previous Call Name	Current Call Name	Reference for Current Call
Cleanup_Address_Space	ATBCMAS	ATBCAS1	“Cleanup_Address_Space” on page 21
Cleanup_TP	ATBCMTP, ATBCTP1	ATBCTP3	“Cleanup_TP” on page 24
Identify	ATBIDEN, ATBIDN1	ATBIDN4	“Identify” on page 38
Join_Sysappc_Group	ATBMIGRP	ATBJGP1	“Join_Sysappc_Group” on page 44
Unidentify	ATBUNID	ATBUID1	“Unidentify” on page 49

Associate

Use the Associate service to associate a particular transaction program and its conversations with either the transaction scheduler address space or one of the transaction scheduler's subordinate address spaces. Any previous association established between this TP and another address space is broken.

When a transaction scheduler receives an inbound allocate request from APPC/MVS, the targeted TP is automatically associated with the transaction scheduler. If the transaction scheduler passes that work to an awaiting subordinate address space, the transaction scheduler or subordinate address space must invoke the Associate service. If you do not use the Associate service for TPs running in subordinate address spaces, APPC/MVS cannot clean up conversation resources when the subordinate address space is terminated.

You can associate multiple transaction programs with a transaction scheduler, but you can only associate one transaction program with a subordinate address space at a time. A transaction scheduler is responsible for ensuring the integrity of TPs that run at the same time in the transaction scheduler's address space.

You can also use the Associate service to provide or change a unit_of_work_id for the transaction program.

Environment

Authorization:	Supervisor state or PSW key 0-7

Dispatchable unit mode:	Task or SRB mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBASOC (TP_ID,
              Current_ASCB_ptr,
              New_ASCB_ptr,
              Unit_of_work_id,
              Return_code
             );
```

Figure 3. ATBASOC - Associate

Parameters

TP_ID
Supplied/Returned parameter

- Type: Character string
- Char Set: No restriction
- Length: 8 bytes

Specifies the transaction program instance assigned to this transaction by APPC/MVS. The TP_ID is a token that uniquely identifies an instance of a program using APPC/MVS services. The TP_ID is passed to the transaction scheduler in the inbound allocate request message. The TP_ID is also generated when a program calls Define_Local_TP or is not started through APPC and calls an allocate service.

A zero TP_ID can be specified if the Current_ASCB_ptr points to a subordinate address space. A zero TP_ID specifies that the transaction program instance in the current address space is to be associated to the new address space. If the Current_ASCB_ptr points to the transaction scheduler address space (that is, a transaction scheduler address space is the one that called Identify), a zero TP_ID will not be allowed.

A transaction scheduler cannot have the TP_ID for a locally started transaction in a subordinate address space. In this situation, the transaction scheduler sets this value to zero, and APPC/MVS sets the TP_ID upon return to the caller.

Current_ASCB_ptr
Supplied parameter

- Type: Pointer
- Char Set: N/A
- Length: 32 bits

Specifies the pointer to the address space control block (ASCB) of the address space where the transaction program currently resides.

New_ASCB_ptr
Supplied parameter

- Type: Pointer
- Char Set: N/A

- Length: 32 bits

Specifies the pointer to the address space control block (ASCB) of the new address space to associate with the transaction program instance.

Unit_of_work_id
Supplied parameter

- Type: Character string
- Char Set: 01134
- Length: 8 Bytes

Specifies an ID assigned to this program instance by the transaction scheduler (for example, a job number or transaction code). This is an optional parameter used only in APPC/MVS diagnostics. It correlates APPC activity to program instances as they are known in APPC/MVS to program instances as they are known to other components and subsystems. To specify no Unit_of_work_ID, set the parameter to 8 blanks. If Unit_of_work_id is specified, TP_ID must also be specified.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Associate may return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
The transaction program association is successful.

8
The specified TP_ID does not exist.

12
Associate failed; this address space is already associated with another TP_ID.

16
The Current_ASCB_ptr is a transaction scheduler address space and not a subordinate address space. A TP_ID of zero cannot be specified for a transaction scheduler address space.

20
The value specified on the New_ASCB_ptr parameter is not valid.

24
The value specified on the Current_ASCB_ptr parameter is not valid.

28
The transaction program to be associated has an active APPC request outstanding.

30
The combination of parameters is not valid.

32
The requested service is not supported in the caller's environment. For example, this return code will be given if the caller invokes any of the transaction scheduler services while holding a lock.

36
The requested transaction scheduler service must be invoked from the transaction scheduler address space or from a transaction scheduler subordinate address space.

38
The specified program is an APPC/MVS server. It cannot be associated with another address space.

44
APPC/MVS is not active.

48
APPC/MVS services failure.

Characteristics and Restrictions

1. The caller does not have to reside in the current or new associated address space of the TP instance.
2. This service cannot be called while there is an APPC request outstanding from the transaction program instance (TP_ID) specified on the call. If the Associate service is called while there is an outstanding APPC request, the system does not perform the Associate service function, and the caller receives a return code of 28 (decimal).
3. You cannot associate a TP that's currently registered for an allocate queue (through the Register_for_Allocates service). If the Associate service is called while the TP is registered for an allocate queue, the system does not perform the Associate service function, and the caller receives a return code of 38 (decimal). For more information about the Register_for_Allocates service, see *z/OS MVS Programming: Writing Servers for APPC/MVS*.
4. The new address space specified on the New_ASCB_ptr parameter cannot be a subordinate address space that is currently running a TP. If the new address space is a subordinate address space that is running a TP, the system does not perform the Associate service function, and the caller receives a return code of 12 (decimal).
5. Transaction schedulers that call the Associate service while in task mode should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the section on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.
6. Two process identifiers (TP_ID and New_ASCB_ptr) are supported to accommodate different types of Associate scenarios. Table 2 on page 20 describes the action taken by APPC/MVS based upon how all these parameters are specified on the Associate service.

Table 2. Relationship between TP_ID and address space parameters

TP_ID	Current ASCB_ptr	New ASCB_ptr	Action taken by APPC/MVS
0	0	0	Parameters not valid - Associate will give return code 30.
0	variable	0	The transaction program residing in the subordinate address space is associated with the home address space of the caller. If the transaction program does not reside in a subordinate address space, Associate gives a return code of 16. If no active transaction program exists, Associate gives a return code of 8.
0	0	variable	The transaction program residing in the home address space of the caller is associated with the new address space identified. If the transaction program does not reside in a subordinate address space, Associate gives a return code of 16. If no active transaction program exists, Associate gives a return code of 8.

Table 2. Relationship between TP_ID and address space parameters (continued)			
TP_ID	**Current ASCB_ptr**	**New ASCB_ptr**	**Action taken by APPC/MVS**
0	variable	variable	The transaction program residing in the current subordinate address space identified is associated with the new address space identified. If the transaction program does not reside in a subordinate address space, Associate gives a return code of 16. If no active transaction program exists, Associate gives a return code of 8.
variable	-	0	The specified TP_ID is associated with the home address space of the caller.
variable	-	variable	The specified TP_ID is associated with the new address space specified by New_ASCB_ptr.

Cleanup_Address_Space

You can use the Cleanup_Address_Space service to clean up all APPC/MVS resources for an address space. APPC/MVS cleans up all conversation resources for all transaction programs that are associated with the address space at the time the Cleanup_Address_Space is issued.

The Cleanup_Address_Space service may be invoked by a transaction scheduler subordinate address space for a transaction program or job that terminates normally or abnormally.

APPC/MVS deletes one or more TP_IDs from the system as a result of this call; this cleanup process might occur asynchronously.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task or SRB mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBCAS1 (ASCB_ptr,
              Condition,
              Notify_Type,
              Return_code
             );
```

Figure 4. ATBCAS1 - Cleanup_Address_Space Service

Parameters

ASCB_ptr
Supplied parameter

- Type: Pointer
- Char Set: N/A
- Length: 32 bits

Specifies the pointer to the address space control block (ASCB) for the address space to be cleaned up. All conversations for all transaction program instances associated with this address space are to be deallocated. Invokers of this service can get this value from the PSAAOLD field in the PSA for the current address space or from the RMPLASCB field in the resource manager parameter list (RMPL). If this parameter is set to zero, the home address space of the program that issued the Cleanup_Address_Space call will be used as the default address space.

Condition
Supplied parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Specifies the deallocation condition that has occurred. This field is used to generate the TYPE of deallocate and sense code that is issued by APPC/MVS to the partner transaction program.

Valid values for this parameter are:

Value
Meaning

0
Normal

Specifies that the transaction program completed normally, even though it may have left active conversations. APPC/MVS deallocates all conversations in a proper state for normal deallocation with Deallocate Type(Sync_Level). All conversations not in the proper state for a normal deallocation are deallocated with Type(Abend_SVC).

1
System

Specifies that the transaction program terminated abnormally, or the transaction program was terminated on behalf of some action by the system (for example, the address space was cancelled or forced). This condition is normally detected by transaction scheduler's subordinate address space.

All active conversations are deallocated with Type(Abend_SVC).

Notify_type
Supplied parameter

- Type: Structure
- Char Set: N/A
- Length: 4-8 bytes

Specifies the type of processing and notification (synchronous or asynchronous) requested for this service. Programs can request asynchronous processing, which returns control to the program immediately and later notifies the program by ECB when the service is complete. The possible types are:

- None

  No notification is requested. The service is performed synchronously, and control is returned to the caller when processing is complete. All returned parameters are set on return to the caller. To specify no notification, set the parameter value to a four-byte structure containing binary zeros.
- ECB

Programs can request asynchronous processing by specifying an ECB to be posted when processing completes. To specify an ECB, set the parameter to an eight-byte structure containing a fullword binary one (X'00000001') followed by the address of a fullword area to be used as the ECB. The ECB must reside in the home address space.

When you specify an ECB, control is returned before processing is complete, with only the return code set. If the asynchronous request was accepted, the return code is set to 0 to indicate that the service is being processed asynchronously. Other returned parameters are filled in during asynchronous processing, and the specified ECB is posted when all returned parameters are set. The completion code field in the ECB contains the return code for the service.

Return_code

Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Cleanup_Address_Space may return one of the following decimal values in the return code parameter:

Decimal	Meaning
0	Request accepted. All conversations owned by the address space are cleaned up asynchronously.
4	No conversations exist to be cleaned up.
8	The ASCB_ptr supplied does not point to a valid ASCB.
12	The asynchronous request failed. Resubmit the request with a Notify_type of None or report the problem to IBM.
20	APPC/MVS was cancelled during an asynchronous request for this service.
32	The requested service is not supported in the caller's environment. For example, this return code will be given if the caller invokes any of the transaction scheduler services while holding a lock.
44	APPC/MVS is not active.
48	APPC/MVS services failure.

Characteristics and Restrictions

1. Conversations with active APPC requests are not immediately deallocated. Once the partner TP responds, APPC/MVS returns a deallocate condition and deallocates the conversation locally.
2. Cleanup_Address_Space may access fields located through the ASCB_ptr parameter before it establishes recovery, to improve performance in the case where no APPC resources must be cleaned up. If an incorrect ASCB_ptr is passed to Cleanup_Address_Space, the caller might abnormally end with completion code 0C4 when Cleanup_Address_Space uses the passed value to get addressability to fields in the ASCB.
3. The Condition parameter defaults to 0 (normal) if an invalid condition is specified.
4. If you call the Cleanup_Address_Space service while a unit of work is waiting on an ECB as a result of an asynchronous call, APPC/MVS does not post the ECB after performing the Cleanup_Address_Space operation (APPC/MVS considers all resources associated with the address space “terminated”). The application's recovery environment must clean up the waiting ECB.

5. Transaction schedulers that call the Cleanup_Address_Space service while in task mode should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.
6. Regardless of the condition parameter value specified for this service, APPC/MVS cleans up protected conversations differently, depending on whether a syncpoint operation is in progress. When a syncpoint operation **is** in progress for the current UR for the context with which the protected conversation is associated, APPC/MVS does not immediately deallocate the conversation. The syncpoint operation is allowed to complete. As part of the syncpoint processing, the protected conversation might be deallocated, in which case no further cleanup is required for that conversation.

   If the conversation was not deallocated, however, cleanup processing proceeds in the same manner as it does when a syncpoint operation **is not** in progress at the time the Cleanup service is issued:

   - The protected conversation is deallocated with TYPE(ABEND_SVC).
   - The current UR is put into backout-required state.
   - If the protected conversation is an inbound conversation, the logical unit of work ID (LUWID) for the next UR is reset.
   - The current UR and subsequent units of recovery for the context will not include the protected conversation being cleaned up by this service.

Cleanup_TP

Cleanup_TP is used to request that APPC/MVS clean up all conversation resources associated with a transaction program instance. Conversation resources include network resources, control blocks, and buffers that are used by APPC/MVS to manage the transaction program instance and its conversations.

Call Cleanup_TP for one of the following reasons:

- The TP requested by an inbound allocate request is not recognized or not available.
- The transaction scheduler cannot queue or schedule the transaction program at this time.
- The requesting userid is not authorized to use the transaction program
- The TP was attached and executed, and has completed normally or abnormally.

The TP_ID is deleted from the system as a result of this call; this cleanup process may occur asynchronously.

When calling Cleanup_TP, you can send *error log information* to a partner TP or system. See "Sending Error Log Information" on page 29 for more information.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task or SRB mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBCTP3 (TP_ID,
              Condition,
              Notify_Type,
              Error_Log_Information_Length,
              Error_Log_Information,
              Return_Code
             );
```

Figure 5. ATBCTP3 - Cleanup_TP

Parameters

TP_ID
Supplied parameter

- Type: Character string
- Char Set: No restriction
- Length: 8 bytes

Specifies the transaction program instance that is to be cleaned up. The transaction program instance does not have to be associated with the caller's address space. All conversations owned by this transaction program instance are to be deallocated.

Condition
Supplied parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Specifies the deallocation condition that has occurred. This field is used to determine the type of deallocate and sense code that is issued by APPC/MVS to the partner transaction program.

Note: If you specify a value of zero on the Condition parameter, you cannot send error log information to partner TPs or systems.

Valid values for this parameter are:

Value
Meaning

0
Normal

Specifies that the transaction program completed normally, even though it might have left active conversations. APPC/MVS deallocates all conversations in a proper state for normal deallocation with Deallocate Type(Sync_Level). All conversations not in the proper state for a normal deallocation are deallocated with Type(Abend_SVC).

1
System

Specifies that the transaction program terminated abnormally, or the transaction program was terminated on behalf of some action by the system (for example, the address space was cancelled or forced). This condition is normally detected by transaction scheduler's subordinate address space. All active conversations are deallocated with TYPE(Abend_SVC).

2
TP_Not_Available_No_Retry

Specifies that the transaction scheduler is not able to schedule the transaction because of a condition that is not temporary. The partner should not attempt to retry the request. APPC/MVS deallocates the conversation with a sense code of X'084C0000'.

3
TP_Not_Available_Retry

Specifies that the transaction scheduler is not able to schedule the transaction because of a condition that might be temporary. The partner can attempt to retry the request. APPC/MVS deallocates the conversation with a sense code of X'084B6031'.

4
TPN_Not_Recognized

Specifies that the transaction scheduler does not recognize the TP_Name passed to it. APPC/MVS deallocates the conversation with a sense code of X'10086021'.

5
Security_Not_Valid

Specifies that the transaction scheduler detected a security violation. APPC/MVS deallocates the conversation with a sense code of X'080F6051'.

6
Sync_Level_Not_Supported_Pgm

Specifies that the transaction program does not support the level of synchronization requested by the sender. APPC/MVS deallocates the conversation with a sense code of X'10086041'.

7
User_Not_Authorized_For_TP

Specifies that the user is not authorized to access the transaction program. APPC/MVS deallocates the conversation with a sense code of X'080F0983'.

Notify_type
Supplied parameter

- Type: Structure
- Char Set: N/A
- Length: 4-8 bytes

Specifies the type of processing and notification (synchronous or asynchronous) requested for this service. Programs can request asynchronous processing, which returns control to the program immediately and later notifies the program by ECB when the service is complete. The possible types are:

- None

  No notification is requested. The service is performed synchronously, and control is returned to the caller when processing is complete. All returned parameters are set on return to the caller. To specify no notification, set the parameter value to a four-byte structure containing binary zeros.
- ECB

  Programs can request asynchronous processing by specifying an ECB to be posted when processing completes. To specify an ECB, set the parameter to an eight-byte structure containing a fullword binary one (X'00000001') followed by the address of a fullword area to be used as the ECB. The ECB must reside in the home address space.

  When you specify an ECB, control is returned before processing is complete, with only the return code set. If the asynchronous request was accepted, the return code is set to 0 to indicate that the service is being processed asynchronously. Other returned parameters are filled in during asynchronous processing, and the specified ECB is posted when all returned parameters are set. The completion code field in the ECB contains the return code for the service.

Error_log_information_length
Supplied parameter

- Type: Integer

- Char Set: N/A
- Length: 4 bytes

Specifies the length of the error log information specified on the Error_log_information parameter:

- If error log information **is not** to be sent, specify 0 on this parameter.
- If error log information **is** to be sent, specify the number of bytes of error log information provided, in the range 1-512 (decimal).

If you specify a value greater than 512 on the Error_log_information_length parameter, the system returns return code 16 (decimal) to the caller.

Error_log_information
Supplied parameter

- Type: Character string
- Char Set: N/A
- Length: 0-512 bytes

Specifies error log information to be sent to all partner systems running TPs that have established conversations with the TP to be cleaned up. This parameter contains information about an error that occurred while scheduling a TP. The scheduler can send error log information only when the Condition parameter (for the Cleanup_TP service) specifies one of the following values:

- System
- TP_Not_Available_No_Retry
- TP_Not_Available_Retry
- TPN_Not_Recognized
- Security_Not_Valid
- Sync_Level_Not_Supported_Pgm
- User_Not_Authorized_For_TP

If you do not specify one of the above values on the Condition parameter, Cleanup_TP does not send error log information, even if it is specified on this parameter.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

When APPC/MVS returns control to your TP, the Return_code parameter contains one of the following decimal return codes:

Return Code	Meaning and Action
0	**Meaning**: Successful completion. All conversations owned by the transaction program instance will be cleaned up asynchronously. **Action**: None required.
4	**Meaning**: No conversations exist to be cleaned up. **Action**: None required.
8	**Meaning**: The TP_ID parameter specified a TP instance that does not exist. **Action**: Specify a valid TP instance on the TP_ID parameter.

Return Code	Meaning and Action
12	**Meaning**: An asynchronous request failed. **Action**: Specify a Notify_Type of None on the call to Cleanup_TP, then submit the request again. If the problem persists, contact the IBM Support Center.
16	**Meaning**: The Error_log_information_length parameter contains a value that is greater than 512 (decimal). The transaction scheduler can only send up to 512 (decimal) bytes of error log information. **Action**: Specify a value between 0 and 512 (decimal) on the Error_log_information_length parameter.
20	**Meaning**: APPC/MVS was cancelled during an asynchronous request for this service. **Action**: Contact the operator to determine if APPC/MVS can be restarted.
32	**Meaning**: The requested service is not supported in the caller's environment. For example, the caller might be holding a lock. **Action**: See the "Environment" section for the required environment for calling Cleanup_TP. Ensure that the scheduler calls Cleanup_TP while running in the required environment.
44	**Meaning**: APPC/MVS is not active. **Action**: Contact the operator to determine if APPC/MVS can be restarted.
48	**Meaning**: APPC/MVS services failure. **Action**: Contact the IBM Support Center.

Characteristics and Restrictions

1. Conversations with active APPC requests are not immediately deallocated. Once the partner TP responds, APPC/MVS returns a deallocate condition and deallocates the conversation locally.
2. The Condition parameter defaults to 0 (normal) if an invalid condition is specified.
3. If you call the Cleanup_TP service while a unit of work is waiting on an ECB as a result of an asynchronous call, APPC/MVS does not post the ECB after performing the Cleanup_TP operation (APPC/MVS considers all resources associated with the TP “terminated”). The application's recovery environment must clean up the waiting ECB.
4. Transaction schedulers that call the Cleanup_TP service while in task mode should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.
5. Regardless of the condition parameter value specified for this service, APPC/MVS cleans up protected conversations differently, depending on whether a syncpoint operation is in progress. When a syncpoint operation **is** in progress for the current UR for the context with which the protected conversation is associated, APPC/MVS does not immediately deallocate the conversation. The syncpoint operation is allowed to complete. As part of the syncpoint processing, the protected conversation might be deallocated, in which case no further cleanup is required for that conversation.

   If the conversation was not deallocated, however, cleanup processing proceeds in the same manner as it does when a syncpoint operation **is not** in progress at the time the Cleanup service is issued:

   - The protected conversation is deallocated with TYPE(ABEND_SVC).
   - The current UR is put into backout-required state.

- If the protected conversation is an inbound conversation, the logical unit of work ID (LUWID) for the next UR is reset.
- The current UR and subsequent units of recovery for the context will not include the protected conversation being cleaned up by this service.

Sending Error Log Information

When calling Cleanup_TP, you can send error log information to a partner TP or system. Error log information describes errors that your scheduler finds when it tries to schedule a TP. Programmers for partner systems can use the information to help diagnose errors in their TPs. For example, Cleanup_TP can send error log information that indicates a partner TP name specified on an inbound allocate request is not acceptable to your scheduler.

The error log information is sent to all systems with TPs that have established conversations with the TP to be cleaned up. If the partner system is MVS, the partner TP can use the Error_Extract service to return the error log information (see *z/OS MVS Programming: Writing Transaction Programs for APPC/MVS* for information about how to use Error_Extract). If the partner system is **not** MVS, the partner system must determine how to obtain and use the error log information. Your can specify error log information for both basic and mapped conversations.

To send error log information to a partner TP or system, you must specify a value other than Normal on the Condition parameter for this service. APPC/MVS sends the error log information in a generalized data stream (GDS) variable, and then sends an FMH-7 to notify the partner system that an error occurred. The GDS variable has the format shown in Figure 6 on page 29:

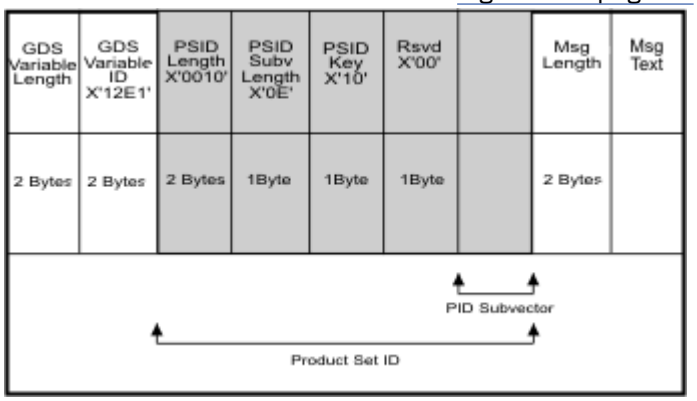
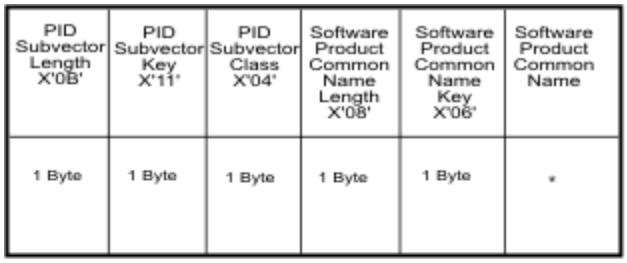


Figure 6. Format of GDS Variable for Sending Log Data

Figure 7 on page 30 shows the format of the product ID subvector in the GDS variable shown in the previous figure.



PID Subvector Length X'0B'	PID Subvector Key X'11'	PID Subvector Class X'04'	Software Product Common Name Length X'08'	Software Product Common Name Key X'06'	Software Product Common Name
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	*

Figure 7. Format of Product ID in GDS Variable

Example

In this example, assume that an alternate scheduler cannot schedule a TP because the user who ran the TP did not have access to a required data set. The scheduler calls Cleanup_TP to clean up the conversation. The scheduler sends a sense code of user_not_authorized_for_TP and error log information to the partner TP:

```
Condition = User_not_authorized_for_TP;
Notify_type = ATB_None;
Error_log_information = ' ';
Error_log_information = 'User does not have access to data set 'user.dsname'';
Error_log_information_length = Length(Error_log_information);
CALL ATBCTP3(TP_ID,
             Condition,
             Notify_type,
             Error_log_information_length,
             Error_log_information,
             Return_code,
             );
```

Connect

The Connect service is used by a transaction scheduler to inform APPC/MVS that an address space is a subordinate address space of a particular transaction scheduler. The subordinate address space is said to be connected to that transaction scheduler. The Connect service must be issued by the same address space that issued the Identify. Connect is only required for a transaction scheduler managing subordinate address spaces.

A connection is required to provide an integrity structure for APPC/MVS. When a transaction scheduler issues an Identify, an implicit Connect is assumed. A transaction scheduler may associate or reassociate transaction programs from one subordinate address space to another. The connection allows APPC/MVS to ensure that TPs attached from one scheduler are always associated with address spaces connected with that scheduler.

A connection also enables APPC/MVS to process outbound Allocate requests from an MVS program. The base LU name of the transaction scheduler associated with the outbound allocate request is defined in the APPCPMxx parmlib member.

An address space remains connected to a particular transaction scheduler until the address space is terminated or issues an explicit Disconnect. (See “Disconnect” on page 36.) Memory termination causes an automatic Disconnect.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBCONN (ASCB_ptr,
              Return_code
             );
```

Figure 8. ATBCONN - Connect Service

Parameters

ASCB_ptr
Supplied parameter

- Type: Pointer
- Char set: N/A
- Length: 32 bits

Specifies the pointer to the address space control block (ASCB) of the address space being connected to the transaction scheduler.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Connect may return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
Address space successfully connected.

4
ASCB_ptr was invalid.

8
Connect was rejected, having specified an address space that already had outstanding APPC conversations or service calls, or an address space that was already connected. You might need to call Cleanup_Address_Space before trying to Connect.

32
The requested service is not supported in the caller's environment. For example, this return code will be given if the caller invokes any of the transaction scheduler services while holding a lock.

34
The requested transaction scheduler service must be invoked from a transaction scheduler address space.

44
APPC/MVS is not active.

48
APPC/MVS services failure.

Characteristics and Restrictions

1. The caller's home address space must be the transaction scheduler address space (that is, the same home address space that issued the Identify).
2. A transaction scheduler must issue Identify before it can issue a Connect.
3. Transaction schedulers that call the Connect service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.

Control

The Control service is used by a transaction scheduler to control the operational characteristics of a specified LU. Control allows a transaction scheduler to temporarily halt or resume processing of inbound Allocate requests received for the LU.

When a transaction scheduler requests that processing be halted for an LU, all subsequent inbound Allocate requests received for that LU are rejected with a sense code of X'084C0000' (TP_Not_Available_No_Retry). However, inbound Allocate requests that have already been received and are being processed will not be halted. Thus, the transaction scheduler can receive inbound Allocate request messages for the LU after Control has been issued. In addition, if the scheduler is processing protected conversations, APPC/MVS continues to accept inbound resynchronization requests for the LU, even after the scheduler issues the Control service. For more information about protected conversations, see Chapter 2, “General Transaction Scheduler Function: From Start-up to Termination,” on page 7.

The LU specified must be assigned to the transaction scheduler requesting the service through the SCHED keyword on the LUADD statement in the APPCPMxx parmlib member.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBCNTL (LU_name,
              Function,
              Return_code
             );
```

Figure 9. ATBCNTL - Control

Parameters

LU_name
Supplied parameter

- Type: Character string
- Char Set: 01134
- Length: 8 bytes

Specifies the name of the LU.

Function
Supplied parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

The Function specifies how the LU operation is to be changed. Valid values for this parameter are:

Value
Meaning

0
Halt_Input

Specifies that APPC/MVS should temporarily halt processing of inbound Allocate requests to the specified LU. The requests are rejected with a sense code of X'084C0000' (TP_Not_Available_No_Retry).

1
Resume_Input

Specifies that APPC/MVS should resume processing of inbound Allocate requests to the specified LU.

2
Halt_All_Input

Specifies that APPC/MVS should temporarily halt processing of Allocate requests to all of the LUs belonging to the transaction scheduler. The requests are rejected with a sense code of X'084C0000' (TP_Not_Available_No_Retry). Only those LUs currently in Active or Outbound_Only state are immediately placed in Outbound_Only state. Those LUs currently in Pending state are eventually placed in Outbound_Only state; the update is not immediate. The state of LUs added by a subsequent SET command will be set to Outbound_Only. (See the LU_Initial_Status parameter of the Identify service for more information.)

3
Resume_All_Input

Specifies that APPC/MVS should resume processing of Allocate requests to all of the LUs belonging to the transaction scheduler. Only those LUs currently in Active or Outbound_Only state are immediately resumed. Those LUs currently in Pending state are eventually placed in Active state; the update is not immediate. The state of LUs added by a subsequent SET command will be set to Active. (See the LU_Initial_Status parameter of the Identify service for more information.)

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Control might return one of the following decimal values in the return code parameter:

Decimal	Meaning
0	Request accepted.
4	Request accepted. One or more requested LUs were not in the appropriate state for the requested function.
8	The LU_name parameter was not valid or was not assigned to the transaction scheduler making the request.
12	The LU is in a state (pending or in_termination) that cannot be changed by this service.
16	The function value specified was not valid.
32	The requested service is not supported in the caller's environment. For example, this return code will be given if the caller invokes any of the transaction scheduler services while holding a lock.
34	The requested transaction scheduler service must be invoked from a transaction scheduler address space.
44	APPC/MVS is not active.
48	APPC/MVS services failure.

Characteristics and Restrictions

Transaction schedulers that call the Control service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.

Define_Local_TP

Define_Local_TP can be used by a transaction scheduler to create a new local transaction program ID (TP_ID) to be associated with the transaction scheduler address space. A transaction scheduler may wish to create a new TP_ID so it can allocate outbound conversations under a TP_ID distinct from any TP_IDs it has received. The Define_Local_TP service will return the TP_ID that represents the new transaction program just created. This TP_ID can then be passed on the Allocate call or returned by the transaction scheduler extract. The Define_Local_TP service can only be used by a transaction scheduler that has identified itself to APPC/MVS.

The Define_Local_TP service gives the transaction scheduler control over defining one or more TP_IDs in the transaction scheduler address space. The transaction scheduler extract exit will be used to resolve ambiguity whenever there is more than one transaction program defined in the address space. See “Extract Exit” on page 57 for more details on this exit.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBDFTP (TP_name_length,
              TP_name,
              LU_name,
              TP_ID,
              Return_code
             );
```

Figure 10. ATBDFTP - Define Local TP

Parameters

TP_name_length
Supplied parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

TP_name_length specifies the length of data contained in the TP_name parameter.

TP_name
Supplied parameter

- Type: Character string
- Char Set: 00640 or Type A
- Length: 1 - 64 bytes

TP_name specifies the name of the local transaction program to be associated with this transaction program instance.

LU_name
Supplied parameter

- Type: Character string
- Char Set: Type A
- Length: 8 bytes

LU_name specifies the name of the LU with which the newly created TP_ID should be associated. This must be an LU that is assigned to the transaction scheduler. If the LU_name parameter specified is all blanks, the base LU, if any, for the transaction scheduler will be used.

TP_ID
Returned parameter

- Type: Character string

- Char Set: N/A
- Length: 8 bytes

TP_ID is a token that represents the transaction program instance that was just created.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Define_Local_TP may return one of the following decimal values in the return code parameter:

Decimal	Meaning
0	Request accepted. The TP_ID is returned in parameter TP_ID.
4	Request rejected. The LU specified was not an LU that is assigned to the transaction scheduler.
8	Request rejected. The TP name is not a valid character string.
32	The requested service is not supported in the caller's environment. For example, this return code will be given if the caller invokes any of the transaction scheduler services while holding a lock.
34	The requested transaction scheduler service must be invoked from a transaction scheduler address space.
44	APPC/MVS is not active.
48	APPC/MVS services failure.

Characteristics and Restrictions

1. The caller must be from a transaction scheduler address space (from the transaction scheduler address space that issued the Identify).
2. If an LU_name of all blanks is specified, and there is no base LU defined for the transaction scheduler, the request will be rejected with return code 4.
3. Transaction schedulers that call the Define_Local_TP service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.

Disconnect

The Disconnect service can be used by a transaction scheduler to inform APPC/MVS that an address space is no longer one of its subordinate address spaces.

An address space remains connected to a particular transaction scheduler until the address space is terminated or issues an explicit Disconnect. Address space termination is an implicit Disconnect. Normally address space termination is all that is required to disconnect an address space from a transaction scheduler.

Environment

Authorization:	Supervisor state or PSW key 0-7

Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBDCON (ASCB_Ptr,
              Return_Code
             );
```

Figure 11. ATBDCON - Disconnect

Parameters

ASCB_ptr
Supplied parameter

- Type: Pointer
- Char Set: N/A
- Length: 32 bits

Specifies the pointer to the address space control block (ASCB) of the address space being disconnected from the transaction scheduler.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Disconnect might return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
Address space was successfully disconnected.

4
The value specified on the ASCB_ptr is not valid.

8
The address space specified was not a subordinate address space connected to the transaction scheduler.

32
The requested service is not supported in the caller's environment. For example, this return code will be given if the caller invokes any of the transaction scheduler services while holding a lock.

34
The requested transaction scheduler service must be invoked from a transaction scheduler address space.

44
APPC/MVS is not active.

48
APPC/MVS services failure.

Characteristics and Restrictions

Transaction schedulers that call the Disconnect service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.

Identify

The Identify service is used by a transaction scheduler to make itself known to APPC/MVS. A transaction scheduler issues Identify after it has initialized itself and is ready to receive or schedule requests from APPC/MVS. The transaction scheduler must supply an XCF member token on Identify to allow APPC/MVS to communicate with it. A transaction scheduler must identify itself to APPC/MVS before its subordinate address spaces can connect to APPC/MVS.

Specifically, this service is used by a transaction scheduler to do the following:

1. Identify itself to APPC/MVS.
2. Provide its XCF member token to APPC/MVS so that it can be notified of inbound allocate requests.
3. Optionally identify an information extract exit that may be invoked by APPC/MVS when it needs information from the transaction scheduler.
4. Determine whether the APPCPMxx parmlib member correctly defines the LUs for the transaction scheduler.
5. Specify initial status for LUs belonging to the transaction scheduler.
6. Identify an exit to convert a TP profile the first time it is referenced, and store the converted profile for future references.
7. Optionally provide a resource manager name, if the transaction scheduler is to process inbound, protected conversations (conversations with a synchronization level of syncpt), **and** is designed to use privately managed contexts to represent each of those inbound Allocate requests. For more information about schedulers that process protected conversations, see the additional considerations listed in Chapter 2, “General Transaction Scheduler Function: From Start-up to Termination,” on page 7.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	PASN = HASN = SASN or PASN ¬= HASN ¬= SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBIDN4 (Scheduler_name,
              Scheduler_extract_exit_addr,
              Scheduler_extract_user_field,
              Scheduler_member_token,
              TP_profile_processing,
              LU_initial_status,
              Scheduler_TP_profile_exit,
              Scheduler_TP_profile_exit_data,
              Resource_Manager_Name,
              Return_code
             );
```

Figure 12. ATBIDN4 - Identify

Parameters

Scheduler_Name
Supplied parameter

- Type: Character String
- Char Set: 01134
- Length: 8 bytes

Specifies the name of the transaction scheduler. This field must match a transaction scheduler name appearing in the LU definitions of an APPCPMxx parmlib member. The value must be the same as the value of the SCHED-keyword of one or more LUADD statements in APPCPMxx. The transaction scheduler name will also be used for operator displays. If the transaction scheduler runs only as a "single instance per system," this value should be a string that suggests the name of the component performing the Identify (for example, "ASCH" is an abbreviation used to identify the APPC transaction scheduler). If the transaction scheduler can run as "multiple copies per system," this value should be a string that identifies a particular copy of the transaction scheduler (for example, subsystems may wish to use the subsystem name that appears in the IEFSSNxx parmlib member). Once a transaction scheduler has successfully been identified, no other Identify call using the same Scheduler_Name will be accepted unless a corresponding Unidentify statement is issued.

Scheduler_Extract_Exit_Addr
Supplied parameter

- Type: Address
- Char set: N/A
- Length: 32 bits

Specifies the address of the transaction scheduler's information extract exit. This is an optional exit and may be left zero. If specified, this exit must reside in the common-area of storage. See "Extract Exit" on page 57 for information about coding a transaction scheduler extract exit.

Scheduler_Extract_User_Field
Supplied parameter

- Type: Character string
- Char set: No restriction
- Length: 8 bytes

Specifies a user-defined field or token passed to the transaction scheduler's information extract exit.

Scheduler_Member_Token
Supplied parameter

- Type: Character string
- Char set: No restriction
- Length: 8 bytes

Specifies an XCF member token. The member token represents a member of the XCF group that is joined when the Join_Sysappc_Group service is invoked. Messages are sent to this member to report when the transaction scheduler's LU name is activated or deactivated. Messages are also sent to report the arrival of inbound allocate requests. APPC/MVS does not check the validity of this member token. If a transaction scheduler passes an unknown member token, then the transaction scheduler will not receive notification of the arrival of inbound allocate requests.

TP_Profile_Processing
Supplied parameter

- Type: Integer
- Char set: N/A
- Length: 32 bits

Specifies the TP_Profile processing characteristics to use for this transaction scheduler.

Valid values for this parameter are:

Value
Meaning

0
Required

Specifies that APPC/MVS should reject any inbound allocate request that specifies a TP_Name for which a TP_Profile entry does not exist. If a TP_Profile entry does not exist, the inbound allocate request is rejected with TP_Not_Recognized (sense code X'10086021').

1
Optional

Specifies that a TP_Profile entry is not required. APPC/MVS will perform all validity and security checks and reject the request if any of these checks fail. If a TP_Profile entry does not exist, APPC/MVS will indicate this in the XCF message sent to the transaction scheduler to notify it of the inbound allocate request.

LU_Initial_Status
Supplied parameter

- Type: Integer
- Char set: N/A
- Length: 32 bits

Specifies the initial status of LUs controlled by this transaction scheduler. Any additional LUs being added for this transaction scheduler after Identify will initially be given this status, unless Control Halt_All or Resume_All is called to set the status.

Valid values for this parameter are:

Value
Meaning

0
Active

Specifies that APPC/MVS should activate the LU(s) controlled by this transaction scheduler. The status of every LU controlled by this transaction scheduler will initially be put into Active state.

1
Outbound_Only

Specifies that APPC/MVS should temporarily halt processing of allocate requests to the LU or LUs controlled by this transaction scheduler. The transaction scheduler has to call Control Resume for the LU to begin accepting inbound requests. The status of every LU controlled by this transaction scheduler, whether it is added to the system at initialization or by a subsequent SET command,

will initially be put into Outbound_Only state, unless Control Resume_All is called to set the status.

When the APPC address space terminates and restarts, the transaction schedulers that have called Identify and Connect before have to reidentify themselves and reconnect all their subordinate address spaces. A transaction scheduler can use this option to temporarily halt processing of inbound allocate requests to the LU while it is in the process of reconnecting its subordinate address spaces. It can issue a Control Resume request to activate all the LUs when the reconnect process is finished.

Scheduler_TP_profile_exit
Supplied parameter

- Type: Character string
- Char set: 01134
- Length: 8 bytes

Specifies the name of the exit that will receive control when the TP profile requires conversion. To specify no exit, set this parameter to 8 blanks. If you specify an exit, it must reside in LPA or in the LNKLST concatenation. (See the PROGxx or LNKLSTxx parmlib member description in *z/OS MVS Initialization and Tuning Reference* for more information about the LNKLST concatenation.) For more information about this exit, see "TP Profile Conversion Exit" on page 59.

Scheduler_TP_profile_exit_data
Supplied parameter

- Type: Character string
- Char set: No restriction
- Length: 8 bytes

Specifies data to be passed to the TP profile conversion exit each time it is invoked; for example, the address of a workarea for the exit to use. For more information about how the exit receives this input data, see "TP Profile Conversion Exit" on page 59.

Resource_Manager_Name
Supplied parameter

- Type: Character string
- Character Set: See the description of the Register_Resource_Manager callable service in *z/OS MVS Programming: Resource Recovery* for more information about the Resource_Manager_Name character set and naming restrictions.
- Length: 32 bytes

Specifies the unique name that identifies the transaction scheduler as a resource manager that is registered with the registration services. Resource_Manager_Name is an optional parameter and may be set to zeroes or blanks if either:

- The LUs for the transaction scheduler do not support protected conversations, or
- The scheduler is not designed to use a privately managed context to represent an inbound Allocate request.

If the transaction scheduler provides a Resource_Manager_Name, but a privately managed context could not be created to represent subsequent inbound Allocate requests, APPC/MVS rejects the inbound Allocate request, and the allocator of the conversation will receive a TP_Not_Available_Retry error return code on the next conversation call that allows a TP_Not_Available_Retry return code to be presented.

For more information about schedulers that process protected conversations, see the additional considerations listed in Chapter 2, "General Transaction Scheduler Function: From Start-up to Termination," on page 7.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Identify might return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
The Identify request was accepted. The LUs are activated asynchronously.

4
The Identify request was accepted. No base LU name is present. The APPCPMxx parmlib member or members specify at least one LU name that is controlled by the transaction scheduler, but no LU name is designated as the transaction scheduler's base LU. This situation might arise because the APPCPMxx parmlib member was incorrectly coded, or because the installation has deliberately chosen this configuration.

8
The Identify request was accepted. No LU names are applicable. APPC/MVS found that the APPCPMxx parmlib member specifies no LU names that are controlled by the transaction scheduler. This situation might arise because the APPCPMxx parmlib member did not specify the correct transaction scheduler name on the SCHED keyword of LUADD, or it might arise because APPC/MVS tried to initialize for the specified LUname and encountered a failure (for example, APPC/MVS was unable to open the required TP profile file).

12
The Identify request was rejected. The calling transaction scheduler address space is already identified using the same scheduler name as the Scheduler_name parameter passed in. This may occur if the caller issued Identify twice with the same scheduler name.

14
The Identify request was rejected. The calling transaction scheduler address space is already identified using a different scheduler name from the Scheduler_name parameter passed in. This may occur if the caller issued Identify twice with different scheduler names.

16
The Identify request was rejected. The Scheduler_Name parameter of Identify is already in use by some other address space that previously issued Identify.

18
The Identify request was rejected. The Scheduler_TP_profile_exit name that was passed could not be loaded.

20
The Identify request was rejected. The Scheduler_Name parameter value is not valid.

22
The Identify request was rejected. The Scheduler_TP_profile_exit name is not valid.

24
The Identify request was rejected. The TP_Profile_Processing parameter value is not valid.

26
The Identify request was rejected. The Resource_Manager_Name value does not represent a Resource Manager registered with RRS.

28
The Identify request was rejected. The LU_Initial_Status parameter value is not valid.

32
The requested service is not supported in the caller's environment. For example, this return code is given if the caller invokes any of the transaction scheduler services while holding a lock.

38
The requested transaction scheduler service cannot be invoked from a subordinate address space, or an address space that has outstanding APPC/MVS conversations.

40
The requested transaction scheduler service cannot be invoked from an APPC/MVS server address space.

44
APPC/MVS is not active.

48
APPC/MVS services failure.

Usage Notes

1. The transaction scheduler will be notified of an inbound allocate request only if the request passes all validity and security checks. The userid specified in the request must have RACF authority to access the TP profile entry (whether or not it exists), and if the TP profile entry is found, it must be marked "activated".
2. Timing restrictions on activities after Identify

   The transaction scheduler might create subordinate address spaces and call Connect before APPC/MVS reports that the base LU was successfully initialized. However, the transaction scheduler must not dispatch any work that might invoke an APPC/MVS Allocate service in these subordinate address spaces, before one of the following occurs:

   - The base LU is successfully initialized
   - ATBSASA is called to prevent allocated conversations being associated with the system default LU. For more information about this option, see "Set_AS_Attributes" on page 47.
3. Factors delaying asynchronous completion of Identify

   Some conditions might substantially delay the activation of an LU; for example, VTAM may be stopped when the Identify is accepted.

   An XCF message will be sent to the XCF-member representing the transaction scheduler when each of its LUs is activated.
4. Factors causing asynchronous failure of Identify

   Some conditions might cause an Identify to fail asynchronously after it has been accepted, for example, VTAM parameters might be mismatched (there might not be an APPL macro for the specified LUname), or APPC/MVS may be unable to open the specified TP profile file.

   An XCF message will be sent to the XCF member representing the transaction scheduler when the attempt to initialize an LU fails asynchronously.

   A transaction scheduler address space must issue Unidentify to undo its Identify, even if all of its LUs fail asynchronously.

   When LU initialization fails asynchronously, the system issues error messages indicating the cause of the failure (for example, unable to open the TP profile file). These messages will be issued to the same operator who receives messages about failures of LUs after initialization is completed.
5. Use of XCF by a transaction scheduler

   See "Join_Sysappc_Group" on page 44 for information regarding joining an XCF group.
6. Asynchronous initialization of the base LU name

   If Identify produces a return code of zero, then the transaction scheduler issuing Identify will receive an LU activation or LU deactivation message, with LU_Flags indicating that the message describes the base LU name. An LU deactivation message will indicate asynchronous failure of the attempt to initialize the LU name; an LU activation message will indicate successful initialization of the LUname.
7. Operation without a base LU name

If Identify produces a return code of 4, then the transaction scheduler will receive neither an LU activation nor an LU deactivation XCF message for the base LU name, unless the operator issues a SET command which establishes a base LU name for the transaction scheduler.

APPC/MVS does not issue any operator message indicating that the operator should do this; the transaction scheduler can to issue its own operator message asking the operator to perform such a SET command.

8. Operation with no LU names

   If Identify produces a return code of 8, then the transaction scheduler will receive neither an LU activation nor an LU deactivation message for the base LU name, unless the operator issues a SET command that establishes a base LU name for the transaction scheduler.

   In contrast to return code 4, APPC/MVS issues an operator message telling the operator to perform such a SET command.

9. Use of privately managed contexts for protected conversations

   A transaction scheduler should not change its resource manager name or remain as an unregistered resource manager while it is identified to APPC/MVS. Remaining unregistered, or changing the resource manager name without notifying APPC/MVS, results in the inability to create a privately managed context for inbound Allocate requests for LUs owned by the transaction scheduler. To avoid this inability:

   - Make sure the scheduler registers its resource manager name with registration services.
   - For a changed resource manager name, make sure the scheduler issues the Unidentify service, followed by the Identify service, to notify APPC/MVS of the name change.

Characteristics and Restrictions

1. Identify performs an automatic Connect of the home address space of the calling transaction scheduler. (See “Connect” on page 30.)
2. APPC/MVS supports one Identify per address space. Because of this, each transaction scheduler must be in its own address space.
3. The Identify service causes APPC/MVS to open one or more VTAM ACBs for the transaction scheduler's LUs. The ACBs are opened asynchronously if the Identify is accepted. Similarly, the TP profile file or files are also opened asynchronously. The asynchronous OPEN lets a transaction scheduler identify itself when VTAM is functioning. APPC/MVS informs a transaction scheduler that its LU is operational.
4. As soon as APPC/MVS accepts the Identify request, the scheduler's corresponding XCF message user routine and information extract exit may be invoked at any time.
5. Transaction schedulers that call the Identify service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on recovery and termination in *z/OS MVS Programming: Authorized Assembler Services Guide*.
6. An APPC/MVS server address space cannot use the Identify service. If an address space calls the Identify service while it is registered for an allocate queue, the system does not perform the Identify service function, and the caller receives a return code of 40 (decimal). For information about APPC/MVS servers, see *z/OS MVS Programming: Writing Servers for APPC/MVS*.

Join_Sysappc_Group

Use the Join_Sysappc_Group service to join the XCF group used by APPC/MVS. Each transaction scheduler must join the APPC XCF group. Other system applications can also join the APPC XCF group to be notified of APPC events.

APPC/MVS communicates with members of its XCF group by invoking their XCF message user routines. APPC/MVS notifies all group members of general interest events such as APPC initialization and

termination. APPC/MVS also notifies individual transaction schedulers when inbound allocate requests arrive for them. To notify individual schedulers, APPC/MVS uses a member_token that the transaction scheduler passes in on the Identify service. A transaction scheduler must call the Join_Sysappc_Group service, which provides the member token, before calling the Identify service. Unlike Identify and most other scheduler services, the Join_Sysappc_Group service can be called when APPC/MVS is not active.

If you do not use the Join_Sysappc_Group service to join the APPC XCF group, you must use APPC_GROUP_NAME as the group name with the IXCJOIN macro. A different group name is chosen on each system; therefore, each of these groups is “local to a system” and APPC/MVS can use the facilities of XCF regardless of whether XCF can perform cross-system communication. Also, the service performs IXCJOIN with the LASTING=NO option; thus, XCF “system-local mode” can be tolerated.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	PASN = HASN = SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBJGP1 (XCFMSGIN_exit_address,
              XCFMSGIN_memdata,
              Member_token,
              XCF_return_code,
              XCF_reason_code,
              Return_code
             );
```

Figure 13. ATBJGP1 - Join_Sysappc_Group

Parameters

XCFMSGIN_exit_address
Supplied parameter

- Type: Address
- Char Set: N/A
- Length: 32 bits

XCFMSGIN_exit_address specifies the address of the transaction scheduler's XCF message user routine. The routine takes control when a message becomes available for this member from another member of the group. For details about the requirements for and processing of the XCF message user routine, see “XCF Message User Routine” on page 53.

XCFMSGIN_memdata
Supplied parameter

- Type: Character
- Char Set: No restriction
- Length: 8 bytes

XCFMSGIN_memdata is an optional parameter that specifies an 8-byte member data field. This field is provided to the message user routine for this member. If you do not specify a value, XCF sets the member data field to binary zero. The transaction scheduler can use this field to pass the address and ASID or ALET of a particular control structure to the XCF message user routine.

Member_token
Returned parameter

- Type: Character
- Char Set: No restriction
- Length: 8 bytes

Member_token specifies the location where this service places the member token that represents the caller of this service.

XCF_return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

The return code passed back from the XCF IXCJOIN macro, if XCF rejects the Join request.

XCF_reason_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

The reason code passed back from the XCF IXCJOIN macro, if XCF rejects the Join request.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Join_Sysappc_Group may return one of the following decimal values in the return code parameter:

Decimal	Meaning
0	Request successful.
8	Request unsuccessful - XCF failed or request denied by XCF.
40	The caller was not running in supervisor state or PSW key 0-7.
48	APPC/MVS services failure.

Characteristics and Restrictions

1. This service will execute successfully even if XCF is operating in XCF local mode.
2. The caller must issue the IXCLEAVE macro to undo the effects of Join_Sysappc_Group. IXCLEAVE processing is performed automatically if the caller's address space or task terminates.
3. The message buffer that is provided in the message user routine must be accessible using the same protect key that is in effect at invocation of Join_Sysappc_Group.

4. The task that calls this service might end abnormally if a privileged program issues the XCF IXCTERM macro against this member. In that case, the task terminates with system completion code 00C, reason code 4, and the task's recovery routine cannot retry. Transaction schedulers can handle this by attaching a subtask that invokes Join_Sysappc_Group, and reattaching the subtask if it terminates with completion code 00C, reason code 4.
5. A transaction scheduler may join XCF groups other than the APPC group joined by this service.
6. The name of APPC's XCF group might vary from system to system and might change during re-IPL. If you need to know the XCF group name used by APPC (to dedicate specific resources to it, for example), you can use the ATBAPPCA mapping macro. The ATBAPPCA mapping macro is described in *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).
7. Transaction schedulers that call the Join_Sysappc_Group service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.

Set_AS_Attributes

The Set_AS_Attributes service lets a transaction scheduler set attributes for a subordinate address space. In particular, this service can set an attribute to designate whether conversations allocated from the specified address space are to be associated to the **system base LU**. You can use this service to prevent an inadvertent association in cases where the subordinate address space allocates a conversation before the transaction scheduler connects the subordinate address space to itself. Without Set_AS_Attributes in such cases, the conversation is assigned to the system base LU, and when the transaction scheduler attempts to connect the subordinate address space to itself later, if the conversation is still outstanding, the connect fails.

For example, you could use Set_AS_Attributes for this purpose in the following manner:

1. The transaction scheduler is active when APPC is not, so Identify and Connect cannot be performed.
2. The transaction scheduler calls Set_AS_Attributes with the Default_LU_Designation parameter set to 1 (to *not* associate conversations with the system base LU).
3. APPC is started on the system.
4. A subordinate address space allocates a conversation before the transaction scheduler identifies itself to APPC and connects the subordinate address space.

In the above scenario, if the transaction scheduler did not call Set_AS_Attributes first, APPC/MVS would assign the subordinate address space to the system base LU as soon as APPC/MVS received the outbound allocate request.

Instead, because the transaction scheduler does call Set_AS_Attributes first, APPC rejects the allocate request. When the transaction scheduler is notified of APPC initialization, the transaction scheduler can identify itself to APPC and connect the subordinate address space to itself.

The Set_AS_Attributes service can be called when APPC/MVS is not active.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	PASN = HASN = SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts

Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBSASA (ASCB_ptr,
              Default_LU_Designation,
              Return_Code
             );
```

Figure 14. ATBSASA - Set_AS_Attributes

Parameters

ASCB_ptr
Supplied parameter

- Type: Pointer
- Char Set: N/A
- Length: 32 bits

Specifies the pointer to the address space control block (ASCB) that represents the subordinate address space whose attributes are to be set.

Default_LU_Designation
Supplied parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Specifies whether conversations allocated from the specified address space should be associated with the system base LU. If conversations are not to be associated with the system base LU, the address space cannot use APPC services until it is explicitly connected using the ATBCONN service.

Valid values for this parameter are:

Value
Meaning

0
Associate outbound conversations with the system base LU. This value is the default.

1
Do **not** associate conversations with the system base LU.

By default, the conversations from any unconnected address space will automatically be associated to the system base LU unless this service is called.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Set_AS_Attributes might return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
Attributes were set successfully.

4
Request failed -- the ASCB_ptr was not valid.

8
Request failed -- the value for Default_LU_Designation was not valid (must be 0 or 1).

40
The caller was not running in supervisor state or with PSW key 0-7.

48
APPC/MVS services failure.

Characteristics and Restrictions

Transaction schedulers that call the Set_AS_Attributes service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.

Unidentify

Unidentify can be used by a transaction scheduler to reverse the effects of the Identify service. Unidentify terminates all APPC services for the transaction scheduler and its subordinate address spaces. Unidentify causes APPC/MVS to shut down the LU or LUs assigned to the transaction scheduler that issued Unidentify. The caller does not have to wait for this to occur. Once the Unidentify request is accepted, APPC/MVS returns control to the caller and assumes responsibility for taking down the LU or LUs. After APPC/MVS returns control, a transaction scheduler may invoke the IXCLEAVE macro to undo the effects of its invocation of Join_Sysappc_Group.

Unidentify automatically disconnects address spaces currently connected to the issuing transaction scheduler. New conversations (that is, inbound or outbound Allocate requests) for the scheduler are rejected. The outcome of existing conversations for the scheduler depends on the type of Unidentify call. (Existing conversations are those for which one LU has successfully sent and its partner LU has successfully received the Allocate request.)

After an Unidentify, the LU is placed in pending state to await another Identify request. A transaction scheduler must issue Identify if it is to restart.

Calls to Unidentify must be issued from the address space that issued the Identify.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	PASN = HASN = SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBUID1 (Unidentify_type,
              Return_code
             );
```

Figure 15. ATBUID1 - Unidentify

Parameters

Unidentify_type
Supplied parameter

- Type: Integer
- Char Set: N/A
- Length: 4 bytes

Specifies the type of Unidentify requested, which affects each LU that is identified with this transaction scheduler. Valid values for this parameter are:

Value	Meaning
0	Unidentify_Normal Calls to ATBUID1 with an Unidentify_type of Normal (or calls to ATBUNID) cause the ACB for each LU to be closed only after all existing conversations are deallocated. Because work for the LU is quiesced, a normal Unidentify is similar to LUDEL processing.
1	Unidentify_Immediate Calls to ATBUID1 with an Unidentify_type of Immediate cause the ACB of each LU to be closed immediately. All existing conversations fail when a TP issues the next APPC/MVS or CPI-C service call. Unidentify_Immediate can be used in situations requiring fast termination, such as takeover by a backup scheduler.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Unidentify might return one of the following decimal values in the return code parameter:

Decimal	Meaning
0	Unidentify was accepted.
16	The Identify_type value passed on ATBUID1 was not valid.
32	The requested service is not supported in the caller's environment. For example, this return code will be given if the caller invokes any of the transaction scheduler services while holding a lock.
34	The requested transaction scheduler service must be invoked from a transaction scheduler address space.

44
APPC/MVS is not active.

48
APPC/MVS services failure.

Characteristics and Restrictions

Transaction schedulers that call the Unidentify service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.

Unidentify

Chapter 5. Transaction Scheduler User Exits

Transaction schedulers need to provide the following user exit routines to perform certain functions.

- XCF message user routine

  Each transaction scheduler must provide an XCF message user routine to obtain information about general APPC/MVS events and to receive inbound allocate requests that are directed to the scheduler.
- Extract exit

  Transaction schedulers must supply an extract exit routine if they:

  – Call the ATBEXAI service to extract information about their scheduling

  – Run more than one transaction program in their address space simultaneously, and one of the TPs issues an Allocate, Get_TP_Properties, or Get_Conversation request.
- TP profile conversion exit

  Schedulers can provide an exit routine to convert a TP profile on its first reference and store the converted form of the profile for future reference, thus avoiding repeated conversion and potentially improving performance.
- TP profile syntax exit and message routine

  Schedulers can provide an exit routine to check the syntax of scheduler-specific information before it is added to the TP profile. An associated message routine can issue messages about syntax errors to the SYSPRINT data set.

XCF Message User Routine

APPC/MVS invokes a transaction scheduler's XCF message user routine to inform the transaction scheduler of general events affecting APPC/MVS, and to pass to it all inbound Allocate requests that are addressed to the transaction scheduler's LU. Depending on the message that APPC/MVS passes, the message user routine might have to issue the XCF IXCMSGI macro to obtain additional information.

The transaction scheduler identifies its XCF message user routine to APPC/MVS on the Join_Sysappc_Group service.

References

- See *z/OS MVS Programming: Sysplex Services Guide* for more information about designing an XCF message user routine.
- See *z/OS MVS Programming: Sysplex Services Reference* for the coding details for the IXCMSGI macro.
- See IXCYMEPL in *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) for complete field names and lengths, offsets, and descriptions of the fields in the message user routine parameter list, which is mapped by the IXCYMEPL mapping macro.
- See ATBXCFMS in *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) for complete field names and lengths, offsets, and descriptions of the fields in XCF messages sent by APPC/MVS, which are mapped by the ATBXCFMS mapping macro.

Environment

The XCF message user routine receives control in the following environment:

Authorization:	Supervisor state and PSW key 0

Dispatchable unit mode:	SRB mode
Cross memory mode:	PASN = HASN = SASN. The primary address space equals the primary address space of the transaction scheduler, and can be swappable or nonswappable.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	The user routine does not hold any locks on entry.

Processing

All XCF messages sent by APPC/MVS contain a type code to indicate the type of message being sent. The type code is the first four bytes of the 32-byte message control area passed in the parameter list (mapped by IXCYMEPL) to the message user routine. The user routine must examine the type code to determine whether it is a general event message or an Allocate request message. The user routine should be optimized to handle requests to allocate TPs, because these will be the most common.

Note: Messages requesting the transaction scheduler to allocate a TP will not be sent until APPC/MVS activates at least one of the transaction scheduler's LUs. However, because XCF messages might be delivered out of sequence, the XCF message user routine might receive an Allocate TP request message before it receives the message reporting that the LU is active. Also, the message user routine is not single-threaded; several processors may execute the user routine simultaneously, with each processor handling a different message.

z/OS MVS Programming: Sysplex Services Guide contains general information about designing and coding an XCF message user routine; you should be familiar with that information before coding the message user routine for a transaction scheduler. The rest of this section contains guidance that applies only to designing a message user routine for use with a transaction scheduler for APPC/MVS.

Message Types

The contents of the 32-byte message control area (MEPLCNTL field) indicate that the XCF message is one of the following types:

- APPC Initialization or Termination
- LU Activation or Deactivation
- Allocate TP request.

The ATBXCFMS mapping macro maps these APPC/MVS messages. The general event message for APPC initialization/termination is small enough to be contained in the 32-byte message control area. However, the LU activation/deactivation and the Allocate TP request messages are each too large to fit in the 32-byte message control area. Also, for the LU activation/deactivation message, additional information is available if optional data was supplied for the transaction scheduler's LU in the USERVAR, ALTLU, and GRNAME keywords on the LUADD statement in the APPCPMxx parmlib member. In these cases, the message user routine must issue the XCF IXCMSGI macro to receive the rest of the message or the additional information.

When you design the message user routine to issue the IXCMSGI macro, provide the message token value in the MEPLMTOK field of the parameter list. Also provide a buffer to contain the data returned by IXCMSGI; the storage key for the buffer must match the PSW key of the caller of Join_Sysappc_Group.

You may receive data in a single or in multiple buffers. See *z/OS MVS Programming: Sysplex Services Guide* for information about designing a message user routine to use single or multiple buffers without encountering errors.

Programming Notes for LU Activation/Deactivation Messages

For LU activation/deactivation messages:

- In the XCF 32-byte control area, a flag indicates whether the LU is capable of handling outbound Allocate requests that use a network-qualified name to identify the partner LU.
- In the IXGMSGI buffer:
  - If the LU is a member of a VTAM generic resource group, a field contains the generic resource name associated with the LU.
  - Optional USERVAR data might indicate that a transaction scheduler has an alias name defined for its local LU. Depending on the release of VTAM your installation is using, one of the following results:
    - With VTAM 4.3 or earlier, conversation allocations from that LU will fail if the partner LU name on the Allocate request is the USERVAR alias of the local LU name.
    - With VTAM 4.4 or later, conversation allocations from that LU will be accepted if the partner LU name on the Allocate request is the USERVAR alias of the local LU name.

Programming Notes for Allocate TP Request Messages

If the message user routine issues IXCMSGI for more data, but the IXCMSGI macro fails, the message user routine should call the Cleanup_TP service to clean up any outstanding APPC/MVS resources, supplying the TP_ID that was passed to the message user routine in the 32-byte control area.

If the IXCMSGI macro successfully returns to the message user routine, the buffer contains the Allocate TP request message, which includes such data as:

TP_ID
A token that uniquely identifies a transaction to MVS. Transaction schedulers use it to inform APPC/MVS:

- Where the transaction executes (through the Associate service)
- When the transaction terminates normally or abnormally (through the Cleanup_TP service).

PROFILE
The TP profile entry contents, if a profile was available. The profile can be mapped by the ATBDFTP mapping macro, which is in *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary). The format and content of the TP_Profile is transaction-scheduler dependent.

CONV_CORR
The conversation correlator associated with this TP. The conversation correlator, which is specified in the FMH-5 that contains the input for the allocate request, associates that request with a response from the transaction scheduler. See *z/OS MVS Programming: Writing Transaction Programs for APPC/MVS* for more information about how partner TPs and transaction schedulers can use a conversation correlator.

CONV_SYNC_LEVEL
The synchronization level of the conversation associated with this TP. The synchronization level is one of the following:

Value	Meaning
0	None
1	Confirm
2	Syncpt

LUWID
The logical unit of work ID is used to identify the most recent sync point, or for accounting purposes.

CONTEXT_TOKEN
A token that identifies the context representing a transaction program's unit of work. This field is meaningful only for protected conversations.

SECTOKN
A token that identifies the security environment created for the user by RACF. If your installation uses RACF, the alternate transaction scheduler can use this token to create a security environment in the program's execution address space. When recalling the RACF ACEE associated with the security token, for performance reasons, code STAT=NO on the RACROUTE macro.

ENVR
A RACF object that the transaction scheduler can use quickly recreate a security environment in the program's execution address space.

For more information about the RACF Security_Token and ENVR_Object, see *z/OS Security Server RACROUTE Macro Reference*.

Programming Considerations

- The message user routine can reside either in the private storage of the address space from which the Join_Sysappc_Group service is invoked, or in common storage.
- The message user routine should return to its caller as soon as possible, because system resources are held until the message user routine gives up control.
- To avoid performance degradation in the XCF signalling service, and in the system as a whole, do not issue the SUSPEND macro within the message user routine.

Entry Specifications

XCF passes information to the message user routine in registers and in a parameter list.

Registers at Entry

On entry to the message user routine, the registers contain the following information:

Register	Contents
GPR 0	Does not contain any information for use by the message user routine.
GPR 1	Address of the message user routine parameter list.
GPRs 2 - 12	Do not contain any information for use by the message user routine.
GPR 13	Address of a standard save area. (The message user routine does not have to save and restore XCF's registers in this save area. The message user routine can use this save area to save its own registers when it uses services that might overwrite the contents of registers.)
GPR 14	Return address
GPR 15	Entry point address of message user routine.
ARs 0 - 15	Do not contain any information for use by the message user routine.

Parameter List Contents

The parameter list that XCF passes to the message user routine is mapped by the IXCYMEPL mapping macro and is pointed to by GPR 1. The parameter list is addressable from the primary address space in which the message user routine runs, and includes the following information:

- A message token (MEPLMTOK) for the message user routine to use when issuing the IXCMSGI macro.
- The member data value (MEPLMDAT) provided on return from the Join_Sysappc_Group service (XCFMSGIN_memdata parameter).
- The length of the message (MEPLMLEN).
- The message control area (MEPLCNTL), which contains the message information from APPC/MVS. The message user routine must look at this 32-byte area to determine the type and contents of the APPC/MVS message.

Figure 16 on page 57 illustrates how APPC/MVS messages are mapped.

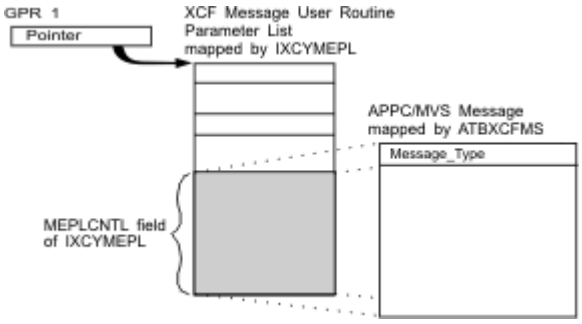


Figure 16. How APPC/MVS Messages are Mapped

Return Specifications

On return, the message user routine does not have to set any return codes or place any information in the GPRs. The message user routine must return control through a BR 14 or a BSM 0,14.

Extract Exit

The transaction scheduler extract exit is an optional exit invoked by APPC/MVS to perform one of the following two functions:

- Provide information requested by a call to the Extract_Information service (ATBEXAI). When a transaction program calls ATBEXAI for information about how it was scheduled, the appropriate transaction scheduler extract exit is driven. The output from this exit is defined by the transaction scheduler. If the exit is not supplied by the transaction scheduler, the transaction program receives a return code indicating that no information was returned.
- The extract exit is also invoked when APPC/MVS needs to determine which transaction program is requesting APPC services. The extract exit is invoked for this reason only when the request is coming from an address space that has more than one TP_ID associated with it, (namely, a transaction scheduler address space). The extract exit is driven to allow the transaction scheduler to specify a TP_ID. It is used when a transaction program in the transaction scheduler address space, or when the transaction scheduler issues one of the following service calls:
  - Allocate (unless a TP_ID is specified)
  - Get_TP_Properties
  - Get_Conversation

  The exit is invoked only when there are two or more TPs associated with the address space.

  The extract exit for a transaction scheduler is established when the transaction scheduler invokes the Identify service. If the transaction scheduler does not supply this exit, requests from the transaction scheduler address space for the above service calls are rejected when more than one TP_ID is associated with the address space. If any of the above service calls are issued from the scheduler address space in SRB mode, the exit will need a mechanism to determine the TP_ID when a TCB is not available.

Environment

The transaction scheduler extract exit is given control on the same dispatchable unit that invoked the particular service: Allocate, Get_TP_Properties, Get_Conversation, or Extract_Information. Note that these services support SRB mode callers. Therefore, if the service is invoked from the transaction scheduler address space in SRB mode, the exit is driven in SRB mode as well, and is restricted in the services that it can issue. For example, the exit cannot issue SVCs, nor issue a WAIT or SUSPEND macro, because the exit might be invoked on the synchronous path of an asynchronous service.

The exit receives control in the following environment:

Authorization:	Supervisor state and PSW key 1

Dispatchable unit mode:	Task or SRB
Cross memory mode:	PASN = APPC, HASN = caller's HASN, SASN = caller's PASN
AMODE:	31-bit
ASC mode:	Primary
Storage key:	1
Interrupt status:	Enabled
Locks:	The exit does not hold any locks on entry.

Exit Recovery

The caller of the exit routine should establish its own recovery environment before calling the exit routine. The exit routine should also establish its own recovery environment and, within its recovery, request a tailored dump. Before each exit routine returns control to its caller, the exit routine must delete the recovery environment it established and free the storage that it obtained. If the exit routine does not establish its own recovery environment, the caller does the following when the exit routine ends abnormally:

- Writes a logrec data set error record, and
- Writes a dump.

The dump and the logrec data set error record might not contain enough information to diagnose the error.

For more information on providing recovery, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Programming Requirements

- This exit must reside in common storage.
- To preserve the registers of the caller, the exit routine must follow the linkage conventions described in "Linkage Conventions" in *z/OS MVS Programming: Assembler Services Guide*.

Entry Specifications

APPC/MVS passes information to the extract exit in registers and in the scheduler extract control block.

Registers on Entry

On entry to the extract exit, the registers contain the following information:

Register	Contents
GPR 0	Does not contain any information for use by the exit.
GPR 1	The address of a one-word parameter list that contains the address of the scheduler extract control block.
GPRs 2-15	Do not contain any information for use by the exit.

Scheduler Extract Control Block

The scheduler extract control block is in key 1 storage, so its contents are immediately accessible by the exit. The scheduler exit control block contains a service indicator that the exit can use to determine what processing needs to be done:

Indicator
Meaning

Get_Info
The extract exit was invoked to supply information requested by a caller of the Extract_Information service.

Get_TP_ID
The extract exit was invoked because APPC/MVS could not determine which TP_ID to use for a service call.

The scheduler extract exit control block is mapped by the ATBSECB mapping macro; for detailed information about all of the fields in ATBSECB, see *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Return Specifications

On return, the extract exit does not have to place any information in the GPRs. However, depending on the service indicator, the extract exit must place values in certain fields before returning to its caller:

- For a Get_Info call, the scheduler exit control block contains the address of a temporary buffer in which the extract exit should return whatever data is required by its published interface.

  The extract exit also must set the Return_Code field to one of the values that APPC/MVS returns for the Extract_Information service.
- For a Get_TP_ID call, the scheduler exit control block contains a TP_ID field in which the extract exit should return the appropriate transaction program ID; that is, the ID passed to the transaction scheduler on either the inbound Allocate TP request message or the Define_Local_TP service.

  The extract exit also must set the Return_Code field to zero (to indicate successful processing) or any non-zero value (to indicate a failure).

TP Profile Conversion Exit

The transaction scheduler TP profile conversion exit is established when the transaction scheduler invokes the Identify service, specifying the exit name. The TP profile conversion exit allows a transaction scheduler to convert the contents of a TP Profile when the first inbound allocate request arrives for the TP. The exit then returns the converted form of the TP profile, which APPC/MVS saves and uses on subsequent inbound requests.

Environment

This exit is invoked on each inbound request for a TP profile that has not been previously converted and saved. The exit receives control in the following environment:

Authorization:	Supervisor state and PSW key 1
Dispatchable unit mode:	Task mode
Cross memory mode:	PASN = HASN = APPC
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Storage key:	1
TCB protect key:	1
Interrupt status:	Enabled
Locks:	The exit does not hold any locks on entry.
Subpool limitations:	1

The TP profile conversion exit receives control after the unconverted TP profile is retrieved on an inbound request. The exit cannot invoke wait routines.

Exit Recovery

The caller of the exit routine should establish its own recovery environment before calling the exit routine. The exit routine should also establish its own recovery environment and, within its recovery, request a

tailored dump. Before each exit routine returns control to its caller, the exit routine must delete the recovery environment it established and free all storage it obtained. If the exit routine does not establish its own recovery environment, the caller does the following when the exit routine ends abnormally:

- Writes a logrec data set error record, and
- Writes a dump.

The dump and the logrec data set error record might not contain enough information to diagnose the error.

For more information on providing recovery, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Programming Requirements

To preserve the registers of the caller, the exit routine must follow the linkage conventions described in "Linkage Conventions" in *z/OS MVS Programming: Assembler Services Guide*.

Installation

The TP profile conversion exit must reside in LPA or in the LNKLST concatenation. It must be a reentrant and reusable module. See the PROGxx or LNKLSTxx parmlib member description in *z/OS MVS Initialization and Tuning Reference* for more information about the LNKLST concatenation.

Entry Specifications

APPC/MVS passes information to the TP profile conversion exit in registers and in a parameter list.

Registers on Entry

On entry to the extract exit, the registers contain the following information:

Register	Contents
GPR 0	Does not contain any information for use by the exit.
GPR 1	The address of the parameter list described in "Parameter List Contents" on page 60.
GPRs 2-12	Do not contain any information for use by the exit.
GPR 13	The address of a standard 18-word save area.
GPR 14	The return address.
GPR 15	The entry point address.

Parameter List Contents

The parameter list is pointed to by GPR 1, and is mapped by the ATBDFTPE mapping macro. It contains such information as:

Parameters
Any data specified on the Scheduler_TP_profile_exit_data parameter of the Identify call.

TP_profile_key_pointer
The address of the TP profile key. Mapped by ATBDFTP mapping macro.

TP_profile_pointer
The address of the unconverted TP profile as retrieved from DASD. Mapped by ATBDFTP mapping macro.

Conv_data_pointer
The address that is to contain the converted TP profile. APPC/MVS obtains and frees this storage.

Conv_data_length
The length that is available for saving a converted TP profile. The converted TP profile must not exceed this length.

Return Specifications

Before returning control to its caller, the exit must place the length of the converted TP profile in the Conv_data_length field, if the conversion was successful. Also, the exit must ensure that the register contents are as follows:

Register	Contents
GPRs 0-14	The exit must restore the contents to what they were when the exit received control.
GPR 15	One of the following return code values: **Value** **Meaning** **0 (0)** Conversion was successful. APPC/MVS saves a copy of the converted TP profile. **4 (4)** Conversion was unsuccessful; the conversion exit did not convert the profile. APPC/MVS saves a copy of the unconverted TP profile. **12 (C)** Conversion was unsuccessful; the conversion exit encountered a syntax error. APPC/MVS does not save a copy of the TP profile.

TP Profile Syntax Exit

This exit is provided to enable transaction schedulers to check the syntax of scheduling information before it is added to the TP profile data set. The syntax exit, which must be provided along with the scheduler and specified in the TP profile's TPSCHED_EXIT keyword, is invoked on TPADD or TPMODIFY commands by the APPC/MVS administration utility or administration dialog. If no syntax exit is specified, the transaction scheduler information in the TP profile is assumed to apply to the APPC/MVS transaction scheduler, and the APPC/MVS administration utility and JCL converter/interpreter check it for the syntax expected by that scheduler.

If a syntax exit is specified and it finds errors in the transaction scheduler information, the exit can invoke an IBM-supplied message routine (see “Profile Syntax Message Routine” on page 63) to write messages to the SYSPRINT data set and can prevent the profile from being added or modified. The administrator who is creating or modifying the TP profile can then correct the error and try again.

Environment

The syntax exit receives control in the following environment:

Authorization:	Supervisor state and PSW key 1
Dispatchable unit mode:	Task mode
Cross memory mode:	PASN = HASN = SASN
AMODE:	31-bit
RMODE:	ANY
ASC mode:	Primary
Storage key:	1
TCB protect key:	1
Interrupt status:	Enabled
Locks:	The exit does not hold locks on entry.

Exit Recovery

The caller of the syntax exit should establish its own recovery environment before calling the exit. The syntax exit should also establish its own recovery environment and, within its recovery, request a tailored dump. Before the exit returns control to its caller, the it must delete the recovery environment it established and free all storage it obtained. If the syntax exit does not establish its own recovery environment, the caller does the following when the exit ends abnormally:

- Writes a logrec data set error record, and
- Writes a dump.

The dump and the logrec data set error record might not contain enough information to diagnose the error.

For more information on providing recovery, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Programming Requirements

To preserve the registers of the caller, the syntax exit must follow the linkage conventions described in "Linkage Conventions" in *z/OS MVS Programming: Assembler Services Guide*.

Installation

The syntax exit must meet all the following conditions:

- Reside in LPA or in the LNKLST concatenation (for example, SYS1.LINKLIB)
- Be in an APF-authorized STEPLIB (see note below)
- Be link-edited with attributes reusable and reentrant.

Note: If the exit resides in the LPALST concatenation or in the LNKLST concatenation, the system automatically considers the exit to be authorized.

See *z/OS MVS Initialization and Tuning Reference* for more information about the LNKLST concatenation and APF-authorized libraries.

Entry Specifications

APPC/MVS passes information to the syntax exit in registers and in the scheduler extract control block.

Registers on Entry

On entry to the syntax exit, the registers contain the following information:

Register	Contents
GPR 0	Does not contain any information for use by the exit.
GPR 1	The address of the parameter list described in "Parameter List Contents" on page 62.
GPRs 2-12	Do not contain any information for use by the exit.
GPR 13	The address of a standard 72-byte save area.
GPR 14	The return address.
GPR 15	The entry point address.

Parameter List Contents

Figure 17 on page 63 illustrates the format and content of the parameter list for the TP profile syntax exit.

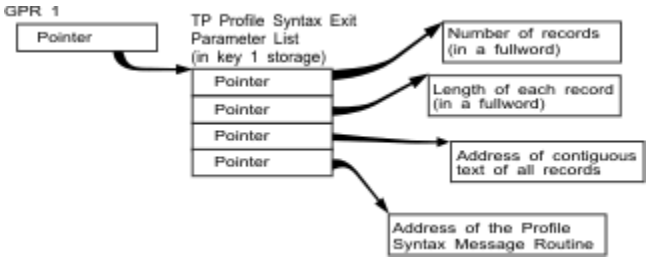


Figure 17. Parameter List of the TP Profile Syntax Exit

Return Specifications

Before returning control to its caller, the syntax exit must ensure that the register contents are as follows:

Register	Contents
GPRs 0-14	The exit must restore the contents to what they were when the exit received control.
GPR 15	One of the following **decimal** return code values: **Value** **Meaning** **0** No syntax errors encountered; okay to add or modify profile. **4** Syntax errors encountered; fix before adding or modifying profile. **8** Processing error; could not check profile. **12** System error encountered.

Profile Syntax Message Routine

This message routine is provided by IBM to enable TP profile syntax exits to write messages to the SYSPRINT data set about any errors that they find in the scheduling information being specified on a TPADD or TPMODIFY command through the APPC/MVS administration utility or dialog.

Environment

The profile syntax message routine receives control in the following environment:

Authorization:	Supervisor state and PSW key 1
Dispatchable unit mode:	Task mode
Cross memory mode:	PASN = HASN = SASN
AMODE:	31-bit
RMODE:	ANY
ASC mode:	Primary
Storage key:	1
TCB protect key:	1
Interrupt status:	Enabled
Locks:	The routine does not hold any locks on entry.

The TP profile syntax message routine runs under the APPC administration utility's task in the APPC administration utility's address space.

Entry Specifications

The profile syntax message routine receives information through registers and a parameter list.

Registers on Entry

On entry to the syntax message routine, the registers contain the following information:

Register	Contents
GPR 0	Does not contain any information for use by the exit.
GPR 1	The address of the parameter list described in “Parameter List Contents” on page 64.
GPRs 2-12	Do not contain any information for use by the exit.
GPR 13	The address of a standard 72-byte save area.
GPR 14	The return address.
GPR 15	The entry-point address.

Parameter List Contents

Figure 18 on page 64 shows the input parameters to the TP profile syntax message routine.

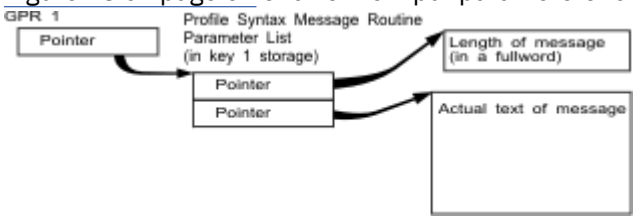


Figure 18. Input to the TP Profile Syntax Message Routine

Return Specifications

Before returning control to its caller, the profile syntax message routine sets the register contents as follows:

Register	Contents
GPRs 0-14	The routine restores the contents to what they were when the routine received control.
GPR 15	One of the following **decimal** return code values: **Value** **Meaning** **0** No syntax errors encountered. **4** Errors encountered; no messages were sent to SYSPRINT.

Appendix A. Character sets

APPC/MVS makes use of character strings composed of characters from one of the following character sets:

- Character set 01134, which is composed of the uppercase letters A through Z and numerals 0-9.
- Character set Type A, which is composed of the uppercase letters A through Z, numerals 0-9, national characters (@, $, #), and must begin with either an alphabetic or a national character.
- Character set 00640, which is composed of the uppercase and lowercase letters A through Z, numerals 0-9, and 19 special characters. Note that APPC/MVS does not allow blanks in 00640 character strings.

These character sets, along with hexadecimal and graphic representations, are provided in the following table:

Table 3. Character Sets 01134, Type A, and 00640

Hex Code	Graphic	Description	Character Set		
			01134	Type A	00640
40		Blank			
4B	.	Period			X
4C	<	Less than sign			X
4D	(	Left parenthesis			X
4E	+	Plus sign			X
50	&	Ampersand			X
5B	$	Dollar sign		X (Note 1)	
5C	*	Asterisk			X (Note 2)
5D	)	Right parenthesis			X
5E	;	Semicolon			X
60	–	Dash			X
61	/	Slash			X
6B	,	Comma			X (Note 3)
6C	%	Percent sign			X
6D	_	Underscore			X
6E	>	Greater than sign			X
6F	?	Question mark			X
7A	:	Colon			X
7B	#	Pound sign		X (Note 1)	
7C	@	At sign		X (Note 1)	
7D	'	Single quote			X
7E	=	Equals sign			X
7F	"	Double quote			X
81	a	Lowercase a			X

Table 3. Character Sets 01134, Type A, and 00640 (continued)

Hex Code	Graphic	Description	Character Set		
			01134	Type A	00640
82	b	Lowercase b			X
83	c	Lowercase c			X
84	d	Lowercase d			X
85	e	Lowercase e			X
86	f	Lowercase f			X
87	g	Lowercase g			X
88	h	Lowercase h			X
89	i	Lowercase i			X
91	j	Lowercase j			X
92	k	Lowercase k			X
93	l	Lowercase l			X
94	m	Lowercase m			X
95	n	Lowercase n			X
96	o	Lowercase o			X
97	p	Lowercase p			X
98	q	Lowercase q			X
99	r	Lowercase r			X
A2	s	Lowercase s			X
A3	t	Lowercase t			X
A4	u	Lowercase u			X
A5	v	Lowercase v			X
A6	w	Lowercase w			X
A7	x	Lowercase x			X
A8	y	Lowercase y			X
A9	z	Lowercase z			X
C1	A	Uppercase A	X	X	X
C2	B	Uppercase B	X	X	X
C3	C	Uppercase C	X	X	X
C4	D	Uppercase D	X	X	X
C5	E	Uppercase E	X	X	X
C6	F	Uppercase F	X	X	X
C7	G	Uppercase G	X	X	X
C8	H	Uppercase H	X	X	X
C9	I	Uppercase I	X	X	X
D1	J	Uppercase J	X	X	X

Table 3. Character Sets 01134, Type A, and 00640 (continued)

Hex Code	Graphic	Description	Character Set		
			01134	Type A	00640
D2	K	Uppercase K	X	X	X
D3	L	Uppercase L	X	X	X
D4	M	Uppercase M	X	X	X
D5	N	Uppercase N	X	X	X
D6	O	Uppercase O	X	X	X
D7	P	Uppercase P	X	X	X
D8	Q	Uppercase Q	X	X	X
D9	R	Uppercase R	X	X	X
E2	S	Uppercase S	X	X	X
E3	T	Uppercase T	X	X	X
E4	U	Uppercase U	X	X	X
E5	V	Uppercase V	X	X	X
E6	W	Uppercase W	X	X	X
E7	X	Uppercase X	X	X	X
E8	Y	Uppercase Y	X	X	X
E9	Z	Uppercase Z	X	X	X
F0	0	Zero	X	X	X
F1	1	One	X	X	X
F2	2	Two	X	X	X
F3	3	Three	X	X	X
F4	4	Four	X	X	X
F5	5	Five	X	X	X
F6	6	Six	X	X	X
F7	7	Seven	X	X	X
F8	8	Eight	X	X	X
F9	9	Nine	X	X	X

Note:

1. Avoid these characters because they display differently depending on the national language code page in use.
2. Avoid using the asterisk in TP names because it causes a subset list request when entered on panels of the APPC administration dialog and in DISPLAY APPC commands.
3. Avoid using the comma in TP names because it acts as a parameter delimiter when entered in DISPLAY APPC commands.

Appendix B. Previous Versions of APPC/MVS System Services

This section describes previous APPC/MVS system service calls that have been replaced by newer versions. The newer versions are documented in Chapter 4, “APPC/MVS System Services Summary,” on page 17. These previous versions remain valid in later releases but contain no enhancements.

Callers of these system services must be in supervisor state or PSW key 0-7. Callers that are not in supervisor state or PSW key 0-7 end with system completion (abend) code 0C2, with the exception of ATBMIGRP, which provides a return code.

ATBCMAS— Cleanup_Address_Space

Note: The ATBCAS1 call is the preferred programming interface for this service.

Cleanup_Address_Space can be used to request APPC/MVS to clean up all APPC/MVS resources for an address space. APPC/MVS cleans up all conversation resources for all transaction programs that are associated with the address space at the time the Cleanup_Address_Space was issued.

The Cleanup_Address_Space service can be invoked by a transaction scheduler subordinate address space for a transaction program or job that terminates normally or abnormally.

APPC/MVS deletes the TP_ID or TP_IDs from the system as a result of this call; this cleanup process might occur asynchronously.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task or SRB mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBCMAS (ASCB_ptr,
              Condition,
              Return_code
             );
```

Figure 19. ATBCMAS - Cleanup_Address_Space Service

Parameters

ASCB_ptr
Supplied parameter

- Type: Pointer
- Char Set: N/A

- Length: 32 bits

Specifies the pointer to the address space control block (ASCB) for the address space to be cleaned up. All conversations for all transaction program instances associated with this address space are to be deallocated. Invokers of this service can get this value from the PSAAOLD field in the PSA for the current address space or from the RMPLASCB field in the RMPL, resource manager parameter list. If this parameter is set to zero, the home address space of the program that issued the Cleanup_Address_Space call will be used as the default address space.

Condition
Supplied parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Specifies the deallocation condition that has occurred. This field is used to generate the TYPE of deallocate and sense code that is issued by APPC/MVS to the partner transaction program.

Valid values for this parameter are:

Value
Meaning

0
Normal

Specifies that the transaction program completed normally, even though it might have left active conversations. APPC/MVS deallocates all conversations in a proper state for normal deallocation with Deallocate Type(Sync_Level). All conversations not in the proper state for a normal deallocation are deallocated with Type(Abend_SVC).

1
System

Specifies that the transaction program terminated abnormally, or the transaction program was terminated on behalf of some action by the system (for example, the address space was cancelled or forced). This condition is normally detected by the transaction scheduler's subordinate address space. All active conversations are deallocated with Type(Abend_SVC).

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Cleanup_Address_Space might return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
Request accepted. All conversations owned by the address space are cleaned up asynchronously.

4
No conversations exist to be cleaned up.

8
The ASCB_ptr supplied does not point to a valid ASCB.

32
The requested service is not supported in the caller's environment. For example, this return code will be given if the caller invokes any of the transaction scheduler services while holding a lock.

44
APPC/MVS is not active.

48
APPC/MVS services failure.

Characteristics and Restrictions

1. Conversations with active APPC requests are not immediately deallocated. Once the partner TP responds, APPC/MVS returns a deallocate condition and deallocates the conversation locally.
2. When no APPC resources are to be cleaned up, Cleanup_Address_Space might access fields located through the ASCB_Ptr parameter before it establishes recovery (to improve performance). If an incorrect ASCB_ptr is passed to ATBCMAS, the caller may abend with completion code X'0C4' when ATBCMAS uses the passed value to get addressability to fields in the ASCB.
3. The Condition parameter defaults to zero (normal) if an incorrect condition is specified.
4. Transaction schedulers that call the Cleanup_Address_Space service while running in task mode should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.
5. Regardless of the condition parameter value specified for this service, APPC/MVS cleans up protected conversations differently, depending on whether a syncpoint operation is in progress. When a syncpoint operation **is** in progress for the current UR for the context with which the protected conversation is associated, APPC/MVS does not immediately deallocate the conversation. The syncpoint operation is allowed to complete. As part of the syncpoint processing, the protected conversation might be deallocated, in which case no further cleanup is required for that conversation.

   If the conversation was not deallocated, however, cleanup processing proceeds in the same manner as it does when a syncpoint operation **is not** in progress at the time the Cleanup service is issued:

   - The protected conversation is deallocated with TYPE(ABEND_SVC).
   - The current UR is put into backout-required state.
   - If the protected conversation is an inbound conversation, the logical unit of work ID (LUWID) for the next UR is reset.
   - The current UR and subsequent units of recovery for the context will not include the protected conversation being cleaned up by this service.

ATBCMTP— Cleanup_TP

Cleanup_TP can be used to request that APPC/MVS clean up all conversation resources associated with a transaction program instance. Conversation resources include network resources, control blocks, and buffers which are used by APPC/MVS to manage the transaction program instance and its conversations.

The Cleanup_TP service can be invoked for the following reasons:

- The transaction program requested by an inbound allocate request is not recognized or not available.
- The transaction scheduler cannot queue or schedule the transaction program at this time.
- The requesting user ID is not authorized to use the transaction program.
- The transaction program has been attached and executed, and has completed normally or abnormally.

The TP_ID is deleted from the system as a result of this call; this cleanup process might occur asynchronously.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task or SRB mode

Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBCMTP (TP_ID,
              Condition,
              Return_Code
             );
```

Figure 20. ATBCMTP - Cleanup_TP

Parameters

TP_ID
Supplied parameter

- Type: Character string
- Char Set: No restriction
- Length: 8 bytes

Specifies the transaction program instance that is to be cleaned up. The transaction program instance does not have to be associated with the caller's address space. All conversations owned by this transaction program instance are to be deallocated.

Condition
Supplied parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Specifies the deallocation condition that has occurred. This field is used to determine the type of deallocate and sense code that is issued by APPC/MVS to the partner transaction program.

Valid values for this parameter are:

Value
Meaning

0
Normal

Specifies that the transaction program completed normally, even though it might have left active conversations. APPC/MVS deallocates all conversations in a proper state for normal deallocation with Deallocate Type(Sync_Level). All conversations not in the proper state for a normal deallocation are deallocated with Type(Abend_SVC).

1
System

Specifies that the transaction program terminated abnormally, or the transaction program was terminated on behalf of some action by the system (for example, the address space was cancelled or forced). This condition is normally detected by the transaction scheduler's subordinate address space. All active conversations are deallocated with Type(Abend_SVC).

2
TP_Not_Available_No_Retry

Specifies that the transaction scheduler is not able to schedule the transaction because of a condition that is not temporary. The partner should not attempt to retry the request. APPC/MVS deallocates the conversation with a sense code of X'084C0000'.

3
TP_Not_Available_Retry

Specifies that the transaction scheduler is not able to schedule the transaction because of a condition that might be temporary. The partner might attempt to retry the request. APPC/MVS deallocates the conversation with a sense code of X'084B6031'X.

4
TPN_Not_Recognized

Specifies that the transaction scheduler does not recognize the TP_Name passed to it. APPC/MVS deallocates the conversation with a sense code of X'10086021'.

5
Security_Not_Valid

Specifies that the transaction scheduler detected a security violation. APPC/MVS deallocates the conversation with a sense code of X'080F6051'.

6
Sync_Level_Not_Supported_Pgm

Specifies that the transaction program does not support the level of synchronization requested by the sender. APPC/MVS deallocates the conversation with a sense code of X'10086041'.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Cleanup_TP might return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
Request accepted. All conversations owned by the transaction program instance will be cleaned up asynchronously.

4
No conversations exist to be cleaned up.

8
The TP_ID parameter specified a nonexistent transaction program instance.

32
The requested service is not supported in the caller's environment. For example, this return code will be given if the caller invokes any of the transaction scheduler services while holding a lock.

44
APPC/MVS is not active.

48
APPC/MVS services failure.

Characteristics and Restrictions

1. Conversations with active APPC requests are not immediately deallocated. Once the partner TP responds, APPC/MVS returns a deallocate condition and deallocates the conversation locally.

2. The Condition parameter defaults to zero (normal) if the specified condition is not valid.
3. If you call the Cleanup_TP service while a unit of work is waiting on an ECB as a result of an asynchronous call, APPC/MVS does not post the ECB after performing the Cleanup_TP operation (APPC/MVS considers all resources associated with the TP "terminated"). The application's recovery environment must clean up the waiting ECB.
4. Transaction schedulers that call the Cleanup_TP service while running in task mode should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.
5. Regardless of the condition parameter value specified for this service, APPC/MVS cleans up protected conversations differently, depending on whether a syncpoint operation is in progress. When a syncpoint operation **is** in progress for the current UR for the context with which the protected conversation is associated, APPC/MVS does not immediately deallocate the conversation. The syncpoint operation is allowed to complete. As part of the syncpoint processing, the protected conversation might be deallocated, in which case no further cleanup is required for that conversation.

   If the conversation was not deallocated, however, cleanup processing proceeds in the same manner as it does when a syncpoint operation **is not** in progress at the time the Cleanup service is issued:

   - The protected conversation is deallocated with TYPE(ABEND_SVC).
   - The current UR is put into backout-required state.
   - If the protected conversation is an inbound conversation, the logical unit of work ID (LUWID) for the next UR is reset.
   - The current UR and subsequent units of recovery for the context will not include the protected conversation being cleaned up by this service.

ATBCTP1— Cleanup_TP

Cleanup_TP can be used to request that APPC/MVS clean up all conversation resources associated with a transaction program instance. Conversation resources include network resources, control blocks, and buffers that are used by APPC/MVS to manage the transaction program instance and its conversations.

The Cleanup_TP service might be invoked for the following reasons:

- The transaction program requested by an inbound allocate request is not recognized or not available.
- The transaction scheduler cannot queue or schedule the transaction program at this time.
- The requesting user ID is not authorized to use the transaction program
- The transaction program has been attached and executed, and has completed normally or abnormally.

The TP_ID is deleted from the system as a result of this call; this cleanup process may occur asynchronously.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task or SRB mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked

Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBCTP1 (TP_ID,
              Condition,
              Notify_Type,
              Return_Code
             );
```

Figure 21. ATBCTP1 - Cleanup_TP

Parameters

TP_ID
Supplied parameter

- Type: Character string
- Char Set: No restriction
- Length: 8 bytes

Specifies the transaction program instance that is to be cleaned up. The transaction program instance does not have to be associated with the caller's address space. All conversations owned by this transaction program instance are to be deallocated.

Condition
Supplied parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Specifies the deallocation condition that has occurred. This field is used to determine the type of deallocate and sense code that is issued by APPC/MVS to the partner transaction program.

Valid values for this parameter are:

Value
Meaning

0
Normal

Specifies that the transaction program completed normally, even though it might have left active conversations. APPC/MVS deallocates all conversations in a proper state for normal deallocation with Deallocate Type(Sync_Level). All conversations not in the proper state for a normal deallocation are deallocated with Type(Abend_SVC).

1
System

Specifies that the transaction program terminated abnormally, or the transaction program was terminated on behalf of some action by the system (for example, the address space was cancelled or forced). This condition is normally detected by transaction scheduler's subordinate address space. All active conversations are deallocated with TYPE(ABEND_SVC).

2
TP_Not_Available_No_Retry

Specifies that the transaction scheduler is not able to schedule the transaction because of a condition that is not temporary. The partner should not attempt to retry the request. APPC/MVS deallocates the conversation with a sense code of X'084C0000'.

3
TP_Not_Available_Retry

Specifies that the transaction scheduler is not able to schedule the transaction because of a condition that might be temporary. The partner can attempt to retry the request. APPC/MVS deallocates the conversation with a sense code of X'084B6031'X.

4
TPN_Not_Recognized

Specifies that the transaction scheduler does not recognize the TP_Name passed to it. APPC/MVS deallocates the conversation with a sense code of X'10086021'.

5
Security_Not_Valid

Specifies that the transaction scheduler detected a security violation. APPC/MVS deallocates the conversation with a sense code of X'080F6051'.

6
Sync_Level_Not_Supported_Pgm

Specifies that the transaction program does not support the level of synchronization requested by the sender. APPC/MVS deallocates the conversation with a sense code of X'10086041'.

7
User_Not_Authorized_For_TP

Specifies that the user is not authorized to access the transaction program. APPC/MVS deallocates the conversation with a sense code of X'080F0983'.

Notify_type
Supplied parameter

- Type: Structure
- Char Set: N/A
- Length: 4-8 bytes

Specifies the type of processing and notification (synchronous or asynchronous) requested for this service. Programs can request asynchronous processing, which returns control to the program immediately and later notifies the program by ECB when the service is complete. The possible types are:

- None

  No notification is requested. The service is performed synchronously, and control is returned to the caller when processing is complete. All returned parameters are set on return to the caller. To specify no notification, set the parameter value to a four-byte structure containing binary zeros.
- ECB

  Programs can request asynchronous processing by specifying an ECB to be posted when processing completes. To specify an ECB, set the parameter to an eight-byte structure containing a fullword binary one (X'00000001') followed by the address of a fullword area to be used as the ECB. The ECB must reside in the home address space.

  When you specify an ECB, control is returned before processing is complete, with only the return code set. If the asynchronous request was accepted, the return code is set to 0 to indicate that the service is being processed asynchronously. Other returned parameters are filled in during asynchronous processing, and the specified ECB is posted when all returned parameters are set. The completion code field in the ECB contains the return code for the service.

Return_code
Returned parameter

- Type: Integer

- Char Set: N/A
- Length: 32 bits

Cleanup_TP may return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
Request accepted. All conversations owned by the transaction program instance will be cleaned up asynchronously.

4
No conversations exist to be cleaned up.

8
The TP_ID parameter specified a nonexistent transaction program instance.

12
The asynchronous request failed. Resubmit the request with a Notify_Type of None or report the problem to IBM.

20
APPC/MVS was cancelled during an asynchronous request for this service.

32
The requested service is not supported in the caller's environment. For example, this return code will be given if the caller invokes any of the transaction scheduler services while holding a lock.

44
APPC/MVS is not active.

48
APPC/MVS services failure.

Characteristics and Restrictions

1. Conversations with active APPC requests are not immediately deallocated. Once the partner TP responds, APPC/MVS returns a deallocate condition and deallocates the conversation locally.
2. The Condition parameter defaults to 0 (normal) if an invalid condition is specified.
3. If you call the Cleanup_TP service while a unit of work is waiting on an ECB as a result of an asynchronous call, APPC/MVS does not post the ECB after performing the Cleanup_TP operation (APPC/MVS considers all resources associated with the TP “terminated”). The application's recovery environment must clean up the waiting ECB.
4. Transaction schedulers that call the Cleanup_TP service while in task mode should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.
5. Regardless of the condition parameter value specified for this service, APPC/MVS cleans up protected conversations differently, depending on whether a syncpoint operation is in progress. When a syncpoint operation **is** in progress for the current UR for the context with which the protected conversation is associated, APPC/MVS does not immediately deallocate the conversation. The syncpoint operation is allowed to complete. As part of the syncpoint processing, the protected conversation might be deallocated, in which case no further cleanup is required for that conversation.

   If the conversation was not deallocated, however, cleanup processing proceeds in the same manner as it does when a syncpoint operation **is not** in progress at the time the Cleanup service is issued:

   - The protected conversation is deallocated with TYPE(ABEND_SVC).
   - The current UR is put into backout-required state.
   - If the protected conversation is an inbound conversation, the logical unit of work ID (LUWID) for the next UR is reset.

- The current UR and subsequent units of recovery for the context will not include the protected conversation being cleaned up by this service.

ATBIDEN— Identify

The Identify service is used by a transaction scheduler to make itself known to APPC/MVS. A transaction scheduler issues Identify after it has initialized itself and is ready to receive or schedule requests from APPC/MVS. The transaction scheduler must supply an XCF member token on Identify to allow APPC/MVS to communicate with it. A transaction scheduler must identify itself to APPC/MVS before its subordinate address spaces can connect to APPC/MVS.

Specifically, this service is used by a transaction scheduler to do the following:

1. Identify itself to APPC/MVS.
2. Provide its XCF member token to APPC/MVS so that it can be notified of inbound Allocate requests.
3. Optionally identify an “information extract exit” that can be invoked by APPC/MVS when it needs information from the transaction scheduler.
4. Determine whether the APPCPMxx parmlib member correctly defines the LUs for the transaction scheduler.
5. Specify initial status for LUs belonging to the transaction scheduler.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBIDEN (Scheduler_name,
              Scheduler_extract_exit_addr,
              Scheduler_extract_user_field,
              Scheduler_member_token,
              TP_profile_processing,
              LU_initial_status,
              Return_code
             );
```

Figure 22. ATBIDEN - Identify

Parameters

Scheduler_Name
Supplied parameter

- Type: Character string
- Char Set: 01134
- Length: 8 bytes

Specifies the name of the transaction scheduler. This field must match a transaction scheduler name appearing in the LU definitions of an APPCPMxx parmlib member. The value must be the same as the value of the SCHED keyword of one or more LUADD statements in APPCPMxx. The transaction scheduler name will also be used for operator displays. If the transaction scheduler runs only as a "single instance per system," this value should be a string which suggests the name of the component performing the Identify (for example, "ASCH" is an abbreviation used to identify the APPC transaction scheduler). If the transaction scheduler can run as "multiple copies per system," this value should be a string which identifies a particular copy of the transaction scheduler (for example, subsystems might wish to use the subsystem name which appears in the IEFSSNxx parmlib member). Once a transaction scheduler has successfully been identified, no other Identify call using the same Scheduler_Name will be accepted unless a corresponding Unidentify statement is issued.

Scheduler_Extract_Exit_Addr
Supplied parameter

- Type: Address
- Char set: N/A
- Length: 32 bits

Specifies the address of the transaction scheduler's information extract exit. This is an optional exit and can be left zero. See "Extract Exit" on page 57 for information about coding a transaction scheduler extract exit.

Scheduler_Extract_User_Field
Supplied parameter

- Type: Character string
- Char set: No restriction
- Length: 8 bytes

Specifies a user defined field or token passed to the transaction scheduler's information extract exit.

Scheduler_Member_Token
Supplied parameter

- Type: Character string
- Char set: No restriction
- Length: 8 bytes

Specifies an XCF member token. The member token represents a member of the XCF group that is joined when the Join_Sysappc_Group service is invoked. Messages are sent to this member to report when the transaction scheduler's LU name is activated or deactivated. Messages are also sent to report the arrival of inbound Allocate requests. APPC/MVS does not check the validity of this member token. If a transaction scheduler passes an unknown member token, then the transaction scheduler will not receive notification of the arrival of inbound Allocate requests.

TP_Profile_Processing
Supplied parameter

- Type: Integer
- Char set: N/A
- Length: 32 bits

Specifies the TP_Profile processing characteristics to use for this transaction scheduler.

Valid values for this parameter are:

Value
Meaning

0
Required

Specifies that APPC/MVS should reject any inbound Allocate request that specifies a TP_Name for which a TP_Profile entry does not exist. If a TP_Profile entry does not exist, the inbound Allocate request is rejected with TP_Not_Recognized (sense code X'10086021').

1
Optional

Specifies that a TP_Profile entry is not required. APPC/MVS will perform all validity and security checks, and will reject the request if any of these checks fail. If a TP_Profile entry does not exist, APPC/MVS will indicate this in the XCF message sent to the transaction scheduler to notify it of the inbound Allocate request.

LU_Initial_Status
Supplied parameter

- Type: Integer
- Char set: N/A
- Length: 32 bits

Specifies the initial status of LUs controlled by this transaction scheduler. Any additional LUs being added for this transaction scheduler after Identify will initially be given this status, unless Control Halt_All or Resume_All is called to set the status.

Valid values for this parameter are:

Value
Meaning

0
Active

Specifies that APPC/MVS should activate the LU or LUs controlled by this transaction scheduler. The status of every LU controlled by this transaction scheduler will initially be put into Active state.

1
Outbound_Only

Specifies that APPC/MVS should temporarily halt processing of Allocate requests to the LU or LUs controlled by this transaction scheduler. The transaction scheduler has to call Control Resume in order for the LU to begin accepting inbound requests. The status of every LU controlled by this transaction scheduler, whether it is added to the system at initialization or by a subsequent SET command, will initially be put into Outbound_Only state, unless Control Resume_All is called to set the status.

When the APPC address space terminates and restarts, the transaction schedulers that have done Identify and Connect before have to reidentify themselves and reconnect all their subordinate address spaces. A transaction scheduler can use this option to temporarily halt processing of inbound Allocate requests to the LU while it is in the process of reconnecting its subordinate address spaces. It can issue a Control Resume request to activate all the LUs when the reconnect process is finished.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Identify may return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
The Identify request was accepted. The LUs are activated asynchronously.

4
The Identify request was accepted. No base LU name is present. The APPCPMxx parmlib member or members specify at least one LU name that is controlled by the transaction scheduler, but no LU name is designated as the transaction scheduler's base LU. This situation may arise because the APPCPMxx parmlib member was incorrectly coded, or because the installation has deliberately chosen this configuration.

8
The Identify request was accepted. No LU names are applicable. APPC/MVS found that the APPCPMxx parmlib member specifies no LU names that are controlled by the transaction scheduler. This situation may arise because the APPCPMxx parmlib member did not specify the correct transaction scheduler name on the SCHED keyword of LUADD, or it may arise because APPC/MVS tried to initialize for the specified LU name and encountered a failure (for example, APPC/MVS was unable to open the required TP profile file).

12
The Identify request was rejected. The calling transaction scheduler address space is already identified using the same scheduler name as the Scheduler_name parameter passed in. This may occur if the caller issued Identify twice with the same scheduler name.

14
The Identify request was rejected. The calling transaction scheduler address space is already identified using a different scheduler name from the Scheduler_name parameter passed in. This may occur if the caller issued Identify twice with different scheduler names.

16
The Identify request was rejected. The Scheduler_Name parameter of Identify is already in use by some other address space that previously issued Identify.

20
The Identify request was rejected. The Scheduler_Name parameter value is not valid.

24
The Identify request was rejected. The TP_Profile_Processing parameter value is not valid.

28
The Identify request was rejected. The LU_Initial_Status parameter value is not valid.

32
The requested service is not supported in the caller's environment. For example, this return code is given if the caller invokes any of the transaction scheduler services while holding a lock.

38
The requested transaction scheduler service cannot be invoked from a subordinate address space, or an address space that has outstanding APPC/MVS conversations.

40
The requested transaction scheduler service cannot be invoked from an APPC/MVS server address space.

44
APPC/MVS is not active.

48
APPC/MVS services failure.

Usage Notes

1. The transaction scheduler will be notified of an inbound Allocate request only if the request passes all validity and security checks. The userid specified in the request must have RACF authority to access the TP profile entry (whether or not it exists) and if the TP profile entry is found it must be marked “activated”.
2. Timing restrictions on activities after Identify

   The transaction scheduler may create subordinate address spaces and invoke Connect before APPC/MVS reports that the base LU was successfully initialized. However, the transaction scheduler

must not dispatch any work that might invoke an APPC/MVS Allocate service in these subordinate address spaces, before one of the following occurs:

- The base LU is successfully initialized.
- ATBSASA is called to prevent Allocated conversations being associated with the system default LU. For more information about this option, see "Set_AS_Attributes" on page 47.

3. Factors delaying asynchronous completion of Identify

   Some conditions may substantially delay the activation of an LU; for example, VTAM may be stopped when the Identify is accepted.

   An XCF message will be sent to the XCF-member representing the transaction scheduler when each of its LUs is activated.

4. Factors causing asynchronous failure of Identify

   Some conditions might cause an Identify to fail asynchronously after it has been accepted; for example, VTAM parameters might be mismatched (there might not be an APPL macro for the specified LU name), or APPC/MVS might not be able to open the specified TP profile file.

   An XCF message will be sent to the XCF-member representing the transaction scheduler when the attempt to initialize an LU fails asynchronously.

   A transaction scheduler address space must issue Unidentify to undo its Identify even if all of its LUs fail asynchronously.

   When LU initialization fails asynchronously, APPC/MVS will issue error-messages indicating the cause of the failure (for example, unable to open the TP profile file). These messages will be issued to the same operator who receives messages about failures of LUs after initialization is completed.

5. Use of XCF by a transaction scheduler

   See "ATBMIGRP— Join_Sysappc_Group" on page 89 for information regarding joining an XCF group.

6. Asynchronous initialization of the base LU name

   If Identify produces a return code of zero, then the transaction scheduler issuing Identify will receive an LU activation or LU deactivation message, with LU_Flags indicating that the message describes the base LU name. An LU deactivation message will indicate asynchronous failure of the attempt to initialize the LU name; an LU activation message will indicate successful initialization of the LU name.

7. Operation without a base LU name

   If Identify produces a return code of 4, then the transaction scheduler will receive neither an LU activation nor an LU deactivation XCF message for the base LU name, unless the operator issues a SET command that establishes a base LU name for the transaction scheduler.

   APPC/MVS does not issue any operator message indicating that the operator should do this; the transaction scheduler might wish to issue its own operator message asking the operator to perform such a SET command.

8. Operation with no LU names

   If Identify produces a return code of 8, then the transaction scheduler will receive neither an LU activation nor an LU deactivation message for the base LU name, unless the operator issues a SET command that establishes a base LU name for the transaction scheduler.

   In this case, APPC/MVS issues an operator message telling the operator to perform such a SET command.

Characteristics and Restrictions

1. Identify performs an automatic Connect of the home address space of the calling transaction scheduler. (See "Connect" on page 30.)
2. APPC/MVS supports one Identify per address space. Because of this, each transaction scheduler must be in its own address space.

3. The Identify service causes APPC/MVS to open a VTAM ACB or ACBs for the transaction scheduler's LUs. The ACB or ACBs are opened asynchronously if the Identify is accepted. Similarly, the TP profile file or files are also opened asynchronously. The asynchronous OPEN lets a transaction scheduler identify itself when VTAM is functioning. APPC/MVS informs a transaction scheduler that its LU is operational.
4. As soon as APPC/MVS accepts the Identify request, the scheduler's corresponding XCF message user routine and information extract exit can be invoked at any time.
5. An APPC/MVS server address space cannot use the Identify service. If an address space calls the Identify service while it is registered for an Allocate queue, the system does not perform the Identify service function, and the caller receives a return code of 40 (decimal). For information about APPC/MVS servers, see *z/OS MVS Programming: Writing Servers for APPC/MVS*.
6. Transaction schedulers that call the Identify service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.

ATBIDN1— Identify

The Identify service is used by a transaction scheduler to make itself known to APPC/MVS. A transaction scheduler issues Identify after it has initialized itself and is ready to receive or schedule requests from APPC/MVS. The transaction scheduler must supply an XCF member token on Identify to allow APPC/MVS to communicate with it. A transaction scheduler must identify itself to APPC/MVS before its subordinate address spaces can connect to APPC/MVS.

Specifically, this service is used by a transaction scheduler to do the following:

1. Identify itself to APPC/MVS.
2. Provide its XCF member token to APPC/MVS so that it can be notified of inbound Allocate requests.
3. Optionally identify an information extract exit that may be invoked by APPC/MVS when it needs information from the transaction scheduler.
4. Determine whether the APPCPMxx parmlib member correctly defines the LUs for the transaction scheduler.
5. Specify initial status for LUs belonging to the transaction scheduler.
6. Identify an exit to convert a TP profile the first time it is referenced, and store the converted profile for future references.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBIDN1 (Scheduler_name,
              Scheduler_extract_exit_addr,
              Scheduler_extract_user_field,
              Scheduler_member_token,
              TP_profile_processing,
              LU_initial_status,
              Scheduler_TP_profile_exit,
              Scheduler_TP_profile_exit_data,
              Return_codes
             );
```

Figure 23. ATBIDN1 - Identify

Parameters

Scheduler_Name
Supplied parameter

- Type: Character String
- Char Set: 01134
- Length: 8 bytes

Specifies the name of the transaction scheduler. This field must match a transaction scheduler name appearing in the LU definitions of an APPCPMxx parmlib member. The value must be the same as the value of the SCHED keyword of one or more LUADD statements in APPCPMxx. The transaction scheduler name will also be used for operator displays.

If the transaction scheduler runs only as a “single instance per system,” this value should be a string that suggests the name of the component performing the Identify (for example, “ASCH” is an abbreviation used to identify the APPC transaction scheduler). If the transaction scheduler can run as “multiple copies per system,” this value should be a string that identifies a particular copy of the transaction scheduler (for example, subsystems may wish to use the subsystem name that appears in the IEFSSNxx parmlib member).

Once a transaction scheduler has successfully been identified, no other Identify call using the same Scheduler_Name will be accepted unless a corresponding Unidentify statement is issued.

Scheduler_Extract_Exit_Addr
Supplied parameter

- Type: Address
- Char set: N/A
- Length: 32 bits

Specifies the address of the transaction scheduler's information extract exit. This is an optional exit and may be left zero. See “Extract Exit” on page 57 for information about the requirements for and processing of a transaction scheduler extract exit.

Scheduler_Extract_User_Field
Supplied parameter

- Type: Character string
- Char set: No restriction
- Length: 8 bytes

Specifies a user-defined field or token passed to the transaction scheduler's information extract exit.

Scheduler_Member_Token
Supplied parameter

- Type: Character string
- Char set: No restriction

- Length: 8 bytes

Specifies an XCF member token. The member token represents a member of the XCF group that is joined when the Join_Sysappc_Group service is invoked. Messages are sent to this member to report when the transaction scheduler's LU name is activated or deactivated. Messages are also sent to report the arrival of inbound Allocate requests. APPC/MVS does not check the validity of this member token. If a transaction scheduler passes an unknown member token, then the transaction scheduler will not receive notification of the arrival of inbound Allocate requests.

TP_Profile_Processing
Supplied parameter

- Type: Integer
- Char set: N/A
- Length: 32 bits

Specifies the TP_Profile processing characteristics to use for this transaction scheduler.

Valid values for this parameter are:

Value
Meaning

0
Required

Specifies that APPC/MVS should reject any inbound Allocate request that specifies a TP_Name for which a TP_Profile entry does not exist. If a TP_Profile entry does not exist, the inbound Allocate request is rejected with TP_Not_Recognized (sense code X'10086021').

1
Optional

Specifies that a TP_Profile entry is not required. APPC/MVS will perform all validity and security checks and reject the request if any of these checks fail. If a TP_Profile entry does not exist, APPC/MVS will indicate this in the XCF message sent to the transaction scheduler to notify it of the inbound Allocate request.

LU_Initial_Status
Supplied parameter

- Type: Integer
- Char set: N/A
- Length: 32 bits

Specifies the initial status of LUs controlled by this transaction scheduler. Any additional LUs being added for this transaction scheduler after Identify will initially be given this status, unless Control Halt_All or Resume_All is called to set the status.

Valid values for this parameter are:

Value
Meaning

0
Active

Specifies that APPC/MVS should activate the LU(s) controlled by this transaction scheduler. The status of every LU controlled by this transaction scheduler will initially be put into Active state.

1
Outbound_Only

Specifies that APPC/MVS should temporarily halt processing of Allocate requests to the LU or LUs controlled by this transaction scheduler. For the LU to begin accepting inbound requests, the transaction scheduler has to call the Control service for the Resume_All_Input function. The

status of every LU controlled by this transaction scheduler, whether it is added to the system at initialization or by a subsequent SET command, will initially be put into Outbound_Only state, unless Control Resume_All_Input is called to set the status.

When the APPC address space terminates and restarts, the transaction schedulers that have called Identify and Connect before have to reidentify themselves and reconnect all their subordinate address spaces. A transaction scheduler can use this option to temporarily halt processing of inbound Allocate requests to the LU while it is in the process of reconnecting its subordinate address spaces. It can issue a Control Resume request to activate all the LUs when the reconnect process is finished.

Scheduler_TP_profile_exit
Supplied parameter

- Type: Character string
- Char set: 01134
- Length: 8 bytes

Specifies the name of the exit that will receive control when the TP profile requires conversion. To specify no exit, set this parameter to 8 blanks. For more information about the requirements for and processing of this exit, see "TP Profile Conversion Exit" on page 59.

Scheduler_TP_profile_exit_data
Supplied parameter

- Type: Character string
- Char set: No restriction
- Length: 8 bytes

Specifies data to be passed to the TP profile conversion exit each time it is invoked; for example, the address of a workarea for the exit to use. For more information about how the exit receives this input data, see "TP Profile Conversion Exit" on page 59.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Identify might return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
The Identify request was accepted. The LUs are activated asynchronously.

4
The Identify request was accepted. No base LU name is present. The APPCPMxx parmlib member or members specify at least one LU name that is controlled by the transaction scheduler, but no LU name is designated as the transaction scheduler's base LU. This situation might arise because the APPCPMxx parmlib member was incorrectly coded, or because the installation has deliberately chosen this configuration.

8
The Identify request was accepted. No LU names are applicable. APPC/MVS found that the APPCPMxx parmlib member specifies no LU names that are controlled by the transaction scheduler. This situation might arise because the APPCPMxx parmlib member did not specify the correct transaction scheduler name on the SCHED keyword of LUADD, or it might arise because APPC/MVS tried to initialize for the specified LU name and encountered a failure (for example, APPC/MVS was unable to open the required TP profile file).

12
The Identify request was rejected. The calling transaction scheduler address space is already identified using the same scheduler name as the Scheduler_name parameter passed in. This may occur if the caller issued Identify twice with the same scheduler name.

14
The Identify request was rejected. The calling transaction scheduler address space is already identified using a different scheduler name from the Scheduler_name parameter passed in. This may occur if the caller issued Identify twice with different scheduler names.

16
The Identify request was rejected. The Scheduler_Name parameter of Identify is already in use by some other address space that previously issued Identify.

18
The Identify request was rejected. The Scheduler_TP_profile_exit name that was passed could not be loaded.

20
The Identify request was rejected. The Scheduler_Name parameter value is not valid.

22
The Identify request was rejected. The Scheduler_TP_profile_exit name is not valid.

24
The Identify request was rejected. The TP_Profile_Processing parameter value is not valid.

28
The Identify request was rejected. The LU_Initial_Status parameter value is not valid.

32
The requested service is not supported in the caller's environment. For example, this return code is given if the caller invokes any of the transaction scheduler services while holding a lock.

38
The requested transaction scheduler service cannot be invoked from a subordinate address space, or an address space that has outstanding APPC/MVS conversations.

40
The requested transaction scheduler service cannot be invoked from an APPC/MVS server address space.

44
APPC/MVS is not active.

48
APPC/MVS services failure.

Usage Notes

1. The transaction scheduler will be notified of an inbound Allocate request only if the request passes all validity and security checks. The userid specified in the request must have RACF authority to access the TP profile entry (whether or not it exists), and if the TP profile entry is found, it must be marked “activated”.
2. Timing restrictions on activities after Identify

   The transaction scheduler might create subordinate address spaces and call Connect before APPC/MVS reports that the base LU was successfully initialized. However, the transaction scheduler must not dispatch any work that might invoke an APPC/MVS Allocate service in these subordinate address spaces, before one of the following occurs:

   - The base LU is successfully initialized
   - ATBSASA is called to prevent allocated conversations being associated with the system default LU. For more information about this option, see “Set_AS_Attributes” on page 47.
3. Factors delaying asynchronous completion of Identify

Some conditions might substantially delay the activation of an LU; for example, VTAM may be stopped when the Identify is accepted.

An XCF message will be sent to the XCF-member representing the transaction scheduler when each of its LUs is activated.

4. Factors causing asynchronous failure of Identify

   Some conditions might cause an Identify to fail asynchronously after it has been accepted, for example, VTAM parameters might be mismatched (there might not be an APPL macro for the specified LU name), or APPC/MVS may be unable to open the specified TP profile file.

   An XCF message will be sent to the XCF member representing the transaction scheduler when the attempt to initialize an LU fails asynchronously.

   A transaction scheduler address space must issue Unidentify to undo its Identify, even if all of its LUs fail asynchronously.

   When LU initialization fails asynchronously, the system issues error messages indicating the cause of the failure (for example, unable to open the TP profile file). These messages will be issued to the same operator who receives messages about failures of LUs after initialization is completed.

5. Use of XCF by a transaction scheduler

   See “Join_Sysappc_Group” on page 44 for information regarding joining an XCF group.

6. Asynchronous initialization of the base LU name

   If Identify produces a return code of zero, then the transaction scheduler issuing Identify will receive an LU activation or LU deactivation message, with LU_Flags indicating that the message describes the base LU name. An LU deactivation message will indicate asynchronous failure of the attempt to initialize the LU name; an LU activation message will indicate successful initialization of the LU name.

7. Operation without a base LU name

   If Identify produces a return code of 4, then the transaction scheduler will receive neither an LU activation nor an LU deactivation XCF message for the base LU name, unless the operator issues a SET command which establishes a base LU name for the transaction scheduler.

   APPC/MVS does not issue any operator message indicating that the operator should do this; the transaction scheduler can issue its own operator message asking the operator to perform such a SET command.

8. Operation with no LU names

   If Identify produces a return code of 8, then the transaction scheduler will receive neither an LU activation nor an LU deactivation message for the base LU name, unless the operator issues a SET command that establishes a base LU name for the transaction scheduler.

   In this case, APPC/MVS issues an operator message telling the operator to perform such a SET command.

Characteristics and Restrictions

1. Identify performs an automatic Connect of the home address space of the calling transaction scheduler. (See “Connect” on page 30.)
2. APPC/MVS supports one Identify per address space. Because of this, each transaction scheduler must be in its own address space.
3. The Identify service causes APPC/MVS to open one or more VTAM ACBs for the transaction scheduler's LUs. The ACBs are opened asynchronously if the Identify is accepted. Similarly, the TP profile file or files are also opened asynchronously. The asynchronous OPEN lets a transaction scheduler identify itself when VTAM is functioning. APPC/MVS informs a transaction scheduler that its LU is operational.
4. As soon as APPC/MVS accepts the Identify request, the scheduler's corresponding XCF message user routine and information extract exit may be invoked at any time.

5. Transaction schedulers that call the Identify service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.
6. An APPC/MVS server address space cannot use the Identify service. If an address space calls the Identify service while it is registered for an allocate queue, the system does not perform the Identify service function, and the caller receives a return code of 40 (decimal). For information about APPC/MVS servers, see *z/OS MVS Programming: Writing Servers for APPC/MVS*.

ATBMIGRP— Join_Sysappc_Group

Note: The ATBJGP1 call is the preferred programming interface for this service.

Use ATBMIGRP to join the XCF group used by APPC/MVS. Each transaction scheduler must join the APPC XCF group. Other system applications can also join the APPC XCF group to be notified of APPC events.

APPC/MVS communicates with members of its XCF group by invoking their XCF message user routines. The APPC/MVS notifies all group members of general interest events such as APPC initialization and termination. APPC/MVS also notifies individual transaction schedulers when inbound allocate requests arrive for them. To notify individual schedulers, APPC/MVS uses a member_token that the transaction scheduler passes in on the Identify service. A transaction scheduler must call the Join_Sysappc_Group service, which provides the member token, before calling the Identify service. Unlike Identify and most other scheduler services, the Join_Sysappc_Group service can be called when the APPC/MVS is not active.

If you do not use the Join_Sysappc_Group service to join the APPC XCF group, you must use APPC_GROUP_NAME as the group name with the IXCJOIN macro. A different group name is chosen on each system; therefore, each such group is “local to a system” and APPC/MVS can use the facilities of XCF regardless of whether XCF can perform cross-system communication. Also, the service performs IXCJOIN with the LASTING=NO option; thus, XCF “system-local mode” can be tolerated.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	PASN = HASN = SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBMIGRP (XCFMSGIN_exit_address,
               XCFMSGIN_memdata,
               Member_token,
               XCF_return_code,
               XCF_reason_code,
               Return_code
              );
```

Figure 24. ATBMIGRP - Join_Sysappc_Group

Parameters

XCFMSGIN_exit_address
Supplied parameter

- Type: Address
- Char Set: N/A
- Length: 32 bits

XCFMSGIN_exit_address specifies the address of the transaction scheduler's XCF message user routine. The routine takes control when a message becomes available for this member from another member of the group. For details about the message user routine, see “XCF Message User Routine” on page 53.

XCFMSGIN_memdata
Supplied parameter

- Type: Character
- Char Set: No restriction
- Length: 8 bytes

XCFMSGIN_memdata is an optional parameter that specifies an 8 byte member data field. This field is provided to the message user routine for this member. If you do not specify a value, XCF sets the member data field to binary zero. The transaction scheduler can use this field to pass the address and ASID or ALET of a particular control structure to the XCF message user routine.

Member_token
Returned parameter

- Type: Character
- Char Set: No restriction
- Length: 8 bytes

Member_token specifies the location where this service places the member token that represents the caller of this service.

XCF_return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

The return code passed back from the XCF IXCJOIN macro, if XCF rejects the Join request.

XCF_reason_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

The reason code passed back from the XCF IXCJOIN macro, if XCF rejects the Join request.

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Join_Sysappc_Group might return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
Request successful.

8
Request unsuccessful - XCF failed or request denied by XCF.

40
The caller was not running in supervisor state or PSW key 0-7.

48
APPC/MVS services failure.

Characteristics and Restrictions

1. This service will execute successfully even if XCF is operating in XCF local mode.
2. The caller must issue the IXCLEAVE macro to undo the effects of Join_Sysappc_Group. IXCLEAVE processing is performed automatically if the caller's address space or task terminates.
3. The message buffer that is provided in the message user routine must be accessible using the same protect key that is in effect at invocation of Join_Sysappc_Group.
4. The task that calls this service might end abnormally if a privileged program issues the XCF IXCTERM macro against this member. In that case, the task terminates with system completion code 00C, reason code 4, and the task's recovery routine cannot retry. Transaction schedulers can handle this by attaching a subtask that invokes Join_Sysappc_Group, and reattaching the subtask if it terminates with completion code 00C, reason code 4.
5. A transaction scheduler can join XCF groups other than the APPC group joined by this service.
6. The name of APPC's XCF group might vary from system to system and might change during re-IPL. If you need to know the XCF group name used by APPC, for example, to dedicate specific resources to it, you can use the ATBAPPCA mapping macro, which appears in *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).
7. Transaction schedulers that call the Join_Sysappc_Group service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.

ATBUNID— Unidentify

Note: The ATBUID1 call is the preferred programming interface for this service.

Unidentify can be used by a transaction scheduler to reverse the effect of invocation of the Identify service. Unidentify terminates all APPC services for the specified transaction scheduler and its subordinate address spaces.

After performing Unidentify, a transaction scheduler can invoke the IXCLEAVE macro to undo the effects of its invocation of Join_Sysappc_Group.

APPC/MVS asynchronously shuts down the LU or LUs assigned to the transaction scheduler that called Unidentify. The calling program does not have to wait for this to occur. Once the Unidentify request is accepted, APPC/MVS returns control to the calling program and assumes responsibility for taking down the LU or LUs.

Shut down automatically disconnects address spaces currently connected to the issuing transaction scheduler. Shutting down an LU also includes setting the session limits to zero. Conversations that are currently running will run to completion. Any outstanding transaction program allocate requests will not be honored. Upon completion of all of the LU's conversations, the ACB is closed. The LU is then placed in pending state, to await another Identify request. A transaction scheduler must issue Identify if it is to restart.

The Unidentify must be issued from the address space that issued the Identify.

Environment

Authorization:	Supervisor state or PSW key 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	PASN = HASN = SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL ATBUNID (Return_Code
             );
```

Figure 25. ATBUNID - Unidentify

Parameters

Return_code
Returned parameter

- Type: Integer
- Char Set: N/A
- Length: 32 bits

Unidentify might return one of the following decimal values in the return code parameter:

Decimal
Meaning

0
Unidentify was accepted.

32
The requested service is not supported in the caller's environment. For example, this return code will be given if the caller invokes any of the transaction scheduler services while holding a lock.

34
The requested transaction scheduler service must be invoked from a transaction scheduler address space.

44
APPC/MVS is not active.

48
APPC/MVS services failure.

Characteristics and Restrictions

Transaction schedulers that call the Unidentify service should not have any enabled unlocked task (EUT) functional recovery routines (FRRs) established. For more information about EUT FRRs, see the information on providing recovery in *z/OS MVS Programming: Authorized Assembler Services Guide*.

Appendix C. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the Contact z/OS web page (www.ibm.com/systems/z/os/zos/webqs.html) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you

hear the lines `3* , 3 HOST, 3 STATE`, you know that you can include `HOST, STATE`, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The `*` symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line `6.1+ data area`, you must include at least one data area. If you hear the lines `2+, 2 HOST,` and `2 STATE`, you know that you must include `HOST, STATE,` or both. Similar to the `*` symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the `*` symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for the Knowledge Centers. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Site Counsel
2455 South Road

Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, JES2, JES3, and MVS™, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming Interface Information

This book is intended to help the customer to write authorized transaction schedulers for use with APPC/MVS. This book documents General-use Programming Interface and Associated Guidance Information provided by z/OS.

General-use programming interfaces allow the customer to write programs that obtain the services of z/OS.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and Trademark information (www.ibm.com/legal/copytrade.shtml).

Index

U

X

Z

SA23-1398-40