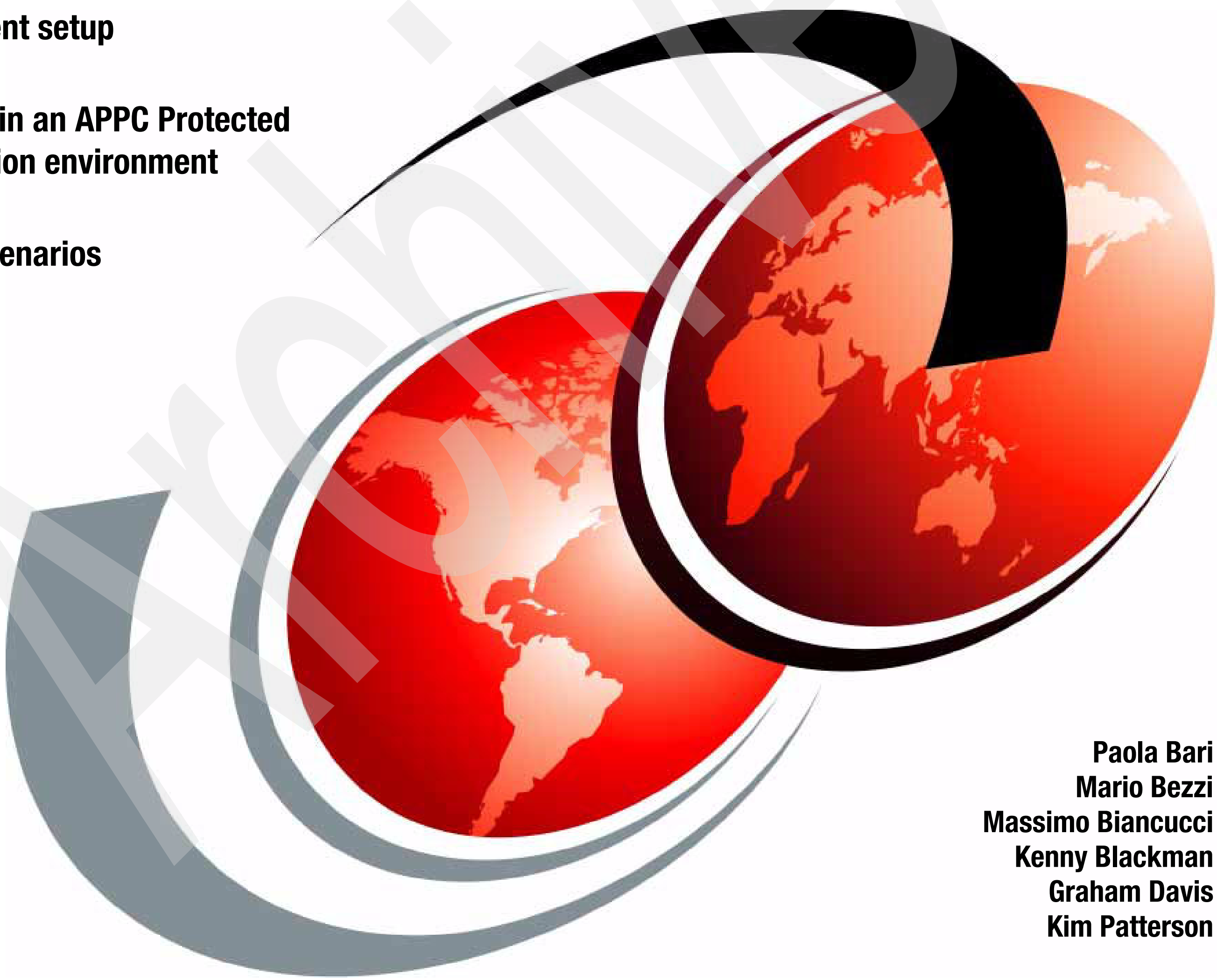
IBM

Implementing and Managing APPC Protected Conversations

- APPC Protected Conversation environment setup
- Operating in an APPC Protected Conversation environment
- Sample scenarios



Paola Bari
Mario Bezzi
Massimo Biancucci
Kenny Blackman
Graham Davis
Kim Patterson

Redbooks

ibm.com/redbooks

International Technical Support Organization

Implementing and Managing APPC Protected Conversations

February 2005

SG24-6486-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (February 2005)

This edition applies to Version 1, Release 6 of z/OS (product number 5694-A01).

Contents

Notices

This information was developed for products and services offered in the U.S.A.

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 give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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.

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

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

This information 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 illustrates 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. 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 IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ™
z/OS®
CICS Connection®
CICS/ESA®
CICS®
DB2 Universal Database™
DB2®
IBM®
IMS™
MQSeries®
MVS™
OS/390®
Parallel Sysplex®
Redbooks™
RACF®
Systems Application Architecture®
SAA®
Tivoli®
VTAM®
WebSphere®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Excel, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Preface

APPC Protected Conversation is a function provided by the operating system to exploiters running on z/OS®. This function improves data integrity in distributed processing environments by enabling participation in the two-phase commit protocol.

This IBM® Redbook provides system programmers with a solid understanding of the APPC Protected Conversation environment. It describes how to upgrade your environment to support protected conversations, how to configure protected conversation exploiters, how to operate in this environment, and how to manage resources. Sample scenarios illustrate how transactions are executed in a protected conversation environment, and how they fail. Design considerations for avoiding failures are also included, as well as a discussion of tools and utilities for monitoring and tuning your APPC environment. Detailed installation definitions are provided for protected conversation exploiters (IMS™, CICS®, and DB2®).

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Paola Bari is an Advisory Programmer in Poughkeepsie, NY. She has 22 years of experience as a systems programmer in OS/390®, z/OS, and Parallel Sysplex®, and several years of experience in WebSphere® MQSeries® and WebSphere Application Server.

Mario Bezzi is a Senior I/T Specialist with IBM Italy. He has 22 years of experience in MVS™ system programming, both as a customer and in various technical support positions. His areas of expertise include system tuning; Parallel Sysplex configuration design, management, tuning and programming; System Logger, RRS, SMF, assembler and C language programming; and system and dasd hardware technology.

Massimo Biancucci is a System Architect working for T-Systems Italia S.p.A. in Italy. He has 16 years of experience in mainframe solutions, both applications and systems. He is a certified Database Administrator on DB2 V7 for z/OS. His areas of expertise include Batch, IMS/TM, IMS/DB, CICS, DB2 and MQSeries environments, user and system application design, application problem determination, and performance optimization.

Kenny Blackman is a certified Senior I/T Specialist in the IMS Advanced Technical Support Group of the IBM Dallas Systems Center. He has 30 years of experience in IMS development, technical support, and as a customer. His primary area of responsibility is connectivity to IMS. He has extensive experience in the following areas: IMS Scheduler, IMS and RRS syncpoint management, Java™ access to IMS, APPC/IMS and TCP/IP connectivity to IMS via IMS Connect. He teaches classes on Connectivity to IMS and recently he has been working with the IMS Java Classes and WebSphere Application Server for z/OS. He is a regular speaker at IBM Technical Conferences and is the ATS-IMS representative to the SHARE User Group in the United States. He holds U.S. Patents in the use of Object-Oriented Technology in an IMS system.

Graham Davis is a z/OS Systems Specialist with IBM Australia, working in the Australian Programming Center in Perth, Western Australia. He has 20 years of experience in Systems Programming and the Systems Management field. His areas of expertise include z/OS, OS/390, CICS TS, CICS/APPC/CPI-C, IBM (Tivoli®) Business Systems Management and OEM Systems Management products. He has worked extensively with international companies and has spent the last few years as a group technical lead designing and developing products for the Australian Programing Center in Assembler, PLX, Cobol and Rexx.

Kim Patterson is a Senior I/T Architect in the Client Technology Center based in Chicago, IL. She has 20 years of experience in z/OS and OS/390 IMS and DB2 Database Administration, application development, and technical support. She is a certified Database Administrator on DB2 V7 for z/OS. She has worked at IBM for eight years.

Thanks to the following people and organizations for their contributions to this project:

T-Systems Italia

Richard Conway
International Technical Support Organization, Poughkeepsie Center

Geoff Meacock
IBM United Kingdom

Kerry Phillips
IBM APC Perth Western Australia

Gianluigi Prada
Banca Intesa S.p.A. Italy

Stephen Warren
IBM Poughkeepsie

Giuseppe Zorzi
X-Sol s.r.l. Italy

Thanks to the following people for their contribution to the review of this redbook:

Thomas Bridges, James Mitchell, Kerry Phillips, Pete Sadler and Stephen Warren for the technical review

Alison Chandler for the editorial review

Ella Buslovich for the graphical review

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

APPC Protected Conversation introduction and theory

This chapter provides a brief introduction to the APPC/MVS function and to Protected Conversation.

It also describes more in detail what a Protected Conversation is, and why and when an installation should use this programming model.

1.1 Introduction to APPC Protected Conversation

Note: If you are not familiar with the APPC terminology, refer to "APPC terminology" on page 5 for details.

APPC/MVS is a VTAM® application that provides full LU 6.2 capability to programs running in z/OS using either APPC verbs or CPI Communications calls. APPC/MVS is implemented as a started task running in a separate address space.

Next to it, APPC/MVS has implemented its own scheduler, ASCH, which controls a pool of address spaces for Transaction Program (TP) scheduling purposes. Any address space can initiate an APPC conversation with a partner TP residing anywhere in the network.

The APPC address space receives and processes inbound (incoming) requests as well as outbound (outgoing) requests. Whenever there is an inbound request, the ASCH is used to schedule the requested TPs. The TP name specified on the incoming attach request is mapped to a so-called TP profile, which contains the necessary information to schedule the TP.

Figure 1-1 shows the components involved in an inbound request.

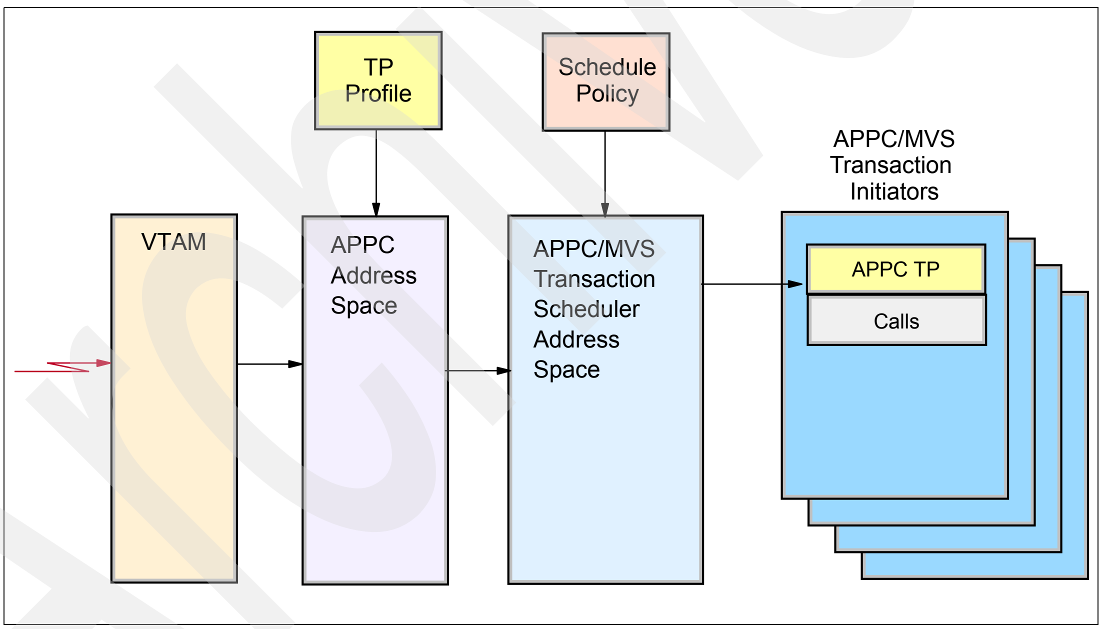


Figure 1-1 Components involved in inbound requests

For outbound requests, the side information data set is (optionally) used to map a symbolic destination name, if supplied, to a partner LU, TP name, and mode name. Figure 1-2 shows the components involved in an outbound request.

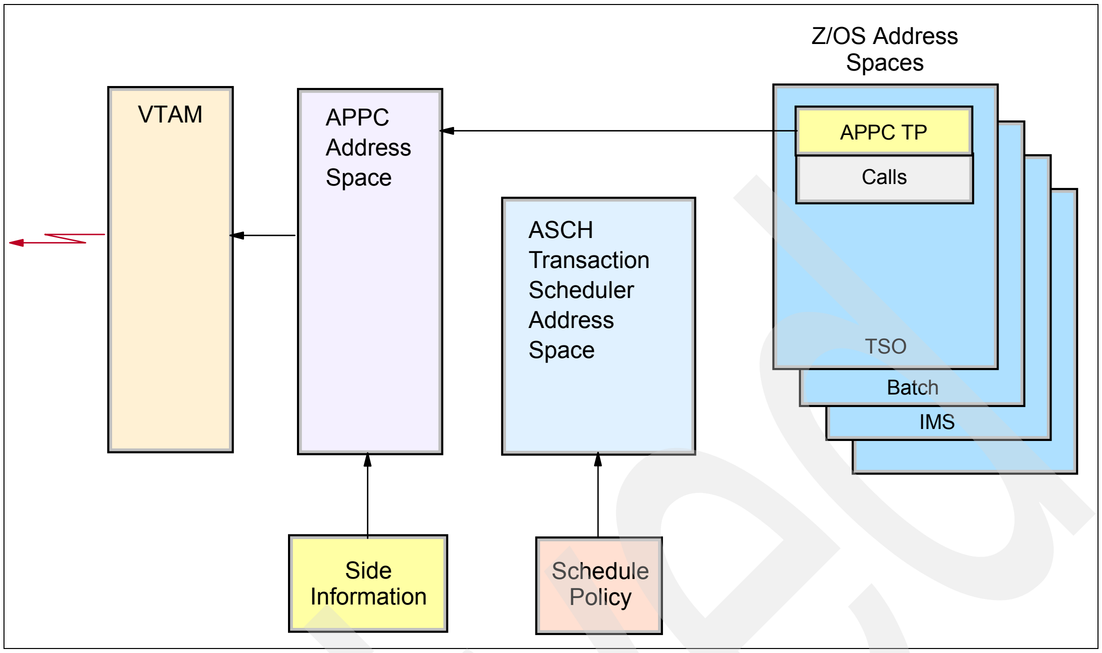


Figure 1-2 Components involved in outbound requests

As an extension to the basic APPC/MVS function, in order to improve data integrity in a distributed processing environment, APPC/MVS provides the possibility to use Protected Conversations where, together with RRS, it participates in the two-phase commit protocol to provide resource recovery for transaction programs.

The two-phase commit protocol is a set of actions that resource managers and a syncpoint manager perform to ensure that a program's updates to distributed resources are coordinated. Through this protocol, a series of resource updates are treated as an atomic action; that is, the updates are either all committed or backed out.

Let's think about a reasonably complex application that verifies a user's identity using digital certificates, links to a WebSphere Application Server that integrates CICS transactions connected to IMS transactions through APPC and DB2 stored procedures spread across four separate enterprises, and sends some multimedia files and a digital receipt back to the user at his Internet screen. And all of this within a single unit of recovery! Sound outlandish? Well, consider that the user is a customer sitting at home using the Internet to book his vacation. Having selected the flights, hotel, and car he wants, he decides to go ahead and book it. This will initiate a transaction to confirm his identify, update the airline's system with his reservation, update the hotel's system to book his room, update the car rental company's system to book his car, debit his bank account with the cost of the package, and finally send him a multi-media file containing information about the package he has selected along with a receipt for the transaction.

The most important thing about this transaction is that all processing must be handled as one atomic transaction. The customer will not be very impressed if his hotel and car are booked and the money is withdrawn from his account, but the flights are not booked! Either *everything* must happen, or *none* of it should. This is the challenge of this new paradigm, and the need to coordinate work across servers and clients is becoming an everyday challenge.

In this redbook, we focus on the APPC portion of the overall transaction process and describe what options are available in building transactions that more accurately reflect your actual business processes and the interactions between your company and those it does business with.

1.1.1 What it is

In z/OS, you can enable APPC/MVS logical units (LUs) to act as resource managers. The resources they manage, or protect, are the conversations established between APPC/MVS transaction programs and their partner TPs. To identify their conversations as protected resources, the TPs allocate the conversations with a synchronization level of syncpoint. When one of the TPs is ready to commit or back out its changes for a particular unit of work, the TP issues either the Commit or Backout callable service to begin a syncpoint operation. During this operation, the local and partner LUs work with system syncpoint managers to coordinate the changes; RRS is the system syncpoint manager for APPC/MVS LUs.

To allow APPC/MVS TPs and their partner TPs to establish protected conversations, your installation must first complete the following steps:

- Set up the APPC/MVS logging environment.
- Set up the APPC/MVS configuration.
- Set up the RRS environment.
- Start APPC and RRS for resource recovery.
- Update existing, or code new, APPC/MVS TPs to allocate protected conversations and request syncpoint services.

Once your installation has met these requirements, APPC/MVS is enabled to support protected conversations, and to participate in resource recovery.

For further information on how to set up the APPC logging environment, refer to 2.2, “APPC log stream” on page 15; for details on setting up the APPC/MVS and VTAM configuration, refer to 2.1, “PARMLIB updates” on page 14; for the RRS environment details, refer to 2.3, “RRS considerations” on page 18.

LU capability and mode name restrictions

APPC/MVS rejects any outbound or inbound requests for protected conversations whenever the partner LU is single-session capable only. Syncpoint-capable LUs accept both inbound and outbound protected conversations, as long as the mode name used for the Allocate call is a value other than SNASVCMG. Defining APPC/MVS LUs as syncpoint capable requires the installation to define the characteristics and resources for LUs through APPL definition statements in VTAMLST, and LUADD statements in APPCPMxx parmlib members. Verify the following elements to define new or alter existing LUs in order to make them syncpoint capable:

- Make sure that each LU’s APPL definition statement contains the SYNCLVL parameter with a value of SYNCPT. This parameter value defines the LU as capable of accepting conversations with any of the following synchronization levels: syncpoint, confirm, or none.
- Make sure that each LU’s APPL definition statement contains the session deallocation ATNLOSS parameter with a value of ALL.
- Check LUADD statements to make sure the values match what you want for specific syncpoint-capable LUs. If you want to restrict the LU to process protected conversations only, for example, check the TPDATA parameter to ensure that the TP profile data set is one containing only TPs that allocate protected conversations.

After you defined the LU with these characteristics, the LUs and, by extension, the schedulers and APPC/MVS servers associated with them, are capable of handling protected conversations.

APPC terminology

The following are common terms used in the APPC/MVS environment. Be sure you understand their meanings before proceeding.

- **Transaction Program (TP):** An application program that uses APPC communication calls is a transaction program, or TP. A TP on one system can communicate with a TP on one or more other systems to access resources on both systems. These systems can be z/OS or any other platform in the network that supports APPC communication. All TPs can be considered a single cooperative processing application that happens to reside on more than one system.
- **Local TP/Partner TP:** Whether a TP is a local TP or a partner TP usually depends on point of view. From the point of view of a z/OS system, TPs residing on the system are local TPs, and TPs on remote systems are partner TPs. However, from the point of view of the remote system, the names are reversed: the TPs that reside on its system are local TPs and the ones on z/OS are the partner TPs. A local TP can initiate communication with one or more partner TPs. The partner might or might not reside on the local system. The TP does not need to know whether the partner TP is on the same system or on a remote system. Other terms for TPs are inbound TP and outbound TP, which convey who establishes the communication. An outbound TP is the one that starts a conversation and an inbound TP is the one that responds. On z/OS, any program that calls APPC/MVS services to start a conversation is considered an outbound TP, while an inbound TP requires special processing by z/OS, such as scheduling and initiation, or processing by an APPC/MVS server.
- **Client TP:** A client transaction program is one that requests the services of an APPC/MVS server.
- **APPC/MVS Server:** An APPC/MVS server is an MVS application program that uses the APPC/MVS Receive_Allocate callable service to receive allocate requests from one or more client TPs. An APPC/MVS server can serve multiple requestors serially or concurrently.
- **Conversation:** The communication between TPs is called a conversation. Like a telephone conversation, one TP calls the other and they *converse*, one TP *talking* at a time, until one TP ends the conversation. The conversation uses predefined communication services that are based on SNA-architected LU 6.2 services called verbs. These verb services are implemented in APPC/MVS as callable services. To start (allocate) a conversation, a TP issues an allocate call that contains specific information, such as the name of the partner TP, the LU in the network where the partner TP resides, and other network and security information. The conversation is established when the partner TP accepts the conversation. After a conversation is established, other calls can transfer and receive data until a TP ends the conversation with a Deallocate call.

  > **Note:** The CPI-Communications protocol requires an Initialize_Conversation (CMINIT) call before an Allocate call.

- **Conversation_ID:** A conversation_ID is an 8-byte token that the Allocate, Initialize_Conversation, Get_Conversation, Accept_Conversation, and Receive_Allocate calls return. APPC requires the conversation_ID to uniquely identify the conversation on subsequent APPC calls.
- **TP_ID:** A TP_ID is a unique 8-byte token that APPC/MVS assigns to each instance of an inbound transaction program. When multiple instances of a TP are running simultaneously under APPC/MVS, they have the same TP name, but each has a unique TP_ID. The TP_ID can be used to trace a specific instance of a TP in the system.

- **Conversation State:** To ensure orderly conversations and prevent both TPs from trying to send or receive data at the same time, APPC enforces conversation states. TPs enter specific conversation states by calling specific APPC services, and the states determine what services the TP may call next. For example, when a local TP allocates a conversation, the local TP is initially in send state; and when the partner TP accepts the conversation, the partner is in receive state. As the need arises, the local TP can call a receive service to enter receive state and put its partner in send state, allowing the partner to send data.
- **Inbound/Outbound Allocate Request:** An outbound allocate request is the initial conversation network flow from the LU to the partner LU as a result of a program attempting to allocate, or start, a conversation with a partner. In technical architecture terminology, this initial flow is called an FMH-5 (Attach) request. An inbound allocate request is simply the partner LU receiving this FMH-5 request.
- **Inbound/Outbound Conversation:** An outbound TP is the one that starts a conversation and an inbound TP is the one that responds. On z/OS, any program that calls APPC/MVS services to start a conversation is considered an outbound TP, while an inbound TP requires special processing by z/OS, such as scheduling and initiation, or processing by an APPC/MVS server.
- **Logical Units (LUs) and LU 6.2:** A logical unit is an SNA addressable unit that manages the exchange of data and acts as an intermediary between an end user and the network. There are different types of logical units. Some LU types support communication between application programs and different kinds of workstations. Other LU types support communication between two programs. LU type 6.2 specifically supports program-to-program communication.
- **Local LU/Partner LU:** Whether an LU is a local LU or a partner LU depends on point of view. From the point of view of a z/OS system, LUs defined to the z/OS system are local LUs and LUs defined to remote systems are partner LUs. However, from the point of view of the remote system, the names are reversed: the LUs that are defined to its system are local LUs and the ones on z/OS are the partner LUs. A partner LU might or might not be on the same system as the local LU. When both LUs are on the same system, the LU through which communication is initiated is the local LU, and the LU through which communication is received is the partner LU. LUs are defined to VTAM on z/OS by APPL statements in VTAMLST. LUs managed by APPC/MVS must also be defined by LUADD statements in APPCPMxx parmlib members.
- **Sessions:** A session is a logical connection that is established or bound between two LUs of the same type. A session acts as a conduit through which data moves between the pair of LUs. A session can support only one conversation at a time, but one session can support many conversations in sequence. Because sessions are reused by multiple conversations, a session is a long-lived connection compared to a conversation. If no session exists when a TP issues an Allocate call to start a conversation, VTAM binds a session between the local LU and the partner LU. After a session is bound, TPs can communicate with each other over the session in a conversation. This sending of data between a local TP and its partner occurs until one TP ends the conversation with a Deallocate call. An installation can define different types of sessions, but sessions are ultimately defined by the LUs they span and by the session characteristics contained in the VTAM logon mode table that is associated with the session. Sessions can span LUs on the same system, LUs on two like systems, and LUs on two unlike systems that are LU 6.2 compatible.
- **Logon Modes:** A logon mode contains the parameters and protocols that determine a session’s characteristics.
- **Contention:** When a TP from each LU in a session simultaneously attempts to start a conversation, the situation that results is called contention. To control which TP can

the contention loser.

1.1.2 Why it is needed

APPC Protected Conversation brings the capability of the two-phase commit protocol where your application is running into a peer-to-peer distributed environment. Let's go through a basic transactional application sample to see why and where APPC Protected Conversation might play its role.

Think about an APPC conversation between two transaction managers, for example IMS1 and IMS2. The flow is pretty simple:

1. A program initiates a conversation with IMS1.
2. IMS1 then establishes a conversation with IMS2, finishes its work and responds to IMS1, then deallocates the conversation. IMS1 in turn finishes its work and responds to the originating program, then deallocates the conversation.
3. The program converses with IMS3.

In this scenario each conversation is treated separately from the others. If a problem occurs in the last conversation, it may cause the work performed by IMS3 and the originating program to be aborted. However, the work performed by IMS1 and IMS2 has completed and been committed. It cannot be reversed.

By using APPC Protected Conversation, the application can gain the capability to provide a distributed commit to ensure that all conversations in the tree are committed or aborted together. Understanding the complexities involved with distributed commit requires knowledge of conversation structure, conversation flow, and commit processing.

Figure 1-3 is a sample of a conversation that occurs between two partners when one initiates the conversation and the second accepts the conversation request. The initiator is in control, that is, in SEND state, and can send to the partner until it wishes to receive some response. When the initiator relinquishes control to the partner and enters RECEIVE state, the partner enters SEND state and has control. The two partners swap control, sending and receiving message data, until the conversation is complete. The partner in SEND state issues the SRRCMIT verb to initiate commit processing before the conversation terminates.

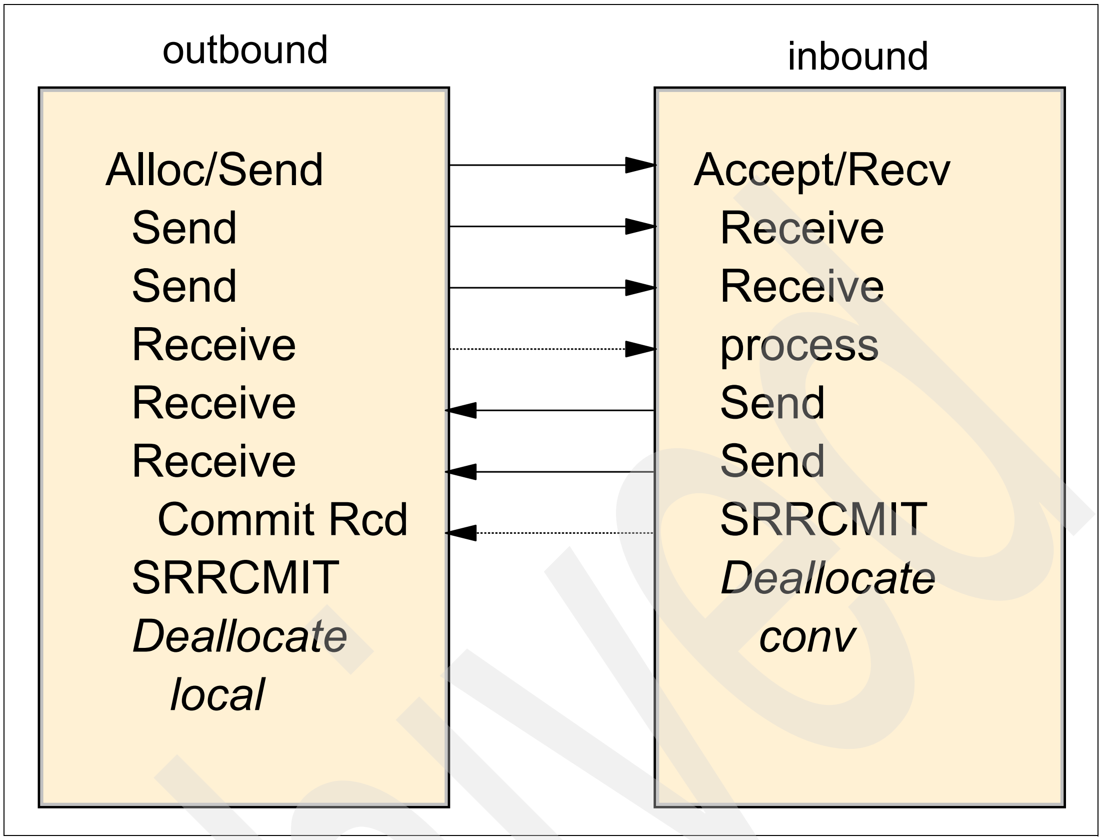


Figure 1-3 Conversation Stages

The commit scope, without distributed commit capability, is local to the resource manager. With SYNCLVL=NONE and SYNCLVL=CONFIRM, when one partner has completed its commit and deallocates the conversation, the other partner is told of the deallocation and issues its own commit. If a problem occurs before the commit completes in the second program there is no way to undo the changes because the backout is not coordinated between the programs, so one will have committed while the other may back out.

The distributed commit in combination with RRS/MVS provides the necessary support for coordinating the commit processes in both partners. The commit manager, RRS, sets a return code for the outstanding receive in the partner telling it to perform commit when the verb is issued. Figure 1-4 and Figure 1-5 show the communication flows between the APPC device and IMS for implicit and explicit conversations with SYNCLVL=SYNCPT.

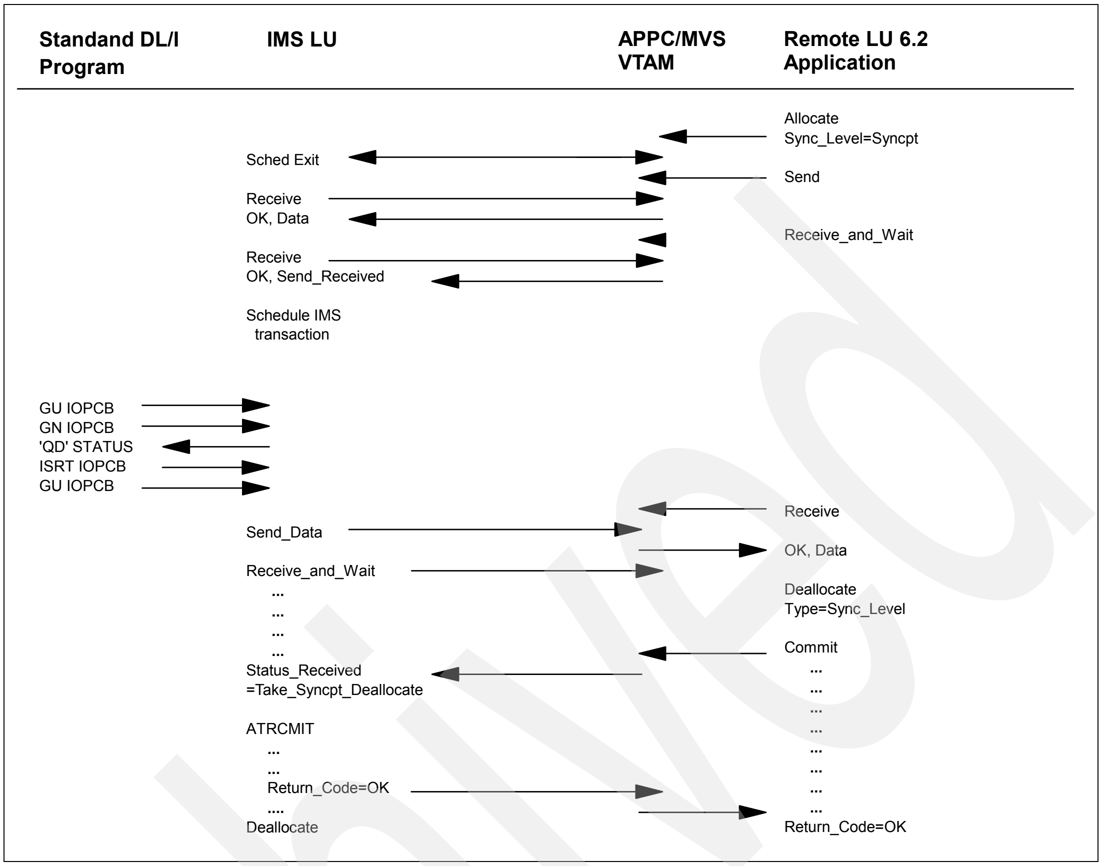


Figure 1-4 Standard DL/I commit scenario for conversation

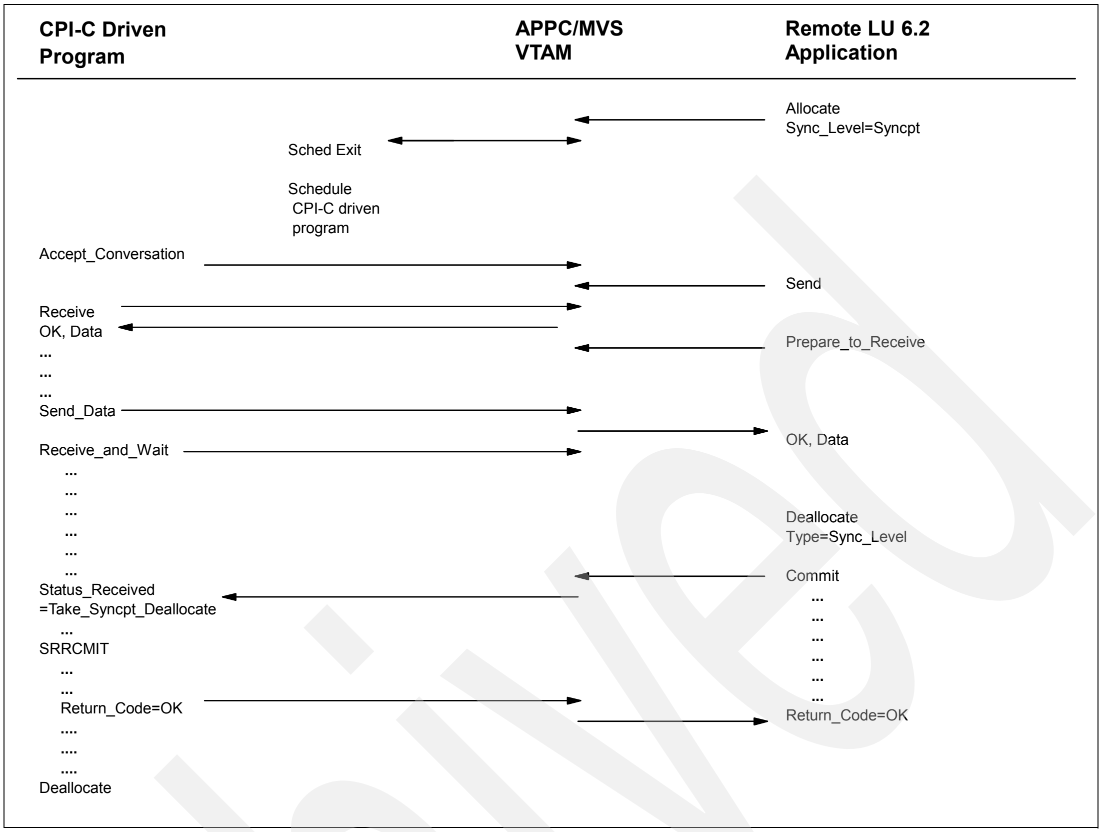


Figure 1-5 CPI-C driven commit scenario for conversation

APPC Protected Conversations are responsible for communications between participating commit managers as shown in Figure 1-6.

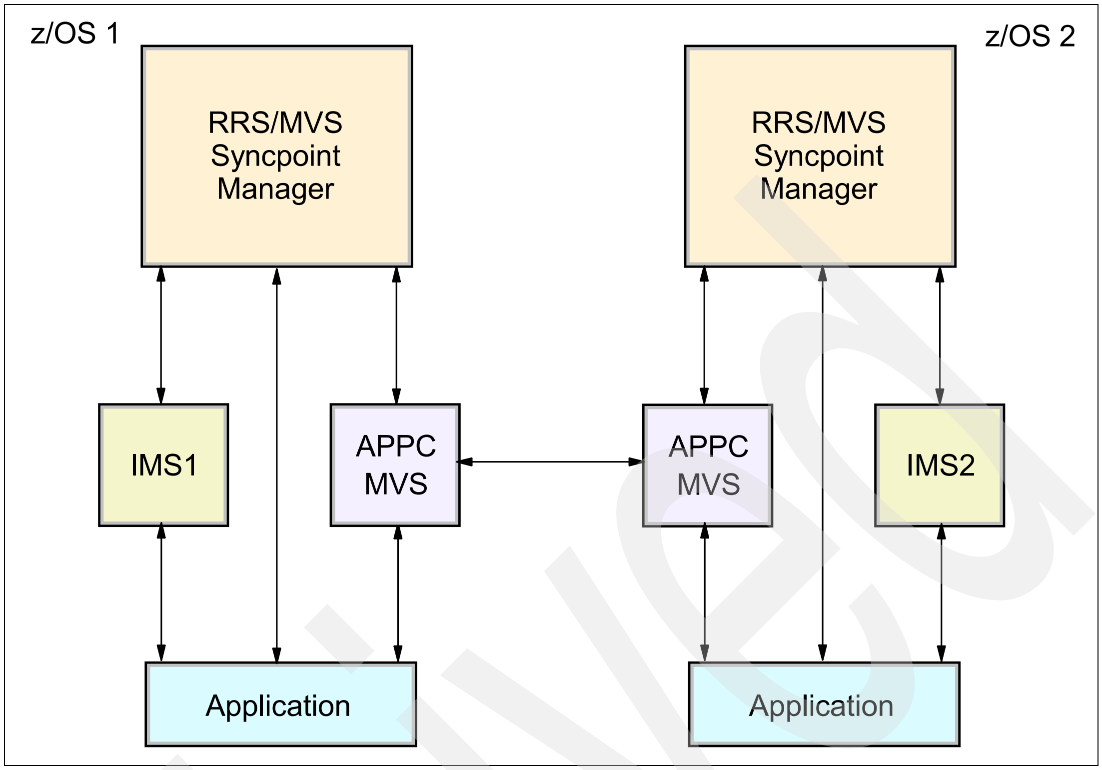


Figure 1-6 Communicating Commit managers

All the resource and commit managers maintain the status of each component of the protected conversation so that they can reestablish consistency after any sort of system failure.

2

Upgrading your configuration to support APPC/MVS Protected Conversations

This chapter describes the tasks you need to perform in order to implement APPC/MVS on your system. The following topics are considered:

- PARMLIB updates
- System Logger
- RRS
- Security
- APPC ISPF panels

2.1 PARMLIB updates

This section describes the changes that are required in the PARMLIB to support an APPC configuration.

2.1.1 Subsystem entries for System Logger and RRS

Ensure the appropriate entries for the System Logger and RRS are in the Subsystem member IEFSSNnn in PARMLIB.

```
SUBSYS SUBNAME(LOGR)                     /* System Logger           */
       INITRTN(IXGSSINT)
SUBSYS SUBNAME(RRS)                      /* Resource Recovery       */
```

Place this statement after the statement that defines the primary subsystem.

2.1.2 Other parmlib entries

Ensure that IEASYSnn has the following statements to ensure your configuration is running in sysplex:

```
COUPLE=nn,                   SYSPLEX(LOCAL)
PLEXCFG=MULTISYSTEM,         SYSPLEX MODE
```

where nn is the COUPLEnn member.

COUPLEnn member

Make sure you defined the SYSPLEX couple data sets and logger couple data sets in the COUPLExx member.

Example 2-1 Contents of SYSx.sysplex.PARMLIB(COUPLE00)

```
COUPLE SYSPLEX(&SYSPLEX.)
        PCOUPLE(SYSx.&SYSPLEX..CDS01,volser1)
        ACOUPLE(SYSx.&SYSPLEX..CDS02,volser2)
        INTERVAL(45)
        OPNOTIFY(48)
        CLEANUP(60)
        MAXMSG(500)
        RETRY(10)
        CLASSLEN(1024)
        CTRACE(CTIXCF00)

 PATHIN  STRNAME(IXCSIG1,IXCSIG2,IXCSIG3,IXCSIG4)
 PATHOUT STRNAME(IXCSIG1,IXCSIG2,IXCSIG3,IXCSIG4)

 DATA   TYPE(CFRM)
        PCOUPLE(SYSx.&SYSPLEX..CFRM01,volser1)       /* PRIMARY COUPLE         */
        ACOUPLE(SYSx.&SYSPLEX..CFRM02,volser2)       /* ALTERNATE  COUPLE      */

 DATA   TYPE(LOGR)
        PCOUPLE(SYSx.&SYSPLEX..LOGR01,volser1)
        ACOUPLE(SYSx.&SYSPLEX..LOGR02,volser2)
```

2.2 APPC log stream

When you enable an APPC Protected Conversation, you must also define to System Logger a log stream for APPC/MVS use. APPC/MVS writes the results of a log name exchange with a partner LU into this log stream.

In fact, to provide resource recovery for protected conversations, APPC/MVS needs to know the names of local and partner LU log streams, and the negotiated syncpoint capabilities for each local/partner LU pair. This information needs to be available and accurate for successful re-synchronization after a failure. To store this information, APPC/MVS uses a log stream.

APPC/MVS does not store conversation-specific information in the log stream. The log stream is used to store data related to unique partner LUs that have protected conversations with APPC/MVS. The log stream contains names of local and partner LU logs, and the negotiated syncpoint capabilities for each local/partner LU pair.

For each protected conversation, information is stored in RRS log streams during the two-phase commit process, but not in APPC/MVS log streams.

APPC/MVS can use both CF-Structure and DASD-only log streams. If you have workload using APPC/MVS protected conversations on multiple images within a sysplex, or if you plan to restart a workload using protected conversations on a different image, then you need to plan for a shared log stream using the CF as the interim media. If you are planning to use APPC/MVS from only one system, only one APPC/MVS will connect to the log stream, in which case you can use a DASD-only log stream if you wish. In this case, only APPC/MVS from this one system processes protected conversations. Other systems in a Parallel Sysplex can use APPC/MVS but they will fail to process any protected conversations since they will not be capable to connect to the log stream and will issue message ATB203I to document the return and reason codes received from the system logger IXGCONN service.

The APPC/MVS log stream is an active log stream, and on an interval basis, APPC/MVS trims unneeded data from the log stream. You should not manage this log stream data using the System Logger parameters RETPD or AUTODELETE, or through any utility.

It is recommended that this log stream is defined with *no* RETPD and AUTODELETE keyword to avoid the deletion of data still in use by APPC/MVS.

There is one IXGWRITE in the log stream for each unique partner LU that uses protected conversations. For example, if you have 50 partner LUs, of which 25 have established protected conversations, you would have at least 25 IXGWRITEs. As you can see, there is not very frequent activity on this log stream. When an LUDEL is performed, the element is removed from the APPC log stream. When this is done, APPC invokes the purge log name affinity processing, which communicates with all partner LUs that have established protected conversations with this LU in the past. If the partner LU gives the OK to purge the affinities, then APPC deletes an entry in the log stream corresponding to that protected partner LU and its corresponding local LU. So, depending on the number of protected partner LUs and the size of the structure, this will determine where the data resides (CF or on DASD).

Criticality/Persistence of data

APPC/MVS keeps information about the conversation partners and their syncpoint in the log stream in order to be able to rebuild the conversation in case of a failure.

For this reason, APPC/MVS data is critical to the installation, and recovery from a loss of the data can be a complex process. To avoid a potential loss of data, we recommend that you use unconditional duplexing with this log stream. Duplexing the log stream should not cause a performance impact to the application since APPC/MVS doesn't update the log stream very frequently.

Losing data in the APPC/MVS log stream means that APPC/MVS loses all knowledge of all partners' log information and syncpoint capabilities.

Log stream sizing

To size the interim media for the APPC/MVS log stream, you can use the following formula:

1. For each local LU that supports SYNCLVL=SYNCPT, calculate the maximum anticipated number of partner LUs that will communicate using protected conversations.
2. After identifying the number of partners for each local LU, add all the values together.
3. Multiply the resulting number by 300 bytes.

This is the *minimum* storage that should be allocated for the interim media for the APPC/MVS log stream. Example 2-2 shows a sample calculation.

Example 2-2 APPC log stream sizing sample

```
There are 8 local LUs on system SC61, 3 of which are defined with SYNCLVL=SYNCPT
   (defined in VTAMLST)
   - SCSIMS8IA communicates with up to 15000 partner LUs, but only 20% of those partners
   communicate using protected conversations.
   - SC61IMAR communicates with up to 5000 partner LUs and all of them could use protected
   conversations.
   - SC61IMSA communicates with 2000 partner LUs, of which about 50% use protected
   conversations.

Based on the above information, the minimum size for the interim storage for the APPC/MVS
log stream is:
(3000+5000+1000) x 300 bytes = 2,700,000 bytes.
```

Log stream definition

APPC/MVS log stream name is pre-defined in z/OS. When a protected conversation is started, APPC connects to the log stream: ATBAPPC.LU.LOGNAMES.

Log stream structure definition in the CFRM policy

If your installation decided to use coupling facility log streams for APPC/MVS, you need to make sure that the corresponding CF structure is defined in the CFRM policy and that the CFRM policy is activated in the sysplex through the SETXCF command. Example 2-3 contains a sample to define the CF structures for the APPC/MVS log stream. You should set SIZE to be roughly twice the size you determined in "Log stream sizing."

This step is not required if you are using DASD-only log stream.

Example 2-3 Sample to define APPC /MVS log stream structure in CFRM policy

```
//APPCSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
   DATA TYPE(CFRM)
   DEFINE STRUCTURE NAME(APPC_STR1) SIZE(8192)
   INITSIZE(4096)
   REBUILDPERCENT(1)
   PREFLIST(CF03, CF06)
```

Log stream definitions in the LOGR policy

Even though you can use DASD-only log streams, most installations configure the APPC/MVS log stream as a CF-Structure log stream. For this reason, the following samples use a CF configuration. Example 2-4 shows the definitions for the APPC/MVS log stream in the LOGR policy followed by the explanation of some of the most significant fields. For a complete description of the fields, refer to *z/OS V1R6.0 MVS Setting Up a Sysplex,* SA22-7625.

Example 2-4 Sample to define APPC /MVS log stream structure in LOGR policy

```
//DEFINE EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
   DATA TYPE (LOGR)

   DEFINE STRUCTURE NAME(APPC_STR1)
      LOGSNUM(1)
      MAXBUFSIZE(65276) AVGBUFSIZE(248)

   DEFINE LOGSTREAM NAME(ATBAPPC.LU.LOGNAMES)
      STRUCTNAME(APPC_STR1)
      LS_DATACLAS(SHARE33)
      STG_DATACLAS(LOGR4K)
      HLQ(IXGLOGR)
      LS_SIZE(1024)
      LOWOFFLOAD(40) HIGHOFFLOAD(80)
      STG_DUPLEX(YES) DUPLEXMODE(UNCOND)
      RETPD(0) AUTODELETE(NO)
```

LOGSNUM The LOGSNUM parameter controls the maximum number of log streams that may be allocated in the associated structure. We recommend specifying a LOGSNUM of 1, meaning that only one log stream will be placed in the structure.

MAXBUFSIZE APPC/MVS requires a buffer size of 65276 bytes. If you use a MAXBUFSIZE value that is less than 65276, APPC/MVS issues message ATB209I and does not allow APPC/MVS LUs to handle any protected conversations until the buffer size is corrected and the LUs restarted.

AVGBUFSIZE The APPC/MVS log stream contains one log block for each of the following:

- Each local/partner LU pair that has established a protected conversation.

- Each LU pair, if any, that has outstanding re synchronization work.

All log blocks are the same size: 248 bytes. Use 248 as the value for the average size of APPC/MVS log blocks.

HIGHOFFLOAD The HIGHOFFLOAD value specifies how full the log stream interim storage should get before an offload process is initiated. We recommend using value not higher than 80%.

LOWOFFLOAD The LOWOFFLOAD value defines the amount of data that is to be retained in the log stream interim storage following an offload process. In the APPC/MVS environment, the LOWOFFLOAD value should be between 40 to 60%.

STG_DUPLEX(YES) and DUPLEXMODE(UNCOND)

If you suffer a loss of data in the APPC/MVS log stream, manual intervention is required to resolve the situation. To avoid this, we recommend defining the APPC/MVS log stream with STG_DUPLEX(YES) and DUPLEXMODE(UNCOND). Because APPC/MVS writes infrequently to its log, the performance impact should be minimal.

AUTODELETE and RETPD

It is very important that AUTODELETE(NO) and RETPD(0) are specified (or use the AUTODELETE and RETPD parameter defaults which are NO and 0, respectively) for the ATBAPPC.LU.LOGNAMES log stream. Using values other than these could result in data being deleted before APPC/MVS is finished with it, or in data being kept for far longer than necessary.

Log stream verification

Once you have defined the APPC log stream, APPC connects to the log stream when the first protected conversation starts. At this point, when you display the status of the log stream using the **D LOGGER, L** command, you should see the status shown in Example 2-5.

Example 2-5 Display APPC log stream

```
D LOGGER,L
INVENTORY INFORMATION BY LOGSTREAM
LOGSTREAM                 STRUCTURE        #CONN  STATUS
---------                 ---------        ------ ------
ATBAPPC.LU.LOGNAMES       APPC_STR1        000002 IN USE
  SYSNAME: SC61
    DUPLEXING: STAGING DATA SET
  SYSNAME: SC62
    DUPLEXING: STAGING DATA SET
```

Log stream security definitions

The APPC/MVS log stream can be protected in the Security Access Facility (SAF) LOGSTRM class. The userid associated with the APPC/MVS address space *must* have UPDATE access to the log stream. The RACF® statements to define the profile and grant access to it are shown in Example 2-6.

Example 2-6 Sample log stream security definitions

```
RDEFINE LOGSTRM ATBAPPC.LU.LOGNAMES UACC(READ)
PERMIT ATBAPPC.LU.LOGNAMES CLASS(LOGSTRM) ID(APPCUR) ACCESS(UPDATE)
SETROPTS CLASSACT(LOGSTRM)
```

The log stream name defined to RACF is the name specified on the NAME parameter of the DEFINE LOGSTREAM statement in the LOGR policy.

2.3 RRS considerations

In order to implement APPC Protected Conversations, you need to set up the RRS environment. This includes the RRS address space, the RRS log streams, and a set of ISPF panels.

Table 2-1 is a summary of the tasks required to set RRS.

Table 2-1 RRS set up tasks

Task	Required
Define the RRS log streams	Required
Establish the priority for the RRS address space	Required
Define RRS as a subsystem	Required
Create procedures to start RRS	Required
Set up automation to restart RRS	Optional
Define RRS security definitions	Optional
Enable RRS ISPF panels	Optional (enables ISPF application to look at RRS log streams)
Set up RRS component trace	Optional

Before you begin these tasks, you need to make certain planning decisions about RRS, such as:

- What log streams your installation will use
- What RRS logging group name to use
- What type of log stream - DASD-only or Coupling Facility
- The size of the log streams

2.3.1 Logging environment

There are five RRS log streams, of which all except the ARCHIVE log stream are *required*. Required means that RRS does not start without being able to connect to these log streams. RRS performs all the logging in the log streams: the resource managers can provide persistent interest data that RRS logs, but RRS does the actual logging.

Table 2-2 summarizes the log streams and their contents.

Table 2-2 RRS Log streams and their content

Log stream type	Log stream name	Content
RRS archive log	ATR.lgname.ARCHIVE	Information about completed UR[a]s.This log is optional. See Note for further details.
RRS main UR state log	ATR.lgname.MAIN.UR	The state of active URs. RRS periodically moves this information into the RRS delayed UR state log when UR completion is delayed.
RRS resource manager data log	ATR.lgname.RM.DATA	Information about the resource managers using RRS services.
RRS delayed UR state log	ATR.lgname.DELAYED.UR	The state of active URs, when UR completion is delayed.

Log stream type	Log stream name	Content
RRS restart log	ATR.lgname.RESTART	Information about incomplete URs needed during restart. This information enables a functioning RRS instance to take over incomplete work left over from an RRS instance that failed.

Note: The ARCHIVE log stream is an optional log stream and it might be useful in situations where you need to keep a history of what happened in coordinating the transactions. If you choose not to use the ARCHIVE log stream, a warning message is issued at RRS startup time about not being able to connect to the log stream, but RRS will continue its initialization process.

a. A UR represents an application program's changes to resources since the last commit or backout or, for the first UR, since the beginning of the application.

RRS logging group name

The RRS images on different systems in a sysplex run independently but share the log streams to keep track of the transactions. An RRS logging group is a group of systems that share an RRS workload. To define a logging group, use the GNAME parameter on the procedure used to start RRS. If you omit the GNAME parameter, the default logging group name is the sysplex name. RRS on each system (there can only be one RRS active on a system) in a sysplex can belong to only one logging group. Within the same logging group, if a system or RRS fails, RRS on a different system can use the shared logs to take over the failed system's outstanding work, thereby enabling quick recovery from system and RRS failures.

Resource Managers do not know about the GNAME. The GNAME is transparent to the RMs. They do not send a call to a specific RRS—unlike DB2, for example, where a caller would identify which DB2 he wants to talk to. On the other hand, all the information in RRS is associated with a particular GNAME, so if you change GNAMEs, all the RRS information from the old GNAME is no longer accessible.

An RM registers with RRS. The installation decides the GNAME name. RMs can provide a logname to RRS and RRS will provide a logname to an RM, but the RRS logname is basically a timestamp. No GNAME is involved.

Some advantages of using multiple RRS logging groups are:

- You can use different log groups to subdivide the transaction work in a sysplex. For example, you can use separate logging groups for test systems and production systems.
- You can restart RRS with a different log group name to cause a cold restart and keep the data in the old logs for debugging and data recovery purposes. (The RRS panels allow you to browse any set of logs, you just need to specify the "gname" on the panel. This option is only meaningful for recovery purposes).

Log stream characteristics

RRS supports both Coupling Facility log streams and DASD-only log streams. A DASD-only log stream has a single-system scope and thus cannot be used in a multi-system sysplex environment except in particular circumstances. For example, you might have an instance of RRS on a test image that uses its own logging group that is not shared with any other system in the sysplex. In this particular configuration, RRS can use DASD-only log streams. Usually, either for restartability issues or because of the workload configuration, RRS is configured to use Coupling Facility log streams.

All instances of RRS in the same logging group must have access to the Coupling Facility structures and log stream data sets used by the RRS log streams for that logging group. This allows other RRS instances in a sysplex to access data in the event of a failure with an RRS instance or system. This is required to allow resource managers to be restarted on different systems within a sysplex.

Log stream structure sizing

Table 2-3 provides initial considerations on the amount of storage required for the RRS log streams. These recommendations should result in reasonably efficient usage of Coupling Facility storage while minimizing the likelihood that you will have to redefine the structures due to the variations in your workload. However, the exact amount of storage you need for each log stream depends on the installation's RRS workload. SMF88 data can be used to determine if the structure sizes require any adjustments.

Prior to starting RRS for the first time, you can get an estimate of the required structure sizes using the CF Sizer tool, available on the Web at:

http://www.ibm.com/servers/eserver/zseries/pso

We used the values shown in Table 2-3 as input to the Sizer tool.

Table 2-3 Sample I/O activity for the RRS log streams

Log Stream	Writes per sec	Storage requirements
RM.DATA	2	Low, if few resource managers; Medium, if many resource managers
RESTART	10	Medium
MAIN.UR	50	High
DELAYED.UR	10	High
ARCHIVE	50	Low

It is still a good practice, once you have run some workload, to re-evaluate the log streams' allocation sizes through the SMF type 88 records.

We suggest mapping each Coupling Facility log stream to a unique CF structure. The CF structures must be defined in the CFRM policy. Log streams are then mapped to those structures through the LOGR policy.

If your installation has a constraint on the number of Coupling Facility structures in your CFRM policy, you can group multiple RRS log streams in a single Coupling Facility structure. In that case, the following grouping is suggested:

- Place the RM.DATA and RESTART log streams in one structure.
- Place the MAIN.UR, DELAYED.UR and ARCHIVE (if used) into another structure.

RRS log streams are "active" log streams, with the exception of the ARCHIVE. What we mean by active is that RRS manages the content of its log streams and keeps it up to date with the current workload running on the system. As a result, these log streams should not require a lot of storage in the interim storage medium and should not generate a lot of offload operations. As opposed to the other log streams, the ARCHIVE is a "funnel-type" log stream, containing a record for each completed transaction. A funnel-type log stream is one where RRS just writes to the log stream and never re-uses or deletes the ARCHIVE log records. For this reason, you should expect this log stream to use offload data sets. The volume of data in the log stream can be managed through a combination of AUTODELETE(YES) and RETPD value set to the number of days you want to keep this data in your installation.

RRS log streams definitions

RRS supports both Coupling Facility log streams and DASD-only log streams. The following list describes the steps required to set up RRS log streams. If you are using DASD-only log streams, you can skip steps 2 and 3.

The following tasks need to be completed:

1. Verify DFSMS definitions required for staging and offload data sets.
2. Define the Coupling Facility structures to the CFRM policy and activate the new policy once they have been updated.
3. Define the structures to the System Logger policy.
4. Define the log streams to the System Logger policy.

Verify DFSMS definitions required for RRS

To ensure successful operation, the data sets used by System Logger must be allocated with the correct attributes. To do this, you need to use the SMS constructs:

- Determine the naming conventions for the log stream data sets. The default data set name for offload and staging is IXGLOGR.ATR.*gname.logstreamname* where *gname* (the logging group name) defaults to the sysplex name. You can specify the high level qualifier for the data sets using the HLQ or EHLQ parameter on the log stream definition in the System Logger policy.
- Ensure that the correct SMS classes are assigned to the offload and staging data sets, either by specifying the SMS class names in the System Logger policy, or by coding appropriate SMS ACS routines. In particular, ensure that SHAREOPTIONS(3,3) are specified in the Data Class to avoid data loss conditions or lockout due to enqueueing.

Example 2-7 Display of the DCRRSLGR (RRS Logs) SMS data class

```
DATA CLASS DISPLAY
Command ===>

CDS Name  . . . : ACTIVE
Data Class Name : LOGR4K

Description : DATA CLASS FOR RRS LOGGER DATA SETS CISIZE 4096

Recfm  . . . . . . . . . :
Lrecl  . . . . . . . . . :
Space Avgrec . . . . . . :
      Avg Value  . . . . :
      Primary  . . . . . :
      Secondary  . . . . :
      Directory  . . . . :
Retpd Or Expdt . . . . . :
Volume Count . . . . . . : 1
  Add'l Volume Amount  . :
Data Set Name Type  . . . :
  If Extended . . . . . . :
  Extended Addressability : NO
  Record Access Bias  . . :
Space Constraint Relief . : NO
  Reduce Space Up To (%)  :
  Dynamic Volume Count  . :
```

```
Compaction  . . . . . . . :
Spanned / Nonspanned  . . :
Media Interchange
  Media Type  . . . . . . :
  Recording Technology  . :
  Performance Scaling . . :
Block Size Limit  . . . . :
Recorg . . . . . . . . . : LS
Keylen . . . . . . . . . :
Keyoff . . . . . . . . . :
Imbed  . . . . . . . . . :
Replicate  . . . . . . . :
CIsize Data  . . . . . . : 4096
% Freespace CI . . . . . :
            CA . . . . . :
Shareoptions Xregion . . : 3
             Xsystem . . : 3

Use UP/DOWN Command to View other Panels;
Use HELP Command for Help; Use END Command to Exit.
```

Define the Coupling Facility structures in the CFRM policy

This task is only required if you are defining Coupling Facility base log streams.

If this is the case, then each log stream needs to be mapped to a Coupling Facility structure defined in the CFRM policy. Log streams are then mapped to these structures in the LOGR policy.

Example 2-8 shows the definitions we used in our configuration. There is a Coupling Facility structure per each log stream.

Example 2-8 Sample for RRS log streams definitions in CFRM policy

```
//MAINSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
   DATA TYPE(CFRM)

   DEFINE STRUCTURE NAME(RRS_ARCHIVE_1) SIZE(72000)
   INITSIZE(16000)
   REBUILDPERCENT(5)
   PREFLIST(CF06, CF03)

   DEFINE STRUCTURE NAME(RRS_DELAYEDUR_1) SIZE(16000)
   INITSIZE(8000)
   REBUILDPERCENT(5)
   PREFLIST(CF03, CF06)

   DEFINE STRUCTURE NAME(RRS_MAINUR_1) SIZE(48000)
   INITSIZE(8000)
   REBUILDPERCENT(5)
   PREFLIST(CF06, CF03)

   DEFINE STRUCTURE NAME(RRS_RESTART_1) SIZE(48000)
   INITSIZE(8000)
   REBUILDPERCENT(5)
   PREFLIST(CF06, CF03)
```

```
   DEFINE STRUCTURE NAME(RRS_RMDATA_1) SIZE(16000)
    INITSIZE(8000)
    REBUILDPERCENT(5)
    PREFLIST(CF03, CF06)
```

Define the CF log stream in the System Logger policy

Once the structures are defined in the CFRM policy, the next step is to define the log streams in the System Logger policy. If your installation is planning to use DASD-only log streams, skip this section and refer to "Define the DASD-only log stream in the System Logger policy" on page 27. Example 2-9 shows a sample we used in our configuration to define the RRS log streams and map them to the CF structure within the System Logger policy.

Example 2-9 Sample to define CF based RRS log streams in the LOGR policy

```
//DEFSTR  EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
   DATA TYPE(LOGR)

   DEFINE STRUCTURE NAME(RRS_ARCHIVE_1) LOGSNUM(1)
      MAXBUFSIZE(64000) AVGBUFSIZE(262)

   DEFINE STRUCTURE NAME(RRS_DELAYEDUR_1) LOGSNUM(1)
      MAXBUFSIZE(64000) AVGBUFSIZE(158)

   DEFINE STRUCTURE NAME(RRS_MAINUR_1) LOGSNUM(1)
      MAXBUFSIZE(64000) AVGBUFSIZE(158)

   DEFINE STRUCTURE NAME(RRS_RESTART_1) LOGSNUM(1)
      MAXBUFSIZE(64000) AVGBUFSIZE(158)

   DEFINE STRUCTURE NAME(RRS_RMDATA_1) LOGSNUM(1)
      MAXBUFSIZE(1024) AVGBUFSIZE(252)

   DEFINE LOGSTREAM NAME(ATR.WTSCPLX1.ARCHIVE) STRUCTNAME(RRS_ARCHIVE_1)
      HLQ(IXGLOGR)
      LS_SIZE(1024) LS_DATACLAS(SHARE33)
      LOWOFFLOAD(0) HIGHOFFLOAD(80)
      STG_DUPLEX(NO)
      RETPD(7) AUTODELETE(YES)

   DEFINE LOGSTREAM NAME(ATR.WTSCPLX1.DELAYED.UR) STRUCTNAME(RRS_DELAYEDUR_1)
      HLQ(IXGLOGR)
      LS_DATACLAS(SHARE33) LS_SIZE(1024)
      STG_SIZE(960)
      LOWOFFLOAD(60) HIGHOFFLOAD(80)
      STG_DUPLEX(YES) DUPLEXMODE(COND)

   DEFINE LOGSTREAM NAME(ATR.WTSCPLX1.MAIN.UR) STRUCTNAME(RRS_MAINUR_1)
      HLQ(IXGLOGR)
      LS_DATACLAS(SHARE33)  LS_SIZE(1024)
      STG_SIZE(1024)
      LOWOFFLOAD(60) HIGHOFFLOAD(80)
      STG_DUPLEX(YES) DUPLEXMODE(COND)

      DEFINE LOGSTREAM NAME(ATR.WTSCPLX1.RESTART) STRUCTNAME(RRS_RESTART_1)
      HLQ(IXGLOGR)
      LS_DATACLAS(SHARE33) LS_SIZE(1024)
      STG_SIZE(1024)
```

```
LOWOFFLOAD(60) HIGHOFFLOAD(80)
STG_DUPLEX(YES) DUPLEXMODE(COND)

DEFINE LOGSTREAM NAME(ATR.WTSCPLX1.RM.DATA) STRUCTNAME(RRS_RMDATA_1)
   HLQ(IXGLOGR)
   LS_DATACLAS(SHARE33) LS_SIZE(1024)
   STG_SIZE(1024)
   LOWOFFLOAD(60) HIGHOFFLOAD(80)
   STG_DUPLEX(YES) DUPLEXMODE(UNCOND)
```

Following is an explanation of some of the fields that might affect the RRS logging environment. For a complete description of all the fields and their values, refer to *z/OS V1R6.0 MVS Setting Up a Sysplex,* SA22-7625.

MAXBUFSIZE/AVGBUFSIZE

MAXBUFSIZE in conjunction with AVGBUFSIZE is used to determine the CF structure ENTRY/ELEMENT ratio. When data is written to the CF, it's written in increments equal to ELEMENT size. A MAXBUFSIZE greater than 65276 gives an element size of 512; a MAXBUFSIZE equal to or less than 65276 results in an element size of 256.

We recommend running for a time with initial AVGBUFSIZE values as suggested in Table 2-4. You can then alter your log stream definition to adjust the AVGBUFSIZE value to match the System Logger recommended value suggested in the IXCMIAPU report.

Table 2-4 Initial AVGBUFSIZE values for RRS log streams

Log stream	Recommended starting AVGBUFSIZE
ARCHIVE	262
DELAYED.UR	158
MAIN.UR	158
RESTART	158
RM.DATA	252

It does not matter if the defined AVGBUFSIZE does not exactly match the average buffer size as reported by IXCMIAPU because System Logger dynamically adjusts the Entry/Element ratio. System logger will adjust the ratio, avoiding potential problems, especially if you don't share the same structure between multiple log streams, each of which could have a different average buffer size.

AUTODELETE and RETPD

AUTODELETE and RETPD can have a disastrous effect if specified other than AUTODELETE(NO) and RETPD(0) for all the RRS log streams except the ARCHIVE. With AUTODELETE(YES) and RETPD>0, even though RRS will attempt to delete unnecessary log entries, all data will be off-loaded to the offload data sets and held for the number of days specified for RETPD. AUTODELETE(YES) allows the System Logger (rather than RRS) to decide when to delete the data. When a new offload data set is allocated and AUTODELETE(YES) is specified, the System Logger will delete the data on the old offload data set that has passed the retention period. Since data in the MAIN.UR log stream is managed by RRS, it will be better to let RRS manage the life of the records on this log stream so there will not be the risk that RRS will need records that have been deleted by System Logger because of the AUTODELETE option.

For the ARCHIVE log stream, RRS never uses information written on the Archive log; the information is intended for the installation to use if a catastrophic problem occurs. You must use retention period and autodelete support to delete old entries; specify these value large enough to manage this log stream to a reasonable size and to provide enough coverage in time to recover any potential situation.

HIGHOFFLOAD

The HIGHOFFLOAD parameter is used to determine when the space dedicated to the log stream in the Coupling Facility is filling up and an offload needs to be initiated to regain available space. HIGHOFFLOAD should be set at 80% for all the RRS log streams, at least initially, and then use the SMF88 report to evaluate if this value need same tuning.

HLQ

Specifies the up to 8-byte high-level qualifier for both the log stream data set name and the staging data set name. HLQ must be 8 alphanumeric or national ($,#,or @) characters, padded on the right with blanks if necessary. The first character must be an alphabetic or national character.

LOWOFFLOAD

The LOWOFFLOAD value defines the amount of data which may be retained in the log stream interim storage following an offload process. In the RRS environment, the LOWOFFLOAD value should be high enough to retain the data required for backout of the UR, but low enough to keep the number of offloads to a minimum.

LOWOFFLOAD should be set between 20% and 60% for all the RRS log streams as described in the examples and 0% for the ARCHIVE.

LS_SIZE

LS_SIZE defines the allocation size for the *offload* data sets. It should be specified large enough to contain several offloads, possibly a day's worth. ALL RRS log streams except ARCHIVE should only offload a minimal amount of data.

STG_SIZE

For a Coupling Facility log stream, STG_SIZE defines the size of the staging data set to be allocated if STG_DUPLEX(YES) and DUPLEXMODE are specified. If STG_DUPLEX(YES) and DUPLEXMODE(UNCOND) are specified the data in the Coupling Facility log stream is always duplexed to the staging data set. If STG_DUPLEX(YES) and DUPLEXMODE(COND) are specified, the data in the Coupling Facility log stream is duplexed to the staging data set only if the CF becomes volatile or failure dependent.

The size of the staging data set (STG_SIZE) must be large enough to hold as much data as the log stream storage in the Coupling Facility.

Data is written to the staging data set in 4096 byte increments, regardless of the buffer size.

STG_DUPLEX

STG_DUPLEX(YES) with DUPLEXMODE(COND) means that if the CF becomes volatile, or resides in the same failure domain as the System Logger, the log stream data is duplexed to the staging data set; otherwise it is duplexed to buffers in the System Logger dataspace. A CF is in the same failure domain when the Coupling Facility LPAR and the LPAR running this z/OS reside in the same physical hardware box, central processing complex (CPC). Duplexing to the staging data set means the cost of an I/O will be incurred for each write.

Define the DASD-only log stream in the System Logger policy

Example 2-10 shows the definitions for the RRS log streams when defined as DASD-only log streams. In this case you do not need any prior definition in the CFRM policy.

Example 2-10 DASD-only definitions for RRS log streams

```
//DEFSTR  EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
   DATA TYPE(LOGR)

   DEFINE LOGSTREAM NAME(ATR.WTSCPLX1.ARCHIVE)
      HLQ(IXGLOGR)
      LS_SIZE(1024) LS_DATACLAS(SHARE33)
      LOWOFFLOAD(0) HIGHOFFLOAD(80)
      STG_SIZE(2000)
      RETPD(7) AUTODELETE(YES)

   DEFINE LOGSTREAM NAME(ATR.WTSCPLX1.DELAYED.UR)
      HLQ(IXGLOGR)
      LS_DATACLAS(SHARE33) LS_SIZE(1024)
      STG_SIZE(960) DASDONLY(YES)
      LOWOFFLOAD(60) HIGHOFFLOAD(80)

   DEFINE LOGSTREAM NAME(ATR.WTSCPLX1.MAIN.UR)
      HLQ(IXGLOGR)
      LS_DATACLAS(SHARE33)  LS_SIZE(1024)
      STG_SIZE(1024) DASDONLY(YES)
      LOWOFFLOAD(60) HIGHOFFLOAD(80)


   DEFINE LOGSTREAM NAME(ATR.WTSCPLX1.RESTART) STRUCTNAME(RRS_RESTART_1)
      HLQ(IXGLOGR)
      LS_DATACLAS(SHARE33) LS_SIZE(1024)
      STG_SIZE(1024) DASDONLY(YES)
      LOWOFFLOAD(60) HIGHOFFLOAD(80)

   DEFINE LOGSTREAM NAME(ATR.WTSCPLX1.RM.DATA) STRUCTNAME(RRS_RMDATA_1)
      HLQ(IXGLOGR)
      LS_DATACLAS(SHARE33) LS_SIZE(1024)
      STG_SIZE(1024) DASDONLY(YES)
      LOWOFFLOAD(60) HIGHOFFLOAD(80)
```

Following is a description of the most significant parameter for the RRS log stream allocation in a DASD-only configuration. For a complete description of the parameters, refer to *z/OS V1R6.0 MVS Setting Up a Sysplex,* SA22-7625.

DASDONLY

Specifies whether the log stream being defined is a coupling facility or a DASD-only log stream.

MAXBUFSIZE

MAXBUFSIZE may be specified for a DASDONLY log stream. It defines the largest block that can be written to the log stream.   The default value of 65532 should be used in an RRS environment.

2.3.2 WLM definitions

Use the Workload Manager (WLM) policy to control the RRS priority. The RRS priority needs to be equal to or higher than the dispatching priority of its resource managers. You can use the SYSSTC service class for RRS address space to achieve a higher dispatching priority.

2.3.3 RRS procedure

ATRRRS is the name of the cataloged procedure that IBM supplies in SYS1.SAMPLIB. Copy SYS1.SAMPLIB(ATRRRS) to SYS1.PROCLIB(RRS). The member name RRS specified here can be replaced with any other member name, as long as it matches the subsystem name specified in the IEFSSNxx PARMLIB member.

2.3.4 RRS ISPF panels

RRS provides ISPF panels to allow an installation to work with RRS. The panels provide a way for you to troubleshoot resource recovery problems. Before you can use the panels, however, you must set up access authorization, allocate the libraries containing the panels, and add the RRS application to the ISPF primary option menu.

Reference:

```
z/OS MVS Programming: Resource Recovery Document Number SA22-7616-03

A.0 Appendix A.  Using RRS Panels

To display resource manager information, select option 2 on the panel RRS primary
options and press ENTER:

+----------------------------------------------------------------------------------+
|                                                                                  |
|                                    RRS                                           |
|    Option ===> ________________________________________________________________ |
|                                                                                  |
|    Select an option and press ENTER:                                             |
|                                                                                  |
|     1 Browse an RRS log stream                                                   |
|     2 Display/Update RRS related Resource Manager information                    |
|     3 Display/Update RRS Unit of Recovery information                            |
|     4 Display/Update RRS related Work Manager information                        |
|     5 Display/Update RRS UR selection criteria profiles                          |
|     6 Display RRS-related system information                                     |
```

To install the panels, follow these steps:

1. Update your logon procedure. Concatenate the following RRS libraries.

   Note: If you have IPCS or WLM installed in your logon procedure then the RRS ISPF environment is installed by default.

   In your SYSPROC concatenation:

   ```
   //       DD DSN=SYS1.SBLSCLI0,DISP=SHR
   ```

In your ISPMLIB concatenation:

```
//        DD DSN=SYS1.SBLSMSGO,DISP=SHR
```

In your ISPPLIB concatenation:

```
//        DD DSN=SYS1.SBLSPNLO,DISP=SHR
```

In your ISPTLIB concatenation:

```
//        DD DSN=SYS1.SBLSTBLO,DISP=SHR
```

In your ISPSLIB concatenation:

```
//        DD DSN=SYS1.SBLSKELO,DISP=SHR
```

2. Add the following lines to your primary options menu, member ISR@PRIM within your ISPPLIB concatenation in your logon procedure.
   a. Add to your menu options definitions:

   ```
   %RRS +-%Resource Recovery Svcs Panels
   ```

   b. Add to your application list within the ISR@PRIM menu definition:

   ```
   RRS,'PANEL(ATRFPCMN) NEWAPPL(RRSP)'
   ```

RRS ISPF Panel example

The following is an example of what should appear once RRS has been successfully activated.

Example 2-11 RRS Panel example

Select **Option 2** from the RRS Panel;

```
2 Display/Update RRS related Resource Manager information

Command ===>                                                Scroll ===> PAGE

Commands: v-View Details u-View URs r-Remove Interest

S   RM Name                          State            System   Logging Group
    ATB.USIBMSC.IMSHAPPC.IBM         Run              SC61     WTSCPLX1
    ATB.USIBMSC.IMSIAPPC.IBM         Run              SC61     WTSCPLX1
    DSN.RRSATF.IBM.DB2               Run              SC61     WTSCPLX1
    DSN.RRSATF.IBM.DBC               Reset            SC61     WTSCPLX1
    IMS.IMSH____V081.STL.SANJOSE.IBM Run              SC61     WTSCPLX1
    IMS.IMSI____V081.STL.SANJOSE.IBM Run              SC61     WTSCPLX1
******************************* Bottom of data ********************************
```

It is recommended that you set up the RRS panels. In the event of a failure you may need to use panel information to clean up outstanding transactions. There is no other mechanism for determining the state of the various resource managers. If you have a problem running RRS, you will need to use the RRS panels for problem determination within your sysplex.

2.3.5 SAF authorization

In your installation, you can configure RRS to allow a user to manage all the RRS images in the sysplex from a single image. Access to RRS system management functions is controlled by the following RACF resource.

To control RRS access across a sysplex, RRS uses the MVSADMIN.RRS.COMMANDS.*gname.sysname* resource in the FACILITY class, where *gname* is the logging group name, and *sysname* is the system name.

If you are running RRS on a single system, RRS can use either the MVSADMIN.RRS.COMMANDS.*gname.sysname* resource or the MVSADMIN.RRS.COMMANDS resource in the FACILITY class to control access to RRS system management functions on the current system.

2.3.6 Component trace

You can use the CTMEM parameter on the RRS procedure to specify the CTnRRSxx parmlib member that RRS component trace is to use. IBM recommends that you run with the minimal trace set as described here. IBM will need this information to debug RRS and resource manager problems.

```
TRACEOPTS ON OPTIONS('EVENTS(EXITS,URSERVS,RESTART)') BUFSIZE(4M)
```

2.4 Security considerations

Due to the cooperative nature of APPC communication, you should carefully consider some aspects of security. The security level you would like to establish depends on the importance of the data the application handles. For certain applications you would like much more control than for others, and APPC, within the application itself, allows us all the control we need. Mainly, there are three levels of control:

- Application level
- Network definition level (Can override a lower application security level.)
- Security Server level (Can override both lower and higher levels of network security.)

2.4.1 Application level

At the application level you decide if the security parameters for the user running the transaction initiating the conversation (outbound) have to be transmitted to the partner; you control this within the SECURITY_TYPE PARAMETER on the ATBALLC API. Table 2-5 shows permitted values for SECURITY_TYPE parameter on the ATBALLC API call.

Table 2-5 Permitted security level for APPC TP

Value	Meaning
ATB_SECURITY_NONE	The outbound TP passes no security information.
ATB_SECURITY_SAME	The outbound TP assumes that the inbound TP has the same security level and provides one of the following, if available: ► A user ID ► A security profile name, which APPC treats as a group ID ► An already verified indicator
ATB_SECURITY_PROGRAM	The outbound TP specifies a user ID and a password.

For further information, refer to “Determining the Application's Security Type” in *z/OS V1R4.0 MVS Planning: APPC/MVS Management,* SA22-7599.

2.4.2 Network level

At the network level, you decide the security level of the LUs within the SECACPT parameter on the APPL macro definition. Table 2-6 shows permitted values.

Table 2-6 Permitted values for secacpt parameter of appl macro

Value	Meaning
NONE	The local LU does not support conversation requests containing access security subfields (ignore information if specified).
CONV	Local LU supports conversation requests containing access security subfields.
ALREADYV	Local LU supports conversation requests containing access security subfields. The LU also accepts the already-verified indications that it receives in conversation requests from partner LUs.
PERSISTV	Local LU supports conversation requests containing access security subfields. The LU also accepts the persistent verification indications that it receives in conversation requests.
AVPV	Local LU supports conversation requests containing access security subfields. The LU also accepts the already-verified indications and the persistent verification indications that it receives in conversation requests.

For further information, refer to *z/OS V1R4.0 CS: SNA Resource Definition Reference,* SC31-8778.

APPC security has to be coordinated between partners. For instance, if your application requests a security level of ATB_SECURITY_SAME and the remote LU is defined with SECACPT=NONE, the security information sent will be ignored and this may not be what you want.

2.4.3 Security Server level

At Security Server level you control the security of LUs in several ways using the APPCLU class. This definition can override the Network SECACPT parameter by specifying a CONVSEC value in the SESSKEY parameter. The CONVSEC values reflect names and meanings of SECACPT parameters shown in Table 2-6. For further information, refer to “Defining Conversation Security Levels that Sessions Allow” in *z/OS V1R4.0 MVS Planning: APPC/MVS Management,* SA22-7599.

You have to plan your security definitions with all the components involved in the communication process. Usually applications use a level of ATB_SECURITY_SAME and leave the Security Server to control the access. Use specific network definitions to avoid eventual forgotten LUs, with SECACPT of CONV or ALREADYV. In this scenario, the userid of the outbound TP, if available, is passed to the inbound TP and, without verifying the password, the inbound side Security Server can control if the userid is authorized to execute the TP. Depending on inbound and outbound TP environments, you have to consider which is the available userid at the inbound side. Table 2-7 summarizes various combinations, assuming that your session is at a security level of same. For further information, refer to “What is APPC security=SAME” in *z/OS V1R2.0-V1R4.0 CS: APPC Application Suite Administration,* SC31-8835.

Table 2-7 Userid granularity in a session with a security level of same

Outbound	Inbound	Settings	Userid sent	Pro	Con
IMS	IMS	None	MSG region userid	Only one definition to permit TP to every single user. Usually the security check is at outbound TP level.	No single userid granularity
IMS	CICS	None	MSG region userid	Only one definition to permit TP to every single user. Usually the security check is at outbound TP level.	No single userid granularity
CICS	IMS	Only single userid	Signed userid for trx	Only one definition to permit TP to every single user. Usually the security check is at outbound TP level.	No single userid granularity
IMS	IMS	Exit DFSBSEX0 to build in pst ACEE for trx userid	Signed userid for trx	Granularity of access control to the single userid.	Number of definitions to permit TP to users
IMS	CICS	Exit DFSBSEX0 to build in pst ACEE for trx userid	Signed userid for trx	Granularity of access control to the single userid.	Number of definitions to permit TP to users
CICS	IMS	Every userid	Signed userid for trx	Granularity of access control to the single userid.	Number of definitions to permit TP to users

When your session is at a security level of none, either because application or network security is specified, you must consider that no userid is available on inbound TP. By default, IMS will use:

- Inbound message region userid, if the inbound TP is a CPI-C driven application.
- Outbound LU name, if the inbound TP is a standard IMS application.

In this case you must authorize the user, depending on your environment, to execute the TP. It's not over, though. Example 2-12 shows some failures due to needed permissions.

Example 2-12 Possible error when no security info is available on inbound TP

```
15.29.54 JOB19170  ICH408I JOB(IVP8IM11) STEP(REGION  ) CL(JESSPOOL)
   468               WTSCPLX1.MASSIMO.IVP8IM11.JOB19170.D0000107.?
   468               WARNING: INSUFFICIENT AUTHORITY - TEMPORARY ACCESS ALLOWED
   468               FROM ** (G)
   468               ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
 15.29.54 JOB19170  +IMS2IMI: USERID=
 15.29.54 JOB19170  ICH408I USER(SCSIM8HA) GROUP(        ) NAME(???                 )
   470               LOGON/JOB INITIATION - USER AT TERMINAL          NOT RACF-DEFINED
 15.29.54 JOB19170  IRR012I  VERIFICATION FAILED. USER PROFILE NOT FOUND.
 15.29.54 JOB19170  +IMS2DB2: DB2_Error Entry, a SQLERROR has occurred.
 15.29.54 JOB19170  +IMS2DB2: DB2_Error Error Text Begins..
 15.29.54 JOB19170  +IMS2DB2: DB2_Error DSNT408I SQLCODE = -922, ERROR: AUTHORIZATION FAILURE: USER
                                                                                 AUTHORIZATION
 15.29.54 JOB19170  +IMS2DB2: DB2_Error           ERROR. REASON 00F30058
 15.29.54 JOB19170  +IMS2DB2: DB2_Error  DSNT418I SQLSTATE   = 42505 SQLSTATE RETURN CODE
 15.29.54 JOB19170  +IMS2DB2: DB2_Error  DSNT415I SQLERRP    = DSNAET03 SQL PROCEDURE DETECTING ERROR
```

Example 2-12 shows, in a conversation with a security level of none, the error that a standard IMS inbound transaction encountered when the LU name (in our test SCSIM8HA) was not RACF-defined.

2.5 APPC/MVS ISPF admin panels

In order to be able to use the APPC/MVS ISPF panels, you need to perform the following steps:

1. Update the logon procedure to allocate the APPC ICQ libraries to ISPPLIB, ISPSLIB, ISPMLIB, ISPTABLS and SYSPROC.

   In your SYSPROC concatenation:

   ```
   //        DD DSN=ICQ.ICQCCLIB,DISP=SHR
   ```

   In your ISPMLIB concatenation:

   ```
   //         DD DSN=ICQ.ICQMLIB,DISP=SHR
   ```

   In your ISPPLIB concatenation:

   ```
   //         DD DSN=ICQ.ICQPLIB,DISP=SHR
   ```

   In your ISPTLIB concatenation:

   ```
   //         DD DSN=ICQ.ICQTABLS,DISP=SHR
   ```

   In your ISPSLIB concatenation:

   ```
   //         DD DSN=ICQ.ICQSLIB,DISP=SHR
   ```

   For installation under Application Manager allocate ICQ.ICQILIB and ICQ.ICQABTXT.

   For more details, see *APPC/MVS Management*, SA22-7599.

2. Enter TSO ICQASRM0 from the TSO command line of TSO/E to start the ISPF TP_Profile System Data Facility Maintenance Utility from a TSO user ID. Once the panels are working, a panel similar to the following should be displayed:

```
-------------------------------------------------------------------------------
                          APPC Administration
Command ===>

Select one of the following with an "S".  Then Enter.
Type information.  Then Enter.

 _  TP Profile Administration
      Current TP Profile
        System file  . .  SYSx.APPCTP_________________________________

 _  Side Information Administration
      Current Side Information
        System file  . .  SYSx.APPCSI_________________________________

 _  Database Token Administration
      Current Database Token
        System file  . .  SYSx.APPCTP_________________________________

Note: For a list of file names, add an "*" suffix to the partial data set name.

PF01 = Help     PF03 = Exit     PF12 = Cancel
-------------------------------------------------------------------------------
```

3

Protected Conversations exploiters

This chapter provides details about IMS, CICS, and DB2 APPC Protected Conversations and the impact of choosing these types of conversational processing.

The following topics are discussed:

- IMS Protected Conversations
- CICS Protected Conversations
- DB2 Protected Conversations

3.1 IMS Protected Conversations

Working in a distributed environment, a conversation is a series of synchronous message exchanges between two partner programs that have a peer-to-peer relationship. Both partners must be active for the duration of the conversation. Either partner can initiate and direct the communications, and both sides can send or receive data. In general, a receiver cannot send data until the sender surrenders that right. In the conversational model both sides have to know the logic of the conversation, and they have to agree on the state of the conversation at any given time.

The conversational models are: Systems Application Architecture® (SAA®), Common Programming Interface Communications (CPI Communications) and Advanced Program-to-Program Communications (APPC).

You can use IMS Message Queue DL/I calls, APPC, or CPI Communications calls to participate in a conversation with a partner program.

Conversations are protected or recoverable when resources are updated in a controlled and synchronized manner. These resources can all reside locally (on the same system) or be distributed (across nodes in the network).

RRS/MVS is the syncpoint manager, also known as the coordinator. The syncpoint manager controls the commitment of protected resources by coordinating the commit request (or backout request) with the resource managers, the participating owners of the updated resources.

The protocols and mechanisms for regulating the updating of multiple protected resources in a consistent manner is provided in z/OS with RRS/MVS. Three participants are involved in the RRS/MVS process: the Syncpoint manager, Resource managers, and the Application program.

A transaction program

The APPC/IMS function in IMS TM supports the use of the CPI communication interface to build CPI-C application programs. Also, APPC/IMS allows distributed and cooperative processing between IMS and systems that have implemented APPC. APPC/IMS delivers support for APPC with facilities provided through APPC/MVS. The APPC/IMS interface is provided by APPC/MVS and supports the CPI Communications interface. IMS TM supports the CPI resource recovery interface.

APPC/IMS supports the CPI resource recovery Commit (SRRCMIT) and Backout (SRRBACK) calls for IMS-managed local resources.

A conversation

IMS functions as a resource manager protecting resources in a conversation. Specifically, IMS:

- Provides an application programming interface (API) to allow application programs to access its needed resources
- Logs changes to data before making the changes permanent
- Logs unit of work status
- Participates in the commit or backout actions for updated resources
- Contains recovery mechanisms to restore data to a previous state

IMS-dependent regions are automatically defined to APPC as subordinate address spaces of the IMS Scheduler. An IMS BMP cannot be defined as an ASCH controlled application. It may use explicit conversation services through the IMS base LU.

IMS manages the APPC/IMS buffers automatically; no definition is necessary. No special considerations are needed for EMH.

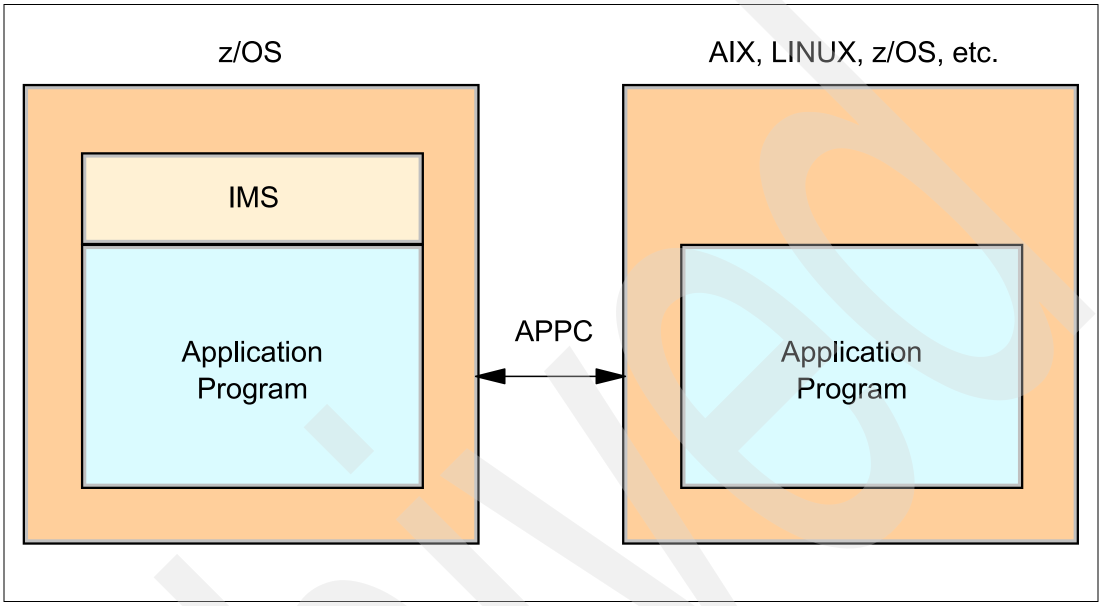


Figure 3-1 APPC support for IMS

Initiating protected conversations

When APPC is the communications manager, RRS/MVS support is activated when a conversation is allocated with SYNCLVL=SYNCPT. This type of conversation is a protected conversation.

When a protected conversation's inbound allocate request arrives at an APPC/MVS managed LU that is associated with IMS, APPC acquires a private context on behalf of IMS. IMS already provided its resource manager name to APPC in its Identify call when APPC/IMS was activated on the system. APPC uses this resource manager name to help create this privately-managed context. APPC passes the context token of this privately-managed context to IMS in a message when it passes the inbound request to IMS. IMS, using this context, then assumes the role of a participant in the two-phase commit process with the sync-point manager, RRS/MVS.

To achieve that, the SYNCLVL=SYNCPT and the keyword ATNLOSS=ALL must be specified in the VTAM definition file for whichever LUs the user wishes to enable for protected conversations.

APPC/IMS protected conversations are based on APPC/MVS services that provide extended functionality. APPC/MVS is a started task in a separate z/OS address space and has its own scheduler.

APPC/IMS delivers support for APPC with facilities provided through APPC/MVS. The APPC/IMS interface is provided by APPC/MVS and supports the CPI Communications interface. IMS TM supports the CPI resource recovery interface.

APPC/IMS supports the CPI resource recovery Commit (SRRCMIT) and Backout (SRRBACK) calls for IMS-managed local resources.

IMS TM allows APPC/IMS CPI Communications interface to build CPI application programs. Also, APPC/IMS allows distributed and cooperative processing between IMS and systems that have implemented APPC.

APPC/IMS also supports the existing IMS DL/I application programming interface, enabling application programs to use LU 6.2 communications without the function of the CPI Communications interface. This allows most existing applications to continue to function with the APPC/IMS support of LU 6.2.

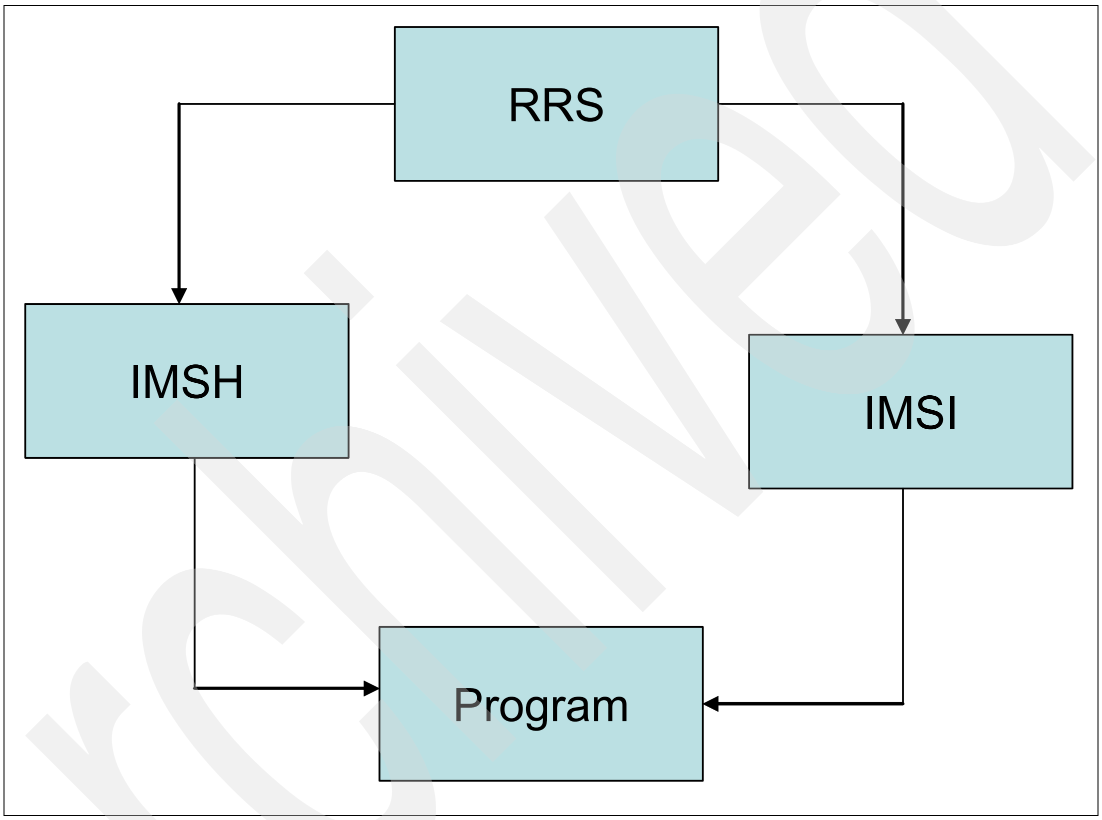


Figure 3-2 Recovery Resource Coordinator

3.1.1 Administering IMS and LU 6.2 devices

This section describes how to administer APPC/IMS and LU 6.2 devices.

Enabling APPC/IMS

Define APPC/IMS to VTAM

Define a VTAM Major Node for APPC/IMS in SYSx.VTAMLST. This node is needed by the IMS Subsystem to support all inbound and outbound conversations. Refer to “Add IMS APPC LUs to VTAM” on page 130 for further details.

Define IMS to APPC/MVS

Define the IMS APPC/IMS node as an APPC/MVS Scheduler. Add a LUADD statement for the APPC/IMS node to the member APPCPMxx in z/OS SYSx.PARMLIB. The name of the IMS Subsystem is coded on the SCHED subparameter; this indicates to APPC/MVS that the

IMS Subsystem assumes responsibility for scheduling inbound conversations into the IMS dependent regions. Refer to"Add IMSH and IMSI APPC LUs to APPC" on page 131 for an example and further details.

Enable IMS/APPC

Start APPC/IMS by specifying APPC=Y on the IMS startup parameter. The default is APPC=N. When Y is specified, IMS establishes a connection with APPC/MVS during IMS initialization. The /START APPC command overrides APPC=N.

APPC/MVS TP Profiles

The TP_Profile is a VSAM data set owned by APPC/MVS and maintained by the MVS System Data File Manager utility (ATBSDFMU) or by the administrator using TSO/ISPF dialogs. The purpose of the TP_Profile entries is to provide attribute information for TP names.

CPI Communications driven application programs must be defined in the APPC/MVS TP_Profile. IMS system-defined transaction codes can optionally be defined in a TP_Profile. The existence of an IMS definition (in the IMS GEN or by online change) causes the transaction to be considered a standard DL/I or modified-standard application.

The TP_Profile, an APPC/MVS resource, contains definitions of transaction program names (TPNs) and their characteristics. You can define a TP_Profile to schedule an IMS transaction program that uses a transaction code that is different from the TPN.

The TP Profile data set and TP Profiles are not required from an IMS point of view. If no TP Profile is defined, then the first 8 characters of the TP name are used as a trancode name unless a different trancode is specified in the TP Profile data set.

3.1.2 APPC/IMS application program interfaces

APPC/IMS has two distinct application program interfaces (APIs): the implicit and explicit interfaces. An existing program (implicit) can be modified to add access to native APPC verbs (explicit).

The IMS APPC API

The implicit API is an extension of the IMS standard DL/I API (call xxxTDLI). It allows IMS application programs to communicate with LU 6.2 application programs without being sensitive to LU 6.2 protocols and without requiring the programmer to have any knowledge of LU 6.2.

APPC/IMS provides functionality not available to an LU 6.2 application program: message queuing, and automatic asynchronous message delivery and recovery. Existing IMS transactions use the implicit API to communicate with APPC with no need for modification.

Implicit API messages are placed on the IMS message queues, or the Fast Path expedited message handling (EMH) buffers for Fast Path transactions. The originating IMS determines whether to mark the input messages as discardable or nondiscardable.

When the implicit API is used, IMS issues all required CPI Communications calls. The application program interacts strictly with the IMS message queues or the Fast Path EMH buffers.

These protected resources include:

- IMS TM message-queue messages
- IMS DB databases
- DB2 databases

APPC/IMS also supports the existing IMS DL/I API, enabling application programs to use LU 6.2 communications without the function of the CPI Communications interface. This allows most existing applications to continue to function within the APPC/IMS support of LU 6.2.

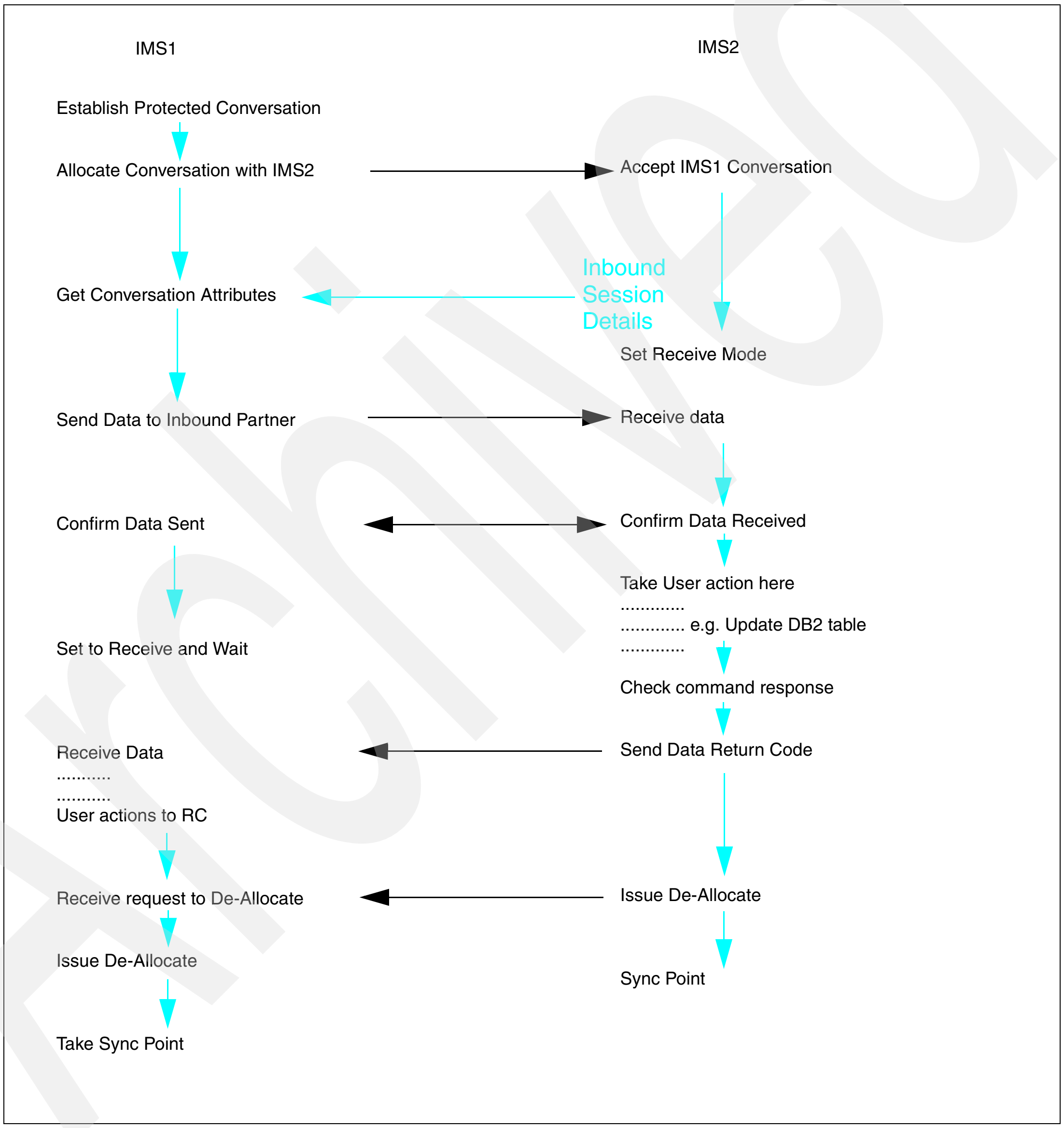


Figure 3-3 Overview APPC conversation

APPC/IMS application programs

APPC/IMS has three different types of application programs:

1. Standard - No explicit use of CPI Communications facilities.

2. Modified - Uses the I/O PCB to communicate with the original input terminal. Uses CPI Communications calls to allocate new conversations and to send and receive data.
3. CPI Communications driven - Uses CPI Communications calls to receive the incoming message and to send a reply on the same conversation. Uses the DL/I APSB call to allocate a PSB to access IMS databases and alternate PCBs.

Standard IMS application programs

Standard IMS application programs use the existing IMS call interface. Application programs that use the IMS standard API can take advantage of the LU 6.2 protocols. Standard IMS application programs use a DL/I GU call to trigger a sync point and to get the incoming transaction. These standard IMS application programs also use DL/I ISRT calls to generate output messages to the same or different terminals, which can be LU 6.2 terminals. The identical program can work correctly for both LU 6.2 and non-LU 6.2 terminal types. IMS generates the appropriate calls to APPC/MVS services.

IMS provides the following services for standard IMS application programs:

- Receives incoming transactions from an LU 6.2 application program
- Calls the Input Message Routing exit routine
- Schedules transactions into local and remote IMS dependent regions
- Provides necessary transaction recoverability
- Provides necessary transaction rollback and retry
- Integrates IMS-controlled conversation flows with database updates during syncpoint for all APPC Sync_Level options (NONE, CONFIRM, SYNCPT)
- Provides all needed LU 6.2 calls to APPC/MVS services
- Sends either synchronous or asynchronous output to an LU 6.2 application program
- Keeps asynchronous output on IMS message queue until successful transmission
- Allocates new LU 6.2 conversations for messages inserted to alternate PCBs using the DL/I ISRT call

Modified application programs

Modified IMS application programs use a DL/I GU call to retrieve the incoming transaction and to trigger a syncpoint. These modified IMS application programs also use DL/I ISRT calls to generate output messages to the same or different terminals, which can be LU 6.2 terminals. Unlike standard IMS application programs, modified IMS application programs use CPI Communications calls to allocate new conversations, and to send and receive data. IMS has no direct control of these CPI Communications conversations.

Modified IMS transactions are indistinguishable from standard IMS transactions until program execution. In fact, the same application program can be a “standard IMS” application on one execution, and a “modified IMS” application on a different execution. The distinction is simply whether the application program has used CPI Communications resources.

When an APPC program enters an IMS transaction that executes on a remote IMS, an LU 6.2 conversation is established between the APPC program and the local IMS. The local IMS is considered the partner LU of the LU 6.2 conversation. The transaction is then queued on the local IMS's remote transaction queue. From this point on, the transaction goes through normal MSC processing. After the remote IMS executes the transaction, the output is returned to the local IMS, and then delivered to the originating LU 6.2 program.

IMS provides the following services for modified IMS application programs:

- Receives incoming transactions from LU 6.2 application programs

- Schedules transactions into local and remote dependent IMS regions
- Provides appropriate transaction recoverability before transaction scheduling
- Integrates IMS-controlled conversation flows with database updates during syncpoint for APPC Sync_Level options (NONE, CONFIRM, SYNCPT)
- Provides all necessary LU 6.2 calls to APPC/MVS services for IMS-controlled LU 6.2 conversations
- Sends either synchronous or asynchronous output to LU 6.2 application programs
- Keeps asynchronous output on the IMS message queue until a successful send occurs
- Allocates new LU 6.2 conversations for any messages inserted to alternate PCBs using the DL/I ISRT calls

Common Programming Interface for Communication (CPI-C)

CPI Communications driven application programs are defined only in the APPC/MVS TP_Profile data set; they are not defined to IMS. Their definition is dynamically built by IMS when a transaction is presented for scheduling by APPC/MVS based on the APPC/MVS TP_Profile definition after IMS restart. The definition is keyed by TP name. APPC/MVS manages the TP_Profile information.

When a CPI Communications driven transaction program requests a PSB, the PSB must already be defined to IMS using the APPLCTN macro for SYSGEN and using PSBGEN/ACBGEN when APPLCTN PSB= is specified. When APPLCTN GPSB= is specified, a PSBGEN/ACBGEN is not required.

CPI Communications driven application programs must use CPI Communications calls to accept the incoming conversation and to send a reply on the same conversation. The DL/I GU call is not used to retrieve the initiating transaction from the LU 6.2 application program. No IMS resources are allocated when the application program is scheduled. Instead, the application program can use the DL/I APSB call to allocate a PSB that provides access to IMS databases and to alternate PCBs. A CPI Communications driven application program can send messages to other terminals (either LU 6.2 or non-LU 6.2) or other IMS transactions (either local or remote) by inserting to an alternate PCB, after allocating the appropriate PSB. Both the explicit and implicit API can be used on the same application program.

IMS provides the following services for CPI Communications driven application programs:

- Schedules the transaction. IMS does not receive input before scheduling. It does not interact with the conversation at any time other than to possibly reject the inbound allocate request. If IMS rejects the inbound allocate request, the transaction is not scheduled.
- Provides sync point of local resources.
- Schedules PSB if called by application program.
- Processes calls to alternate or database PCB made by the application program.

The explicit API is the CPI Communications API; it is available to any IMS application program. The application program makes calls to APPC using the CPI Communications interface without using IMS. These CPI calls are handled directly by APPC/MVS. Messages sent or received by the CPI Communications interface are not stored on the IMS message queues or in the EMH buffers, and these messages are not available for transaction restart. No IMS-provided functions are involved for these messages.

Attention: During APPC/IMS setup, we recommend the following:

- Define your APPC/IMS LUs for use by APPC/MVS, as well as by any APPC application program.
- Use the LTERM and the MOD name in the first segment of your message. Use the LTERM to change the destination for your output to a non-LU 6.2 device. Use the MOD name to format error messages.
- Use a network-qualified LU name. You do not need to have unique names for the LUs on different systems.

3.2 CICS protected conversations

Figure 3-4 depicts the basic components involved in an APPC Conversation.

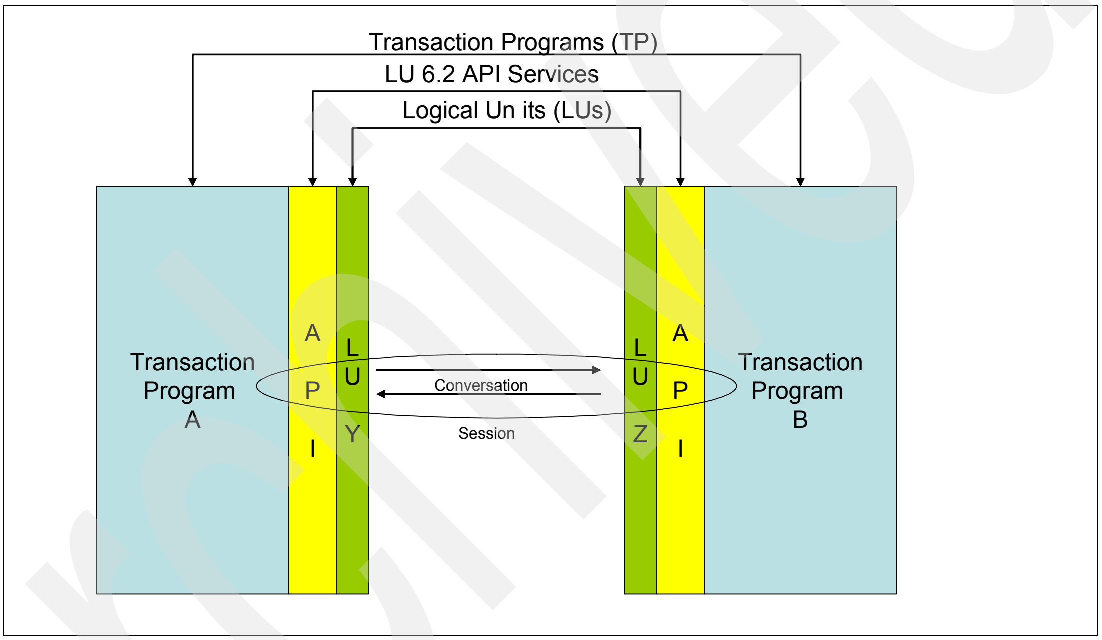


Figure 3-4 Overview APPC conversation

A connection between LUs is called a session. This connection can pass through intermediate network nodes. However, LU 6.2 programs do not need to account for the details of the connection. It makes no difference to the client transaction program whether the server transaction program is in the same room or thousands of miles away. The LU 6.2 API is responsible for starting and ending sessions between LUs of type 6.2.

A transaction program

A transaction program is a part of an application program that uses APPC communications functions. Application programs use these functions to communicate with application programs on other systems that support APPC. Your transaction program can obtain LU 6.2 services through either of the following APIs:

- APPC: Advanced Program-to-Program Communication allows transaction programs to exchange information across an IBM SNA network using the syntax and verbs defined by IBM for using an LU 6.2 session.

- CPI-C: Common Programming Interface for Communications (CPI-C) allows transaction programs to exchange information across an IBM SNA network using the syntax, defined by IBM in the Common Programming Interface component of the SAA(R), for using an LU 6.2 session. Because this API is implemented for many platforms, CPI-C applications can be easily ported.

A conversation

The communication between transaction programs is called a conversation. Conversations occur across LU-LU sessions. A conversation starts when a transaction program issues an APPC verb or CPI Communications call that allocates a conversation. When a conversation is allocated to a session, a send-receive relationship is established between the transaction programs connected to the conversation. One transaction program issues verbs to send data, and the other transaction program issues verbs to receive data. When it finishes sending data, the sending transaction program can transfer send control of the conversation to the receiving transaction program. One transaction program decides when to end the conversation and informs the other when it has ended.

The scope of a conversation_ID within CICS is one CICS task. A conversation_ID is created when a task initializes or accepts a conversation. Thereafter, any CICS application running under this task can use the conversation_ID to issue verbs against the conversation during its lifetime.

Initiating protected conversations

We initiate a CICS APPC conversation by using SYNCPOINT two-phase commit processing. You can set this scope for conversation by initializing the Outbound program with a:

```
synclvl=syncpnt
```

Once we have established a syncpoint level of “syncpnt” we register our intention with Resource Recovery Service (RRS/MVS). CICS becomes the Resource Manager and RRS the recovery controller. In plain terms, what this means is that when the partner program performs a COMMIT or ROLLBACK, RRS informs CICS about the COMMIT or ROLLBACK so that the conversation remains in a protected state. CICS is notified of the change of state and in response, can carry out its COMMIT or ROLLBACK processing.

3.2.1 Administering CICS and LU 6.2 devices

This section describes how to administer CICS and LU6.2 devices. First of all, we need to introduce the concept of a logical unit or LU device.

A logical unit

Every transaction program gains access to an SNA network through a logical unit (LU). An LU is SNA software that accepts verbs from your programs and acts on those verbs. A transaction program issues APPC verbs to its LU. These verbs cause commands and data to flow across the network to a partner LU. An LU also acts as an intermediary between the transaction programs and the network to manage the exchange of data between transaction programs. A single LU can provide services for multiple transaction programs. Multiple LUs can be active simultaneously.

Enabling CICS APPC

You have to define a CICS CONNection and a CICS SESSion to a CICS system on which you wish to execute APPC conversations. Refer to“Add CICS APPC support” on page 116 for further details on the installation definitions and requirements.

CICS VTAM Major Node

You must define a CICS VTAM Major Node in SYSx.VTAMLST. Refer to “Update CICS VTAM APPL definition” on page 118 for further details. CICS VTAM Major node definitions require extra care when defining them for LU 6.2 communications. Pay particular attention to the values associated with the operands in the VTAM APPL statement. For more information see “ACF/VTAM definition for CICS” on page 119.

For further information about the VTAM APPL statement, refer to *OS/390 eNetwork Communications Server: SNA Resource Definition Reference.*

CICS LOGMODE tables

A CICS system requires a Modetable and an entry within the table to be used for LU6.2 conversations. The VTAM APPL parameter MODETAB=LOGMODES indicates the Mode Table. The VTAM APPL parameter DLOGMOD indicates the entry (MODEENT) within the Mode Table. The Logmode entry used by the IMS system and CICS systems needs to be aligned.

For example, for CICS-to-IMS links that are cross-domain, you must associate the IMS LOGMODE entry with the CICS applid (the generic applid for XRF systems), using the DLOGMOD or MODETAB parameters. Ensure the CICS Mode Table entry contains a Logmode table entry (MODEENT) which is the same as the Logmode table entry used by IMS when allocating the conversation.

With APPC sessions, you can use the MODENAME as specified in the CICS DEFINE SESSIONS definitions. Any modename that you supply for a CICS session must be matched.

For details of installation requirements used here, see“ACF/VTAM LOGMODE table entries” on page 114.

For programming information about coding the VTAM LOGON mode table, refer to *CICS Transaction Server for z/OS V2.3 CICS Customization Guide.*

3.2.2 APPC/CICS application program interface

APPC/CICS has distinct APIs to perform the communication protocol; they are described in this section.

The CPI-C

The Common Programming Interface for Communications (CPI-C) is a powerful and flexible API, but it can be complex and difficult to use. CPI-C complies with Systems Application Architecture (SAA) mandates to unify different platforms and operating systems. CPI-C uses a set of syntax rules that is common to all systems. CPI-C applications follow the peer-to-peer model in which all partners in a conversation are equal peers. Data must be explicitly sent to and received from a peer. Conversations are not automatically allocated, deallocated, or reused. CPI-C supports all conversation synchronization levels.

> **Note:** A major benefit of the common APPC standard is that applications that use CPI-C can communicate with applications on any system that provides an APPC API. This includes applications on different CICS platforms.

CPI-C in CICS provides an alternative API to existing CICS APPC communications support. Users who have already made a skill investment in the existing EXEC CICS programming interface or who do not expect to require the cross-system consistency benefits offered by

CPI-C, might choose to continue using the EXEC CICS API. CICS itself provides no CPI-C extension calls.

Attention: A CICS transaction program can use both CICS APPC API commands and CPI-C calls in the same program, but may *not* use both in the same conversation.

The CICS APPC API (Exec CICS commands)

Lets take a look at what types of techniques we have to employ with the CICS-implemented APPC architecture in order to ensure a protected conversation in various situations.

These examples show how to commit and back out changes to recoverable resources in a conversation using synchronization level 2 (Protected).

These examples illustrate the following scenarios:

- SYNCPOINT in response to SYNCPOINT
- SYNCPOINT in response to ISSUE PREPARE
- SYNCPOINT ROLLBACK in response to SYNCPOINT ROLLBACK
- SYNCPOINT ROLLBACK in response to SYNCPOINT
- SYNCPOINT ROLLBACK in response to ISSUE PREPARE
- ISSUE ERROR in response to SYNCPOINT
- ISSUE ERROR in response to ISSUE PREPARE
- ISSUE ABEND in response to SYNCPOINT
- ISSUE ERROR in response to ISSUE PREPARE

SYNCPOINT in response to SYNCPOINT

Figure 3-5, Figure 3-6, and Figure 3-7 illustrate the effect of EXEC CICS SEND, EXEC CICS SEND INVITE, or EXEC CICS SEND LAST preceding EXEC CICS SYNCPOINT on an APPC mapped conversation. These figures also show the conversation state before each command and the state and EIB fields set after each command.

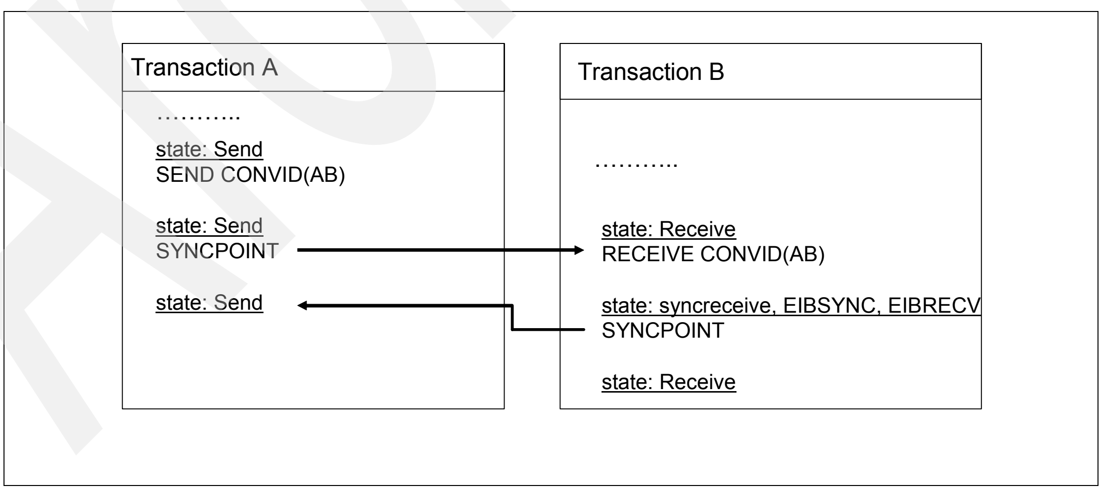


Figure 3-5 EXEC CICS SYNCPOINT in response to EXEC CICS SEND followed by EXEC CICS SYNCPOINT on a conversation

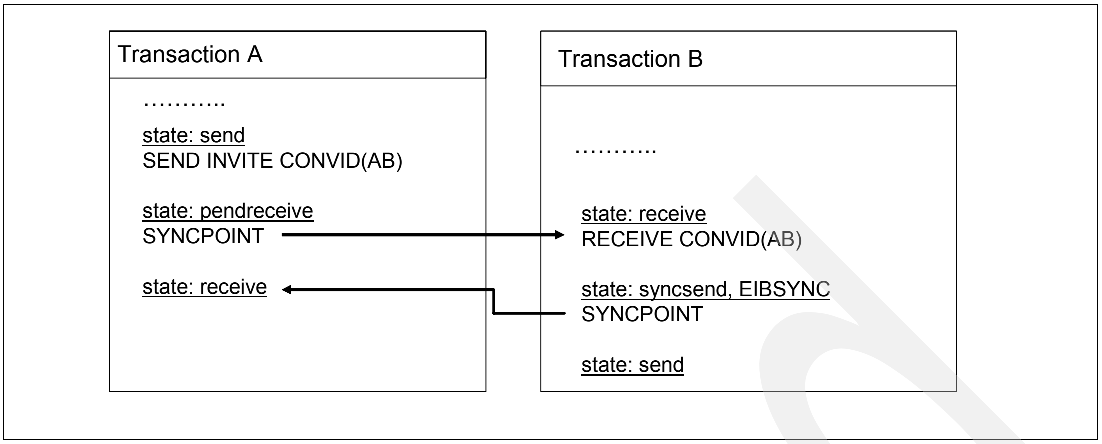


Figure 3-6 EXEC CICS SYNCPOINT in response to EXEC CICS SEND INVITE followed by EXEC CICS SYNCPOINT on a conversation

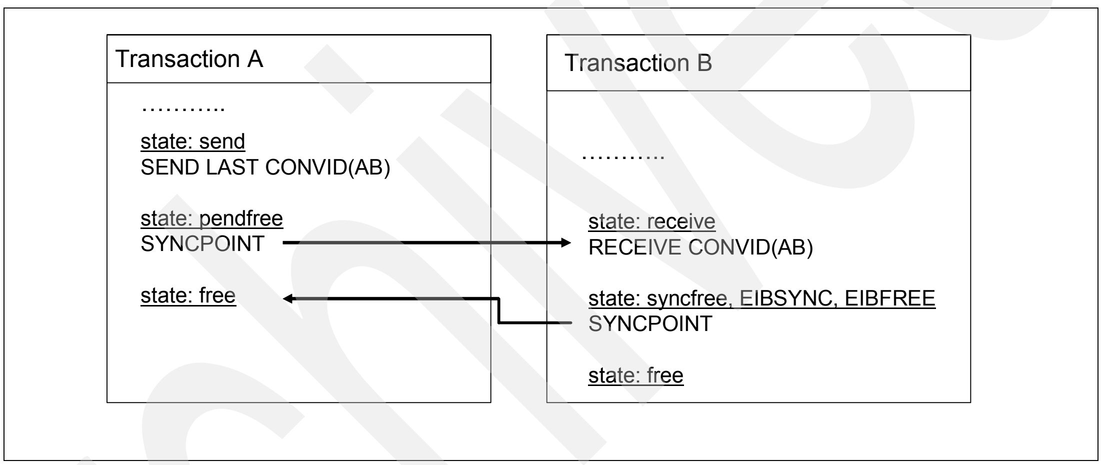


Figure 3-7 EXEC CICS SYNCPOINT in response to EXEC CICS SEND LAST followed by EXEC CICS SYNCPOINT on a conversation

SYNCPOINT in response to ISSUE PREPARE

Figure 3-8 illustrates an EXEC CICS SYNCPOINT command being used in response to EXEC CICS ISSUE PREPARE on a conversation. This figure also shows the conversation state before each command and the state and EIB fields set after each command.

> **Note:** It is also possible to use an EXEC CICS ISSUE PREPARE command in pendreceive state (state 3) and pendfree state (state 4).
>
> Also, although the EXEC CICS ISSUE PREPARE command in the figure returns with the conversation in syncsend state (state 10), the only commands available for use on that conversation are EXEC CICS SYNCPOINT and EXEC CICS SYNCPOINT ROLLBACK. All other commands abend ATCV.

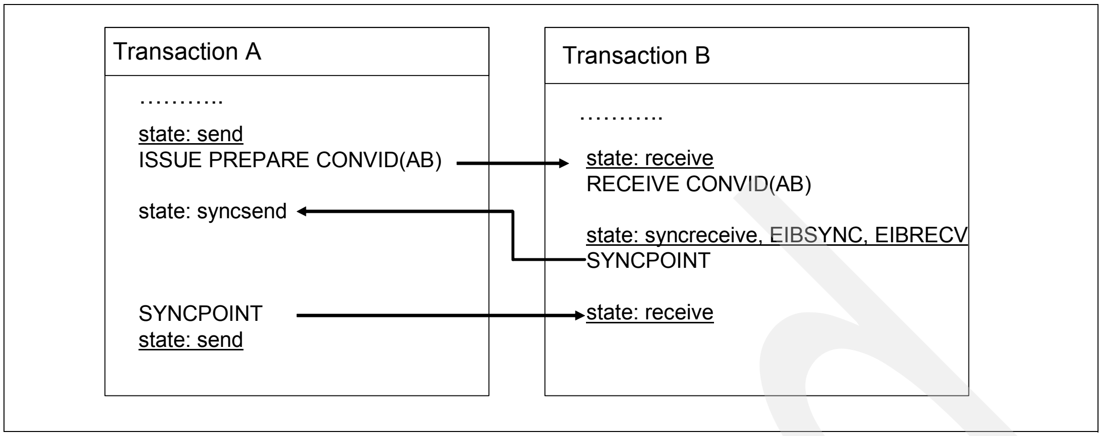


Figure 3-8 EXEC CICS SYNCPOINT in response to EXEC CICS ISSUE PREPARE on a conversation

SYNCPOINT ROLLBACK in response to SYNCPOINT ROLLBACK

Figure 3-9 illustrates an EXEC CICS SYNCPOINT ROLLBACK command being used in response to EXEC CICS SYNCPOINT ROLLBACK on a conversation. This figure also shows the conversation state before each command and the state and EIB fields set after each command.

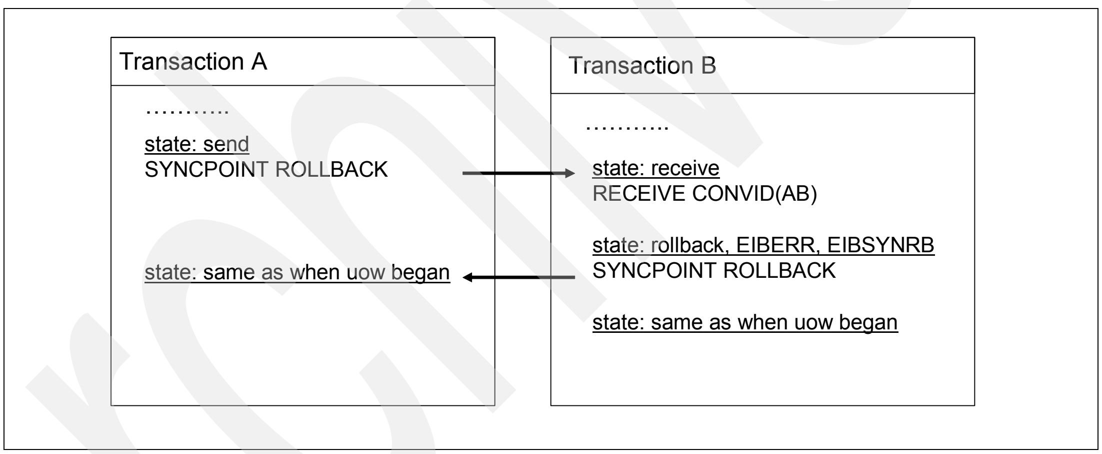


Figure 3-9 EXEC CICS SYNCPOINT ROLLBACK in response to EXEC CICS SYNCPOINT ROLLBACK on a conversation

SYNCPOINT ROLLBACK in response to SYNCPOINT

Figure 3-10 illustrates an EXEC CICS SYNCPOINT ROLLBACK command being used in response to EXEC CICS SYNCPOINT on a conversation. This figure also shows the conversation state before each command and the state and EIB fields set after each command.

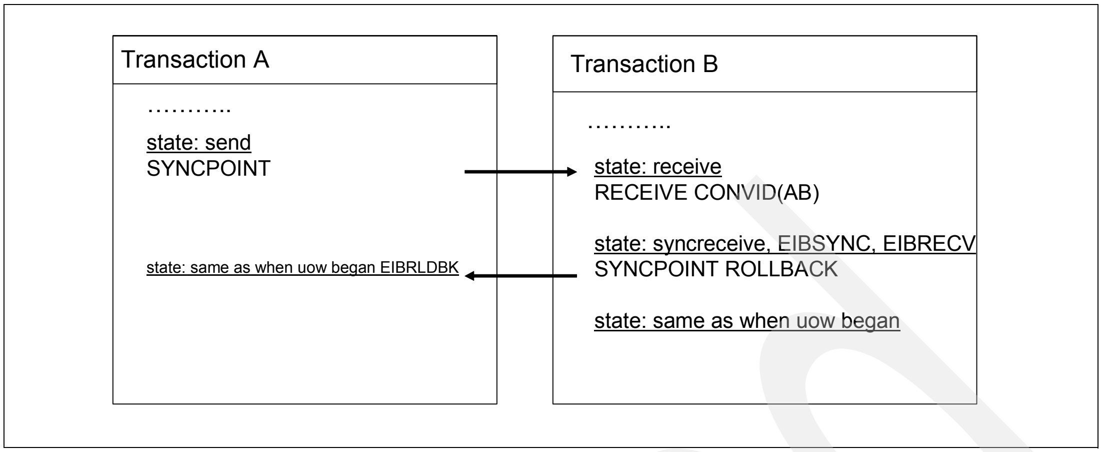


Figure 3-10 EXEC CICS SYNCPOINT ROLLBACK in response to EXEC CICS SYNCPOINT on a conversation

SYNCPOINT ROLLBACK in response to ISSUE PREPARE

Figure 3-11 illustrates an EXEC CICS SYNCPOINT ROLLBACK command being used in response to EXEC CICS ISSUE PREPARE on a conversation. This figure also shows the conversation state before each command and the state and EIB fields set after each command.

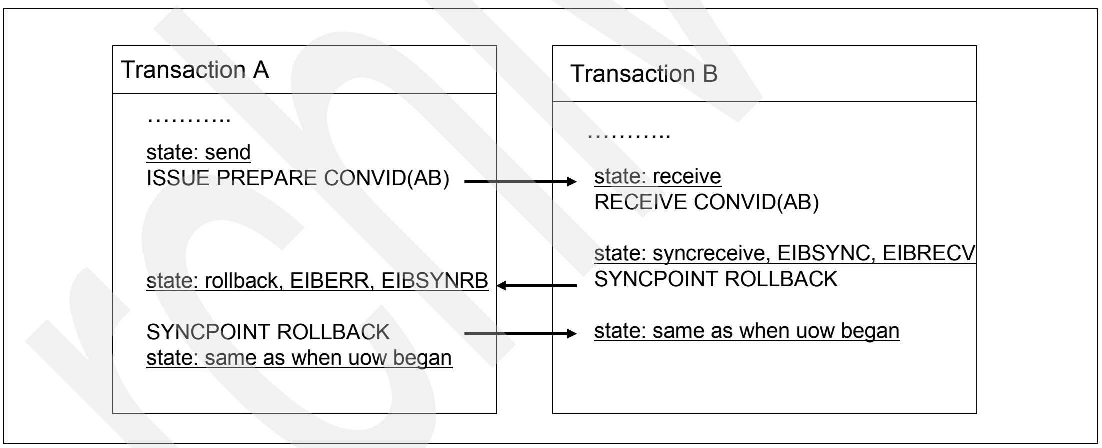


Figure 3-11 EXEC CICS SYNCPOINT ROLLBACK in response to EXEC CICS ISSUE PREPARE on a conversation

ISSUE ERROR in response to SYNCPOINT

Figure 3-12 illustrates an EXEC CICS ISSUE ERROR command being used in response to EXEC CICS SYNCPOINT on a conversation. The figure also shows the conversation state before each command and the state and EIB fields set after each command. You can also send EXEC CICS ISSUE ERROR before receiving EXEC CICS SYNCPOINT; this is not shown because the results are the same.

It is pointless to use EXEC CICS ISSUE ERROR as a response to EXEC CICS SYNCPOINT, because this causes the syncpoint initiator to discard all data transmitted with the EXEC CICS ISSUE ERROR by the syncpoint agent. To safeguard integrity, the syncpoint agent has to issue a EXEC CICS SYNCPOINT ROLLBACK command.

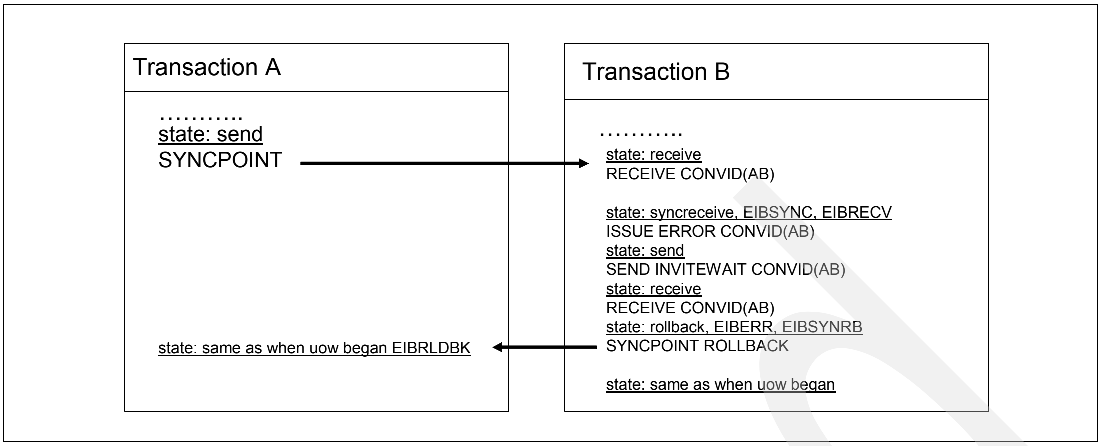


Figure 3-12 EXEC CICS ISSUE ERROR in response to EXEC CICS SYNCPOINT on a conversation

ISSUE ERROR in response to ISSUE PREPARE

Figure 3-13 illustrates an EXEC CICS ISSUE ERROR command being used in response to EXEC CICS ISSUE PREPARE on an APPC mapped conversation. This figure also shows the conversation state before each command and the state and EIB fields set after each command. You can also send EXEC CICS ISSUE ERROR before receiving EXEC CICS ISSUE PREPARE; this is not shown because the results are the same.

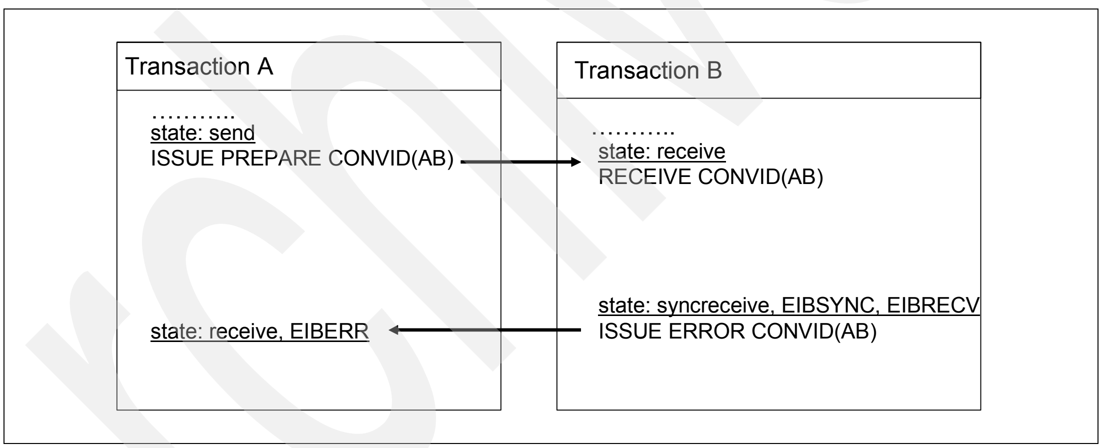


Figure 3-13 EXEC CICS ISSUE ERROR in response to EXEC CICS ISSUE PREPARE on a conversation

ISSUE ABEND in response to SYNCPOINT

Figure 3-14 illustrates an EXEC CICS ISSUE ABEND command being used in response to EXEC CICS SYNCPOINT on a conversation. The figure also shows the conversation state before each command and the state and EIB fields set after each command. You can also send EXEC CICS ISSUE ABEND before receiving EXEC CICS SYNCPOINT; this is not shown because the results are the same.

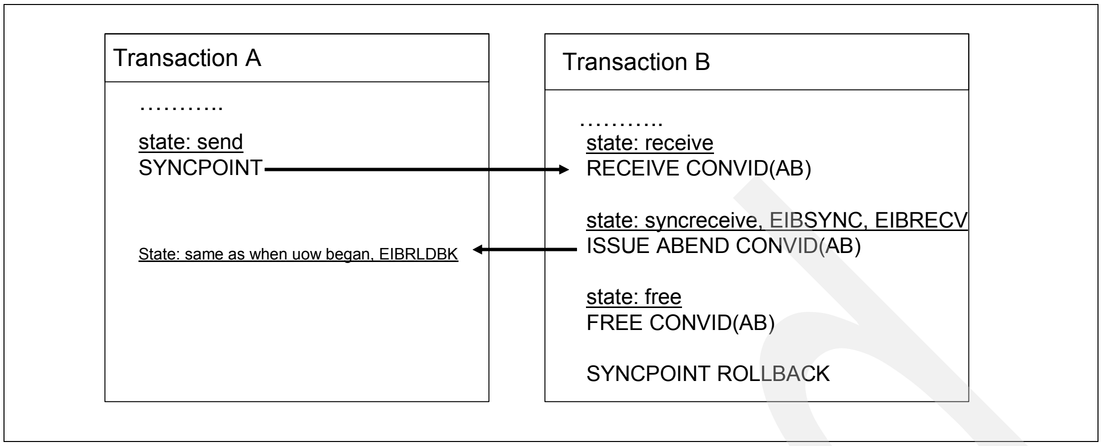


Figure 3-14 EXEC CICS ISSUE ABEND in response to EXEC CICS SYNCPOINT on a conversation

ISSUE ABEND in response to ISSUE PREPARE

Figure 3-15 illustrates an EXEC CICS ISSUE ABEND command being used in response to EXEC CICS ISSUE PREPARE on a conversation. The figure also shows the conversation state before each command and the state and EIB fields set after each command. You can also send EXEC CICS ISSUE ABEND before receiving EXEC CICS ISSUE PREPARE; this is not shown because the results are the same.

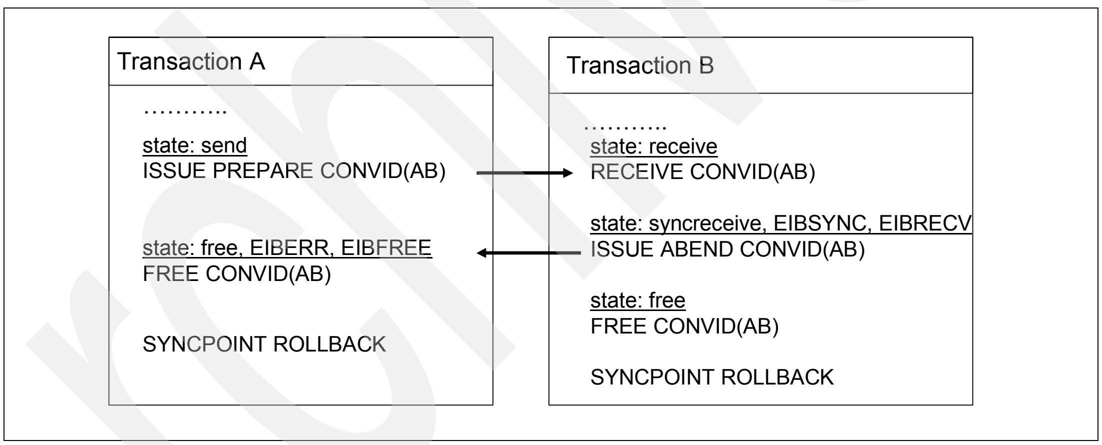


Figure 3-15 EXEC CICS ISSUE ABEND in response to EXEC CICS ISSUE PREPARE on a conversation

Summary

Two transaction programs use LU 6.2 to exchange data in a conversation. One, the client transaction program, is typically started by a user. The other, the server transaction program, can be started automatically to render a service to the client. A transaction program uses one of two APIs: APPC, or CPI-C, which have different styles and similar, but not identical, sets of services.

When using CICS the conversation takes place over a session between two LUs. An LU represents a point at which a transaction program can access the SNA network. A session represents the connection between two LUs, without regard to their location or the distance between them; in context here, a CICS LU and a partner LU. The Partner LU can be either Inbound or Outbound and can reside anywhere within the SNA network.

The most significant decision to be addressed when developing a CICS APPC Protected Conversation application is whether you employ the CICS APPC architecture as implemented by CICS TS 1.3 and latter or whether you use the more flexible but complicated CPI-Communications. CPI-C in CICS provides an alternative API to existing CICS APPC support. CPI-C provides distributed transaction processing (DTP) on APPC sessions and can be used in assembler language, COBOL, PL/I, or C.

CPI-C defines an API that can be used in APPC networks that include multiple system platforms, where the consistency of a common API is seen to be of benefit. The CPI-C interface can converse with applications on any system that provides an APPC API. This includes applications on CICS platforms. You may use APPC API commands on one end of a conversation and CPI-C commands on the other. CPI-C requires specific information (side information) to begin a conversation with a partner program. CICS implementation of side information is achieved using the partner resource which your system programmer is responsible for maintaining. The application's calls to the CPI-C interface are resolved by link-editing it with the CICS CPI-C link-edit stub (DFHCPLC).

For further details, refer to *CICS Applications Programmers Guide.*

Note: The CPI Communications API is defined as a general call interface. The interface is described in the *SAA Common Programming Interface Communications Reference*.

3.3 DB2

DB2, since V5 and within WLM-managed Stored Procedures, is able to manage APPC Protected Conversation too. We don't go into further detail on this topic because it is clearly and completely discussed in two other Redbooks. For further informations, refer to *Getting started with DB2 Stored Procedures: Give Them a Call through the Network,* SG24-4693 and *DB2 for z/OS Stored Procedures: Through the CALL and Beyond,* SG24-7083.

4

How to operate in an APPC/MVS Protected Conversations environment

This chapter describes the operational aspects of an APPC environment that relate to protected conversations. The following topics are discussed:

- Managing the resources
- Handling the failures

4.1 How to manage the resources

There is no online monitor that can be used to completely manage the APPC protected conversations environment. Because of the "multi-TP-monitor" nature of this environment, you will need to collect data from different sources to have a complete vision of the environment.

4.1.1 APPC commands

First of all, we need to know if our IMS LU has syncpoint capability. We can use the command **d appc,lu,all** to see the information shown in Example 4-1.

Example 4-1 Output from **d appc,lu,all** *command*

```
D APPC,LU,ALL
ATB121I  14.37.01  APPC DISPLAY 364
  ACTIVE LU'S    OUTBOUND LU'S    PENDING LU'S    TERMINATING LU'S
     00011          00000           00010              00000
  SIDEINFO=SYS1.APPCSI
LLUN=SC61OSA           SCHED=*NONE*        BASE=NO          NQN=NO
   STATUS=ACTIVE       PARTNERS=00000      TPLEVEL=SYSTEM   SYNCPT=NO
   GRNAME=*NONE*       RMNAME=*NONE*
   TPDATA=SYS1.APPCTP
LLUN=SCSIM8IA          SCHED=IMSI          BASE=YES         NQN=NO
   STATUS=PENDING      PARTNERS=00000      TPLEVEL=SYSTEM   SYNCPT=NO
   GRNAME=*NONE*       RMNAME=*NONE*
   TPDATA=SYS1.APPCTP
LLUN=SCSIM8HA          SCHED=IMSH          BASE=YES         NQN=NO
   STATUS=ACTIVE       PARTNERS=00001      TPLEVEL=SYSTEM   SYNCPT=YES
   GRNAME=*NONE*       RMNAME=ATB.USIBMSC.SCSIM8HA.IBM
   TPDATA=SYS1.APPCTP
 PLUN=USIBMSC.SCSCPA8K
LLUN=SC61IMAR          SCHED=IMSR          BASE=YES         NQN=NO
   STATUS=PENDING      PARTNERS=00000      TPLEVEL=SYSTEM   SYNCPT=NO
   GRNAME=*NONE*       RMNAME=*NONE*
   TPDATA=SYS1.APPCTP
LLUN=SC61IMSA          SCHED=IMSA          BASE=YES         NQN=NO
   STATUS=PENDING      PARTNERS=00000      TPLEVEL=SYSTEM   SYNCPT=NO
   GRNAME=*NONE*       RMNAME=*NONE*
   TPDATA=SYS1.APPCTP
LLUN=SC61SRV           SCHED=ASCH          BASE=NO          NQN=NO
   STATUS=ACTIVE       PARTNERS=00000      TPLEVEL=SYSTEM   SYNCPT=NO
   GRNAME=SCSSRV       RMNAME=*NONE*
   TPDATA=SYS1.APPCTP
```

Let's look more closely at the output in Example 4-1.

- LLUN=SCSIM8HA is the local LU name.
- SCHED=IMSH is the scheduler name as known in SYSx.PARMLIB(APPCPMxx).
- STATUS=ACTIVE means the LU is active.
- RMNAME=ATB.USIBMSC.SCSIM8HA.IBM is the APPC-generated resource manager name for the LU when LU is registered as a communications resource manager with RRS, and is capable of supporting protected conversations.
- SYNCPT=YES indicates the local LU is registered with RRS and is capable of supporting protected conversations.

- PLUN=USIBMSC.SCSCPA8K is the name of the partner LU which is already connected to (in our test the CICS LU name).

Therefore, if the **display appc,lu,all** command shows SYNCPT=YES, our IMS or DB2 LU has capability for protected conversations. In addition, APPC cuts many messages into the syslog, including at the time an LU is restarted. This is shown in Example 4-2.

Example 4-2 Syslog output extract after a successful restart of an IMS LU with syncpoint capability

```
ATB227I LOCAL LU USIBMSC.SCSIM8HA IS WARM STARTING AS A RESOURCE
MANAGER WITH RRS/MVS.
LOCAL LOG: ATR.BA04D30D06ADBE08.IBM
ATB201I LOGICAL UNIT SCSIM8HA FOR TRANSACTION SCHEDULER IMSH NOW
ACCEPTS PROTECTED CONVERSATIONS.
ATB207I EXCHANGE LOG NAME PROCESSING HAS COMPLETED SUCCESSFULLY
BETWEEN LOCAL LU USIBMSC.SCSIM8HA AND PARTNER LU
USIBMSC.SCSCPA8K.
LOCAL LOG: ATR.BA04D30D06ADBE08.IBM
PARTNER LOG: 00121160
ATB207I EXCHANGE LOG NAME PROCESSING HAS COMPLETED SUCCESSFULLY
BETWEEN LOCAL LU USIBMSC.SCSIM8HA AND PARTNER LU
USIBMSC.SCSIM8IA.
LOCAL LOG: ATR.BA04D30D06ADBE08.IBM
PARTNER LOG: ATR.BA04D30D06ADBE08.IBM
ATB207I EXCHANGE LOG NAME PROCESSING HAS COMPLETED SUCCESSFULLY
BETWEEN LOCAL LU USIBMSC.SCSIM8HA AND PARTNER LU
USIBMSC.SCSCPA8K.
LOCAL LOG: ATR.BA04D30D06ADBE08.IBM
PARTNER LOG: 00121160
```

In Example 4-2, the IMS LU, after it has been activated, contacts all the available partners which were connected to it before the failure. Let's look more closely at the messages from APPC (ATB*):

- ATB227I shows that this is a warm restart.
- ATB201I shows that the LU is registered to RRS and can accept protected conversation (as well as unprotected ones).
- ATB207I shows that the exchange log name (synchronization) between the two partners has been successfully completed. APPC sends this message for every partner joining the other. In our test you can look at the message for the SCSCPAK8 LU that, because it resides on the same system of SCSIM8HA, seems duplicated, but with the inversion of local and partner LUs.

APPC gives us a set of commands to display various aspects of its conversations status. In particular, we are interested in the following two commands because they provide different views of protected conversations information:

- **d appc,tp,all**
- **d appc,ur,all**

The **d appc,tp,all** command, within the ATB122I message, gives information about the TPs actually in use, their direction, partner, and so forth. The output of the commands, for both the systems involved in the conversation, are in Example 4-3 and Example 4-4.

Example 4-3 Output from **d appc,tp,all** *command at outbound system*

```
D APPC,TP,ALL
ATB122I  18.10.52  APPC DISPLAY 567
  LOCAL TP'S      INBOUND CONVERSATIONS      OUTBOUND CONVERSATIONS
```

```
    00001                00000                       00001
LTPN=*UNKNOWN*
   LLUN=SCSIM8HA   WUID=*UNKNOWN*    CONVERSATIONS=00001   ASID=0029
   SCHED=IMSH      ASNAME=IVP8HM11   TPID=224B29100000007F
 PTPN=IMS2IMP
   PLUN=USIBMSC.SCSIM8IA
   PROTECTED=YES   USERID=MASSIMO    DIRECTION=OUTBOUND
   VERBS=00000006  IT=027.923S
   MODE=APPCHOST   VTAMCID=0100015C  SYNC POINT IN PROG=NO
   LUWID=USIBMSC.SCSIM8HA 0CB3A2C1AC44 0001
```

Let's look more closely at the output in Example 4-3, referencing the outbound system:

- LTPN=*UNKNOWN* Outbound transaction programs are not required to have a TP name. The only TP Name required is the Transaction Program that the outbound is trying to establish a conversation with on the other side of the conversation APPC started program. In our example it is a "modified CPI-C" IMS transaction. If the outbound TP is an APPC started program the field is consequently filled.
- LLUN=SCSIM8HA is the local LU from which the outbound TP requests the connection.
- ASNAME=IVP8HM11 is the asname where the outbound TP runs.
- ASID=0029 is the asid of the asname where the outbound TP runs.
- SCHED=IMSH is the scheduler name for APPC as known in SYSx.PARMLIB(APPCPMxx).
- PTPN=IMS2IMP is the name of the inbound TP.
- PLUN=USIBMSC.SCSIM8IA is the name of the inbound LU.
- PROTECTED =YES means that this is an APPC/MVS Protected Conversation (Synclevel=Syncpoint).
- USERID=MASSIMO is the userid that initiates the conversation. For further information see 2.4, "Security considerations" on page 30.
- DIRECTION=OUTBOUND means that it is an outbound conversation.
- LUWID=USIBMSC.SCSIM8HA 0CB3A2C1AC44 0001 is an APPC identifier, and it is a useful token to chain this conversation with the IMS log records at the inbound side (basically IMS log records x"01" and x"0A08"). For further information, refer to *IMS Version 8 Diagnosis Guide and Reference,* LY37-3742.

Example 4-4 Output from **d appc,tp,all** *command at inbound system*

```
D APPC,TP,ALL
ATB122I  18.10.55  APPC DISPLAY 083
  LOCAL TP'S      INBOUND CONVERSATIONS      OUTBOUND CONVERSATIONS
    00001                00001                       00000
LTPN=IMS2IMP
   LLUN=SCSIM8IA   WUID=*UNKNOWN*    CONVERSATIONS=00001   ASID=0415
   SCHED=IMSI      ASNAME=IMS810I    TPID=224B291000000034
 PTPN=*UNKNOWN*
   PLUN=USIBMSC.SCSIM8HA
   PROTECTED=YES   USERID=MASSIMO    DIRECTION=INBOUND
   VERBS=00000005  IT=*NONE*
   MODE=APPCHOST   VTAMCID=0100014B  SYNC POINT IN PROG=NO
   LUWID=USIBMSC.SCSIM8HA 0CB3A2C1AC44 0001
```

Let's consider the output in Example 4-4, referencing the inbound system:

- LTPN=IMS2IMP is the name of the local inbound TP. Remember it must be defined in the TP or SIDEINFO profile of APPC. If it is an IMS CPI-C driven application, you must specify the RM-specific data.
- LLUN=SCSIM8IA is the local LU managing the request for the outbound TP.
- ASNAME=IMS810I is the asname where the inbound TP runs. For IMS it is the name of a dependent region. Otherwise, if it is the name of the control region, the transaction has not been already scheduled in any dependent region.
- ASID=0415 is the asid of the asname where the inbound TP runs. If the transaction has not been already scheduled, it is the IMS control region asid.
- SCHED=IMSI is the scheduler name for APPC as known in SYSx.PARMLIB(APPCPMxx).
- PTPN=*UNKNOWN* For inbound conversations, *UNKNOWN* always appears in this field.
- PLUN=USIBMSC.SCSIM8HA is the name of the outbound LU.
- PROTECTED=YES means that this is an APPC/MVS Protected Conversation (Synclevel=Syncpoint).
- USERID=MASSIMO is the userid transmitted from the TP that initiates the conversation. For further information refer to 2.4, "Security considerations" on page 30.
- DIRECTION=INBOUND means that it is an inbound conversation.
- LUWID=USIBMSC.SCSIM8HA 0CB3A2C1AC44 0001 is an APPC identifier and it is a useful token to chain this conversation with the local IMS log records (basically x"01" and x"0A08" IMS log records). For further information refer to *IMS Version 8 Diagnosis Guide and Reference,* LY37-3742.

> **Note:** To manage, from the APPC ISPF panels, the scheduling data of IMS, your TSO profile must have access to the IMS DFSTPPE0 load module. The message: `"TP Profile contains the unsupported scheduler exit DFSTPPE0 ICQAS594"` shows the unavailability of the load module.

The **`d appc,ur,all`** command, within the ATB104I message, gives information about the APPC unit of recovery actually running, the APPC luw, the RRS urid, and so forth. The outputs of the command, for both systems involved in the conversation, are in Example 4-5 and Example 4-6.

Example 4-5 Output from **`d appc,ur,all`** *command at outbound system*

```
D APPC,UR,ALL
ATB104I  18.11.00  APPC DISPLAY 569
  APPC UR'S                EXPRESSIONS OF INTEREST
    00001                           00001
URID=BC020CB37E6C4000000000CA01050000
   EXPRESSION OF INTEREST COUNT=00001          SYNC POINT IN PROG=NO
   LUWID=USIBMSC.SCSIM8HA 0CB3A2C1AC44 0001
 LTPN=*UNKNOWN*
 PTPN=IMS2IMP
   CONV CORRELATOR=BC020CB3A2B54204
   PLUN=USIBMSC.SCSIM8IA    LLUN=SCSIM8HA      DIRECTION=OUTBOUND
   RESYNC REQUIRED=NO       IMPLIED FORGET=NO
```

Let's consider the output in Example 4-5, referencing the outbound system:

- URID=BC020CB37E6C4000000000CA01050000 is the local RRS unit of recovery ID. It is useful to chain this information with outbound IMS log records (basically x"5611" and x"5616") or with the outbound RRS Archive Log. For further information refer to *IMS Version 8 Diagnosis Guide and Reference,* LY37-3742, and 7.4, "The RRS REXX batch log processor" on page 109. This URID can be used in the RRS ISPF interface to see the state of this particular UR from an RRS perspective.
- LUWID=USIBMSC.SCSIM8HA 0CB3A2C1AC44 0001 is the APPC luwid identifier and allows a chain to the **`d appc,tp,all`** command output.
- CONV CORRELATOR=BC020CB3A2B54204 is an ID to chain the inbound and outbound information together. If the outbound program, within the same urid, starts multiple inbound transactions on the same system, it becomes very useful.

Example 4-6 Output from **`d appc,ur,all`** *command at inbound system*

```
D APPC,UR,ALL
ATB104I  18.11.02  APPC DISPLAY 085
  APPC UR'S                EXPRESSIONS OF INTEREST
    00001                          00001
URID=BC020CB37E6D0000000000C2010E0000
   EXPRESSION OF INTEREST COUNT=00001          SYNC POINT IN PROG=NO
   LUWID=USIBMSC.SCSIM8HA 0CB3A2C1AC44 0001
 LTPN=IMS2IMP
 PTPN=*UNKNOWN*
   CONV CORRELATOR=BC020CB3A2B54204
   PLUN=USIBMSC.SCSIM8HA    LLUN=SCSIM8IA      DIRECTION=INBOUND
   RESYNC REQUIRED=NO       IMPLIED FORGET=NO
```

Let's look more closely at the output in Example 4-6, referencing the inbound system:

- URID=BC020CB37E6D0000000000C2010E0000 is the local RRS unit of recovery ID. It is useful to chain this information with inbound IMS log records (basically x"5611" and x"5616") or with the inbound RRS Archive Log. For further information refer to *IMS Version 8 Diagnosis Guide and Reference,* LY37-3742, and 7.4, "The RRS REXX batch log processor" on page 109. This URID can be used with the RRS ISPF interface to see the state of this particular UR from an RRS perspective.
- LUWID=USIBMSC.SCSIM8HA 0CB3A2C1AC44 0001 is the APPC luwid identifier and allows a chain to the **`d appc,tp,all`** command output.
- CONV CORRELATOR=BC020CB3A2B54204 is an ID to chain the inbound and outbound information together. If the outbound program, within the same urid, starts multiple inbound transactions on the same system, it becomes very useful.

> **Tip:** Because there is no direct correlation between the two (or more) RRS urids (inbound and outbound), you have to collect the conv correlator from both sides and find out the twins. Now you have the two (or more) RRS urids to look for in the IMS or RRS logs, or both.

For the same conversation we can have a different view looking from the RRS panels. The output is shown in Example 4-7 and Example 4-8.

Example 4-7 Output from the RRS panels at outbound system

```
RRS Resource Manager List         Row 1 to 18 of 18
 Command ===>                                           Scroll ===> CSR

 Commands: v-View Details u-View URs r-Remove Interest

 S   RM Name                         State            System   Logging Group
 u   ATB.USIBMSC.SCSIM8HA.IBM        Run              SC61     WTSCPLX1
     BBO.CL611.CLU611.WS611.IBM      Reset            SC61     WTSCPLX1
     BSS00.SC61.TIMING01.IBM         Reset            SC61     WTSCPLX1
     BSS00.SC61.TIOASR2A.IBM         Reset            SC61     WTSCPLX1
     BSS00.SC61.TIOTRADA.IBM         Reset            SC61     WTSCPLX1
     BSS00.SC61.TIOTREDA.IBM         Reset            SC61     WTSCPLX1
     BSS00.SC61.TISMGT01.IBM         Reset            SC61     WTSCPLX1
     BSS00.SC61.TITFRP01.IBM         Reset            SC61     WTSCPLX1
     CSQ.RRSATF.IBM.WMQX             Reset            SC61     WTSCPLX1
     DFHRXDM.SCSCBUD1.IBM            Reset            SC61     WTSCPLX1
     DFHRXDM.SCSCPA8K.IBM            Run              SC61     WTSCPLX1
     DSN.RRSATF.IBM.DB8D             Run              SC61     WTSCPLX1
     DSN.RRSATF.IBM.DB8K             Run              SC61     WTSCPLX1
     DSN.RRSATF.IBM.D7K1             Reset            SC61     WTSCPLX1
     DSN.RRSPAS.IBM.DB8D             Run              SC61     WTSCPLX1
     DSN.RRSPAS.IBM.DB8K             Run              SC61     WTSCPLX1
     DSN.RRSPAS.IBM.D7K1             Reset            SC61     WTSCPLX1
     IMS.IMSH____V081.STL.SANJOSE.IBM Run             SC61     WTSCPLX1
--------------------------------------------------------------------------------
                           RRS Unit of Recovery List          Row 1 to 1 of 1
 Command ===>                                           Scroll ===> CSR

 Commands: v-View Details c-Commit b-Backout r-Remove Interest f-View UR Family

 S   UR Identifier                    System    Logging Group
                                         State         Type  Comments
 v BC020CB37E6C4000000000CA01050000  SC61      WTSCPLX1
                                         InFlight      Prot
--------------------------------------------------------------------------------
                          RRS Unit of Recovery Details        Row 1 to 2 of 2
 Command ===>                                           Scroll ===> CSR

 Commands r-Remove Interest v-View URI Details

 UR identifier : BC020CB37E6C4000000000CA01050000
 Create time : 2004/10/22 22:10:24.724959      Comments :
 UR state : InFlight      UR type : Prot
 System : SC61      Logging Group : WTSCPLX1
 SURID : N/A
 Work Manager Name : SC61.IVP8HM11.0029
    Display Work IDs             Display IDs formatted
    Luwid  . : Present
    Eid  . . : Not Present
    Xid  . . : Present
 Expressions of Interest:
 S   RM Name                          Type  Role
     IMS.IMSH____V081.STL.SANJOSE.IBM Unpr  Participant
     ATB.USIBMSC.SCSIM8HA.IBM         Prot  Participant
```

Example 4-7 shows how RRS, for the urid BC020CB37E6C4000000000CA01050000, is registered as syncpoint coordinator for the TP running on the work manager SC61.IVP8HM11.0029 (in our example the message region for the outbound IMS TP) within the resource manager ATB.USIBMSC.SCSIM8HA.IBM (in our example APPC for the IMS outbound LU).

> **Tip:** APPC creates and registers a Resource Manager to RRS for every LU with syncpoint capability. The naming convention is:
>
> ATB.network-qualified-network-name.IBM
>
> In our test, for example, the name is ATB.USIBMSC.SCSIM8IA.IBM.

Example 4-8 Output from the RRS panels at inbound system

```
RRS Resource Manager List          Row 1 to 16 of 16
 Command ===>                                                Scroll ===> CSR

 Commands: v-View Details u-View URs r-Remove Interest

 S   RM Name                         State            System   Logging Group
 u   ATB.USIBMSC.SCSIM8IA.IBM        Run              SC62     WTSCPLX1
     BBO.CL622.CLU622.WS622.IBM      Reset            SC62     WTSCPLX1
     BSS00.SC62.TIMING02.IBM         Reset            SC62     WTSCPLX1
     BSS00.SC62.TIOASR2B.IBM         Reset            SC62     WTSCPLX1
     BSS00.SC62.TIOTRADB.IBM         Reset            SC62     WTSCPLX1
     BSS00.SC62.TIOTREDB.IBM         Reset            SC62     WTSCPLX1
     BSS00.SC62.TISMGT02.IBM         Reset            SC62     WTSCPLX1
     BSS00.SC62.TITFRP02.IBM         Reset            SC62     WTSCPLX1
     DFHRXDM.SCSCBUD2.IBM            Reset            SC62     WTSCPLX1
     DSN.RRSATF.IBM.DB7I             Reset            SC62     WTSCPLX1
     DSN.RRSATF.IBM.DB8L             Run              SC62     WTSCPLX1
     DSN.RRSATF.IBM.D7K2             Reset            SC62     WTSCPLX1
     DSN.RRSPAS.IBM.DB7I             Reset            SC62     WTSCPLX1
     DSN.RRSPAS.IBM.DB8L             Run              SC62     WTSCPLX1
     DSN.RRSPAS.IBM.D7K2             Reset            SC62     WTSCPLX1
     IMS.IMSI____V081.STL.SANJOSE.IBM Run             SC62     WTSCPLX1
--------------------------------------------------------------------------------
                          RRS Unit of Recovery List           Row 1 to 1 of 1
 Command ===>                                                Scroll ===> CSR

 Commands: v-View Details c-Commit b-Backout r-Remove Interest f-View UR Family

 S   UR Identifier                    System    Logging Group
                                         State         Type  Comments
 v   BC020CB37E6D0000000000C2010E0000  SC62      WTSCPLX1
                                         InFlight      Prot
--------------------------------------------------------------------------------
                         RRS Unit of Recovery Details         Row 1 to 1 of 1
 Command ===>                                                Scroll ===> CSR

 Commands r-Remove Interest v-View URI Details

 UR identifier : BC020CB37E6D0000000000C2010E0000
 Create time : 2004/10/22 22:10:24.985354      Comments :
 UR state : InFlight      UR type : Prot
 System : SC62      Logging Group : WTSCPLX1
 SURID : N/A
 Work Manager Name : IMS.IMSI____V081.STL.SANJOSE.IBM
    Display Work IDs              Display IDs formatted
    Luwid  . : Present
    Eid  . . : Not Present
    Xid  . . : Present
```

```
Expressions of Interest:
S   RM Name                          Type  Role
    ATB.USIBMSC.SCSIM8IA.IBM         Prot  Participant
```

Example 4-8 shows how RRS, for the urid BC020CB37E6D0000000000C2010E0000, is registered as syncpoint coordinator for the TP scheduled on the work manager IMS.IMSI___V081.STL.SANJOSE.IBM (in our example, the inbound IMS) within the resource manager ATB.USIBMSC.SCSIM8IA.IBM (in our example, APPC for the IMS inbound LU). At the moment of the command the inbound TP was not already scheduled. When the TP runs, IMS.IMSI___V081.STL.SANJOSE.IBM will be registered as resource manager too. For a different test, in Example 4-9 you can look at the relative output.

Example 4-9 Output from the RRS panels at inbound system. TP is already running on IMSI

```
                      RRS Unit of Recovery Details         Row 1 to 2 of 2
Command ===>                                                Scroll ===> CSR

Commands r-Remove Interest v-View URI Details

UR identifier : BC05E2C07E6D000000000108010E0000
Create time : 2004/10/25 23:24:02.010658      Comments :
UR state : InFlight      UR type : Prot
System : SC62      Logging Group : WTSCPLX1
SURID : N/A
Work Manager Name : IMS.IMSI____V081.STL.SANJOSE.IBM
   Display Work IDs           /  Display IDs formatted
   Luwid  . : Present
   Eid  . . : Not Present
   Xid  . . : Present
Expressions of Interest:
S   RM Name                           Type  Role
    ATB.USIBMSC.SCSIM8IA.IBM          Prot  Participant
    IMS.IMSI____V081.STL.SANJOSE.IBM  Prot  Participant
```

Important: Because every inbound TP is, from a resource manager point of view, a separate unit of work, you can experience a time-out or deadlock condition if during the same outbound RRS unit of recovery, your inbound TPs access concurrently and update the same resources. Remember this behavior when you design your application.

4.2 How to handle failures

Because of the number of products involved in a protected conversation environment, we have many points where a failure can occur. One failure reason of interest is the abend of one of the control address spaces: APPC, RRS, CICS, Logger, IMS, DB2, and so forth. We are less interested in a single IMS, CICS, or DB2 (stored procedure) transaction abend because the recovery is guaranteed by the relative control address space. In the case of application failure, the control address space will signal the event with a specific code or main abend. In the case of the control address space going down, usually restarting the failed address spaces fixes everything. However, sometimes we can face unpredictable situations where we, as system programmers, become responsible for the data integrity of the transaction. APPC notifies us of these kinds of problems by issuing ATB2xx messages. Because APPC is not a database, it doesn’t give any direct command to handle transaction failure. We have to use RRS panels to issue the proper commands. During the startup, or at the first allocate request (ATBALLC) between two LUs, APPC does an early cross-checking between the two LUs. This is called “exchange log names.” During this phase the two partners discover if the other is in the same state of the last contact. If it is, you can see the message shown in Example 4-10.

Example 4-10 Syslog for a successful exchange log name between two APPC LUs

```
ATB207I EXCHANGE LOG NAME PROCESSING HAS COMPLETED SUCCESSFULLY
BETWEEN LOCAL LU USIBMSC.SCSIM8HA AND PARTNER LU
USIBMSC.SCSIM8I$.
LOCAL LOG: ATR.BA04D30D06ADBE08.IBM
PARTNER LOG: ATR.BA04D30D06ADBE08.IBM

ATB207I EXCHANGE LOG NAME PROCESSING HAS COMPLETED SUCCESSFULLY
BETWEEN LOCAL LU USIBMSC.SCSIM8I$ AND PARTNER LU
USIBMSC.SCSIM8HA.
LOCAL LOG: ATR.BA04D30D06ADBE08.IBM
PARTNER LOG: ATR.BA04D30D06ADBE08.IBM
```

There's a message for each system owning the LU. If there is a failure, depending on the reason, you can have one or more of the messages shown in Example 4-11.

Example 4-11 Syslog for unsuccessful exchange log name between APPC LUs

```
ATB227I LOCAL LU USIBMSC.SCSIM8IA IS WARM STARTING AS A RESOURCE
MANAGER WITH RRS/MVS.
LOCAL LOG: ATR.BA04D30D06ADBE08.IBM
ATB225I LOGICAL UNIT SCSIM8IA IS ACTIVE, BUT WILL REJECT ALL 507
PROTECTED CONVERSATIONS BECAUSE OF A FAILURE RETURN CODE FROM THE
ATRIBRS SERVICE.  RETURN CODE IS 00000FFF.

ATB225I LOGICAL UNIT SCSIM8IA IS ACTIVE, BUT WILL REJECT ALL 558
PROTECTED CONVERSATIONS BECAUSE OF A FAILURE RETURN CODE FROM THE
ATRIBRS SERVICE.  RETURN CODE IS 00000FFF.

+ATB70042I APPC/MVS cannot schedule allocate request.  LU SCSIM8IA
can not process syncpt conversations.
```

In Example 4-11, the ATB225I message shows an APPC problem encountered during an RRS call, while ATB70042I is a message that APPC sends only to the TP program. Basically you can have three level of problems, shown in increasing level of severity:

- Unit of recovery problems, single or multiple
- LUs warm/cold or name mismatch problems
- RRS or Logger problems

4.2.1 Solving unit of recovery problems

This is the lowest severity problem. Some information is available in RRS's logs because of a missed or failed resynch process after a UOR failure. You have to remove the interest of APPC for this UOR using the RRS panels. Example 4-12 shows a sample of RRS UORs.

Example 4-12 Sample RRS UOR list for an APPC LU and a detail

```
                        RRS Unit of Recovery List            Row 1 to 3 of 3
Command ===>                                                Scroll ===> CSR
Commands: v-View Details c-Commit b-Backout r-Remove Interest f-View UR Family

S   UR Identifier                    System    Logging Group
                                        State         Type  Comments
    BC0AC5E07E6CC00000000006010E0000  SC62      WTSCPLX1
                                        InFlight      Prot
    BC0AC5E07E6CC37400000008010E0000  SC62      WTSCPLX1
                                        InFlight      Prot
    BC0AC5E07E6CC6E800000006010E0000  SC62      WTSCPLX1
                                        InFlight      Prot
--------------------------------------------------------------------------------
                        RRS Unit of Recovery Details         Row 1 to 2 of 2
Command ===>                                                Scroll ===> CSR
Commands r-Remove Interest v-View URI Details

UR identifier : BC0AC5E07E6CC00000000006010E0000
Create time : 2004/10/29 20:41:28.243319      Comments :
UR state : InFlight      UR type : Prot
System : SC62      Logging Group : WTSCPLX1
SURID : N/A
Work Manager Name : IMS.IMSI____V081.STL.SANJOSE.IBM
   Display Work IDs          /  Display IDs formatted
   Luwid  . : Present
   Eid  . . : Not Present
   Xid  . . : Present
Expressions of Interest:
S   RM Name                          Type  Role
    ATB.USIBMSC.SCSIM8I$.IBM         Prot  Participant
    IMS.IMSI____V081.STL.SANJOSE.IBM  Prot  Participant
--------------------------------------------------------------------------------
                     RRS Unit of Recovery Work IDs
UR identifier : BC0AC5E07E6CC00000000006010E0000
                                                              More:     +
Logical Unit of Work Identifier (LUWID):
 USIBMSC.SCSIM8HA C5E06397CB05 0001

 NetID.LuName : USIBMSC.SCSIM8HA
 TP Instance  : C5E06397CB05
 SeqNum . . . : 0001

Enterprise Identifier (EID)
 TID  :               (decimal)
 GTID :

X/Open Transactions Identifier (XID)
 Format ID : 003654612722  (decimal)
           D9D4F6F2  (hexadecimal)
 GTRID : 00-0F F0F0F2F0 F6F4F1C3 F7C9C2D4 F0F2F0F0 |0020641C7IBM0200|
         10-1F F0F0F0F0 F0F1F0C5 C3C210E4 E2C9C2D4 |0000010ECB.USIBM|
         20-2F E2C34BE2 C3E2C9D4 F8C8C1C5 E06397CB |SC.SCSIM8HAE\.p.|
         30-33 05000100                            |....            |

 BQUAL : 00-0F E4E2C9C2 D4E2C34B E2C3E2C9 D4F8C8C1 |USIBMSC.SCSIM8HA|
         10-1F 00E4E2C9 C2D4E2C3 4BE2C3E2 C9D4F8C9 |.USIBMSC.SCSIM8I|
         20-21 5B00                                |$.              |
```

The update actions you can do against the UOR depend on the state of the UOR itself. You can back out or commit any "in doubt" UOR, while you can only remove interest for the other states. Removing interest is possible only if the resource manager of the UOR is not active. In terms of APPC, the LU must be inactive. You can inactivate the LU with the VTAM command **`v net,id=LUname,inact,f`**. After the LU activation, with the command **`v net,id=LUname,act`**, it will restart in warm mode without any memory of the UOR you removed. It is your responsibility to do what is needed on every subsystem involved in the protected conversation to solve the UOR's status.

> **Tip:** Before acting to remove the UOR interest, go through the RRS panels to discover which partners are involved in the conversation. Looking at Example 4-12, LUWID information on the "RRS Unit of Recovery Work IDs" is very useful. Of course a thorough knowledge of the application flow is a plus.

4.2.2 Solving LUs warm/cold or name mismatch problems

If, during an exchange log name transaction, the local LU or partner LU detects a warm/cold log status mismatch, APPC/MVS issues operator message ATB210E. Messages ATB70052I and ATB80129I may be returned to the TP. The log status mismatch may be caused by:

- The wrong level of log data at the local or partner LU
- A cold log start at one of the partners

If the cold log status is valid for one of the logical units, and if the warm partner is an APPC/MVS managed logical unit of work, then to resolve the warm/cold mismatch, take one of the following actions against the warm partner (listed in order of increasing potential disruption):

- Restart the warm APPC/MVS LU after removing all interests for the APPC/MVS LU using the RRS ISPF panels.
- Delete the LU from the APPC/MVS configuration by issuing a SET command for a parmlib member with an LUDEL statement for the LU.
- Remove all interests for the cold status partner using the RRS ISPF panels as described in 4.2.1, "Solving unit of recovery problems" on page 62.
- Add the LU to the APPC/MVS configuration by issuing a SET command for a parmlib member with an LUADD statement for the LU. The APPC/MVS LU will now restart; however, this will be without incomplete logical units of work.
- Attempt to initiate a protected conversation between the affected LUs.

If a CICS is involved in the mismatch problem, you have to solve the problem using the CEMT transaction against UOR and CONNection. CICS (we are interested only in APPC communications to non-CICS subsystems and not to the classic DPL between CICS subsystems) maintains for itself the log name information. Example 4-13 shows a sample syslog output for a mix of successful and unsuccessful exchange log names.

Example 4-13 Syslog sample for a mix condition on an exchange log name.

```
ATB217I EXCHANGE LOG NAME PROCESSING INITIATED BY LU USIBMSC.SCSIM8IA
787
WITH LU USIBMSC.SCSCPA8K HAS FAILED ON 11/02/2004 AT 17:48:57.
THE LOCAL LU WILL TRY AGAIN TO COMPLETE A SUCCESSFUL EXCHANGE LOG NAME
WITH LU USIBMSC.SCSCPA8K.SOME LOGICAL UNITS OF WORK MIGHT NOT
BE AUTOMATICALLY RESOLVED BY RESYNCHRONIZATION AND NO NEW PROTECTED
CONVERSATIONS MAY BE ALLOCATED BETWEEN THE TWO LOGICAL UNITS
UNTIL AN EXCHANGE LOG NAME TRANSACTION COMPLETES.
```

```
ATB207I EXCHANGE LOG NAME PROCESSING HAS COMPLETED SUCCESSFULLY 788
BETWEEN LOCAL LU USIBMSC.SCSIM8IA AND PARTNER LU
USIBMSC.SCSIM8HA.
LOCAL LOG: ATR.BC09EDFF7FEA6AE5.IBM
PARTNER LOG: ATR.BC09EDFF7FEA6AE5.IBM
```

Message ATB217I shows an unsuccessful exchange log name between LUs SCSIM8IA (IMS) and SCSCPA8K (CICS), while, at the same time, message ATB207I shows a successful exchange log name between the LUs SCSIM8IA (same IMS of the failure) and SCSIM8HA (IMS). Then, even if the SCSIM8IA LU has syncpoint capability, due to a previous failure there are synchronization problems that abort the process and no protected conversations can be started between the two partners. In CICS, you can observe the connection in status Xno.

After solving the UORs status in CICS, and of course on RRS panels for the APPC-managed LU, you have to set the connection in RELEASE and OUTSERVICE status and perform the set NOTPENDING and NORECOVDATA status. Now you can set the connection in INSERVICE and ACQUIRED status and the connection should be Xok. Example 4-14 and Example 4-15 display the output of the CEMT transaction to reset a log name mismatch and the subsequent status query.

Example 4-14 Use of CICS CEMT transaction to reset logname

```
S CONN(IM8I) NOTP NOREC
  STATUS:  RESULTS - OVERTYPE TO MODIFY
   Con(IM8I) Net(SCSIM8IA)     Out Rel Vta Appc                NORMAL
                                           Nqn(USIBMSC.SCSIM8IA )


                                                    SYSID=PA8K APPLID=SCSCPA8K
   RESPONSE: NORMAL                                TIME:  19.33.44  DATE: 10.29.04
 PF 1 HELP       3 END       5 VAR        7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

Example 4-15 CICS APPC connection in the right status

```
I CONN(IM8I)
  STATUS:  RESULTS - OVERTYPE TO MODIFY
   Con(IM8I) Net(SCSIM8IA)     Ins Acq Vta Appc     Xok Nrs
          Una                                 Nqn(USIBMSC.SCSIM8IA )


                                                    SYSID=PA8K APPLID=SCSCPA8K
   RESPONSE: NORMAL                                TIME:  19.37.22  DATE: 10.29.04
 PF 1 HELP       3 END       5 VAR        7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

If these steps don’t solve the problem, you can consider the option of deleting and redefining the APPC/MVS LU log stream.

Attention: This will erase APPC/MVS's knowledge of all partners' log information and syncpoint capabilities, not just the cold status partner affected by the problem. Use it only if really necessary.

Tip: Because deleting the APPC log stream affects all the LUs, you can consider this work-around:

If your subsystem is an IMS, for example, you can change the LU name for IMS APPC LU defining a new one on the APPC configuration and deleting the previous one. This could be much more useful in the situation in which you have to cold start RRS to resolve the LU mismatch. You can use the work-around to make your IMS work up to the time of the cold start. You have to consider, too, that you have to modify all the APPC sideinfo information, if you use this kind of definition to point to the inbound system.

4.2.3 Solving RRS or System Logger problems

If all the steps described in 4.2.1, "Solving unit of recovery problems" on page 62 and 4.2.2, "Solving LUs warm/cold or name mismatch problems" on page 64 didn't solve the problem, or if your system was affected by a problem on RRS or System Logger, like abends and so on, probably you are in a no return condition in which the only solution is a cold restart of RRS.

Attention: RRS cold start affects all the users of this service—IMS, CICS, DB2, and so forth. Because you must stop all the subsystems, or at least the functions that need RRS, it is a very painful situation. Take this action only if the previous actions do not solve the problem. You should probably enlist the assistance of your local IBM representative to gather all the necessary documentation before acting.

5

Sample scenario: IMS to IMS

This chapter describes the IMS runtime environments, the IMS PL/I application programs, and the failure scenarios we executed to test APPC protected conversations.

5.1 Description

For IMS-to-IMS APPC protected conversations we wanted to test the interaction between two IMS application programs running in two different IMS systems. We wanted to verify actions taken for normal processing and for failure scenarios. IMS requires APPC/MVS and MVS Recoverable Resource Management Services to enable APPC protected conversations.

APPC/IMS is an APPC/MVS scheduler that supports implicit and explicit APPC application programming. An implicit application program uses IMS DL/I message calls for the communication interface and APPC/IMS provides the interaction with APPC. This means that existing IMS applications are able to participate in an APPC environment without modification.

An explicit application uses APPC calls for direct interaction with an APPC partner program. IMS supports the following explicit application programs:

- Modified Standard
- CPI-C Driven

The Modified Standard application program uses a combination of IMS DL/I message calls and the APPC calls to communicate directly with APPC/MVS as shown in Figure 5-1.

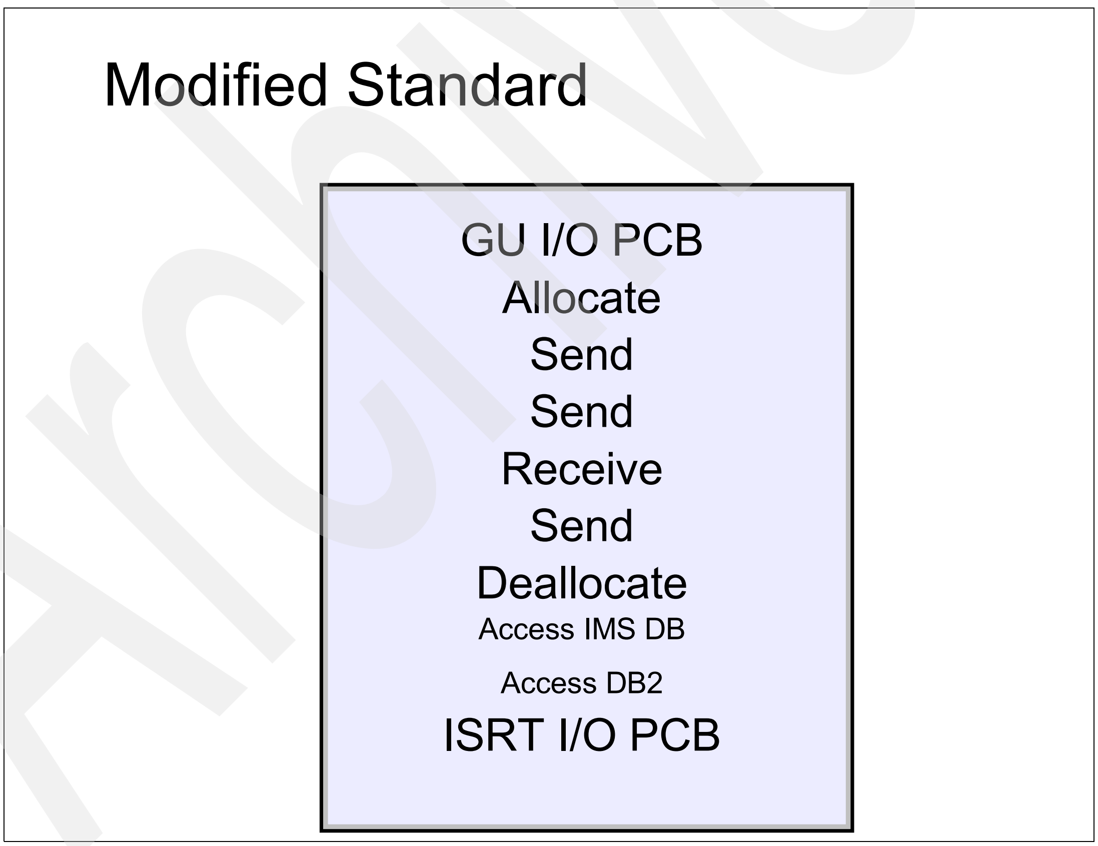


Figure 5-1 Sample of modified standard application program

An IMS Modified Standard application program can initiate an APPC Protected Conversation; however, APPC/IMS has to initiate the coordinated syncpoint process. When the IMS Modified Standard application program Deallocates the protected conversation, the APPC conversation is in a Defer-Deallocate state. The IMS syncpoint manager recognizes that APPC/MVS has expressed interest in the Context and Unit of Recovery associated with the

IMS Modified Standard application program. At the start of the IMS syncpoint process IMS will issue the ATRCMIT call to RRS to initiate the coordinated syncpoint process. The Partner Program will be notified to issue a Commit when it Receives Take_Commit_Deallocate.

RRS is the syncpoint coordinator that drives Two-Phase commit flows to IMS and to APPC/MVS. APPC/MVS provides the Communication Resource Manager support to flow Two-Phase commit protocols using the protected conversation as shown in Figure 5-2.

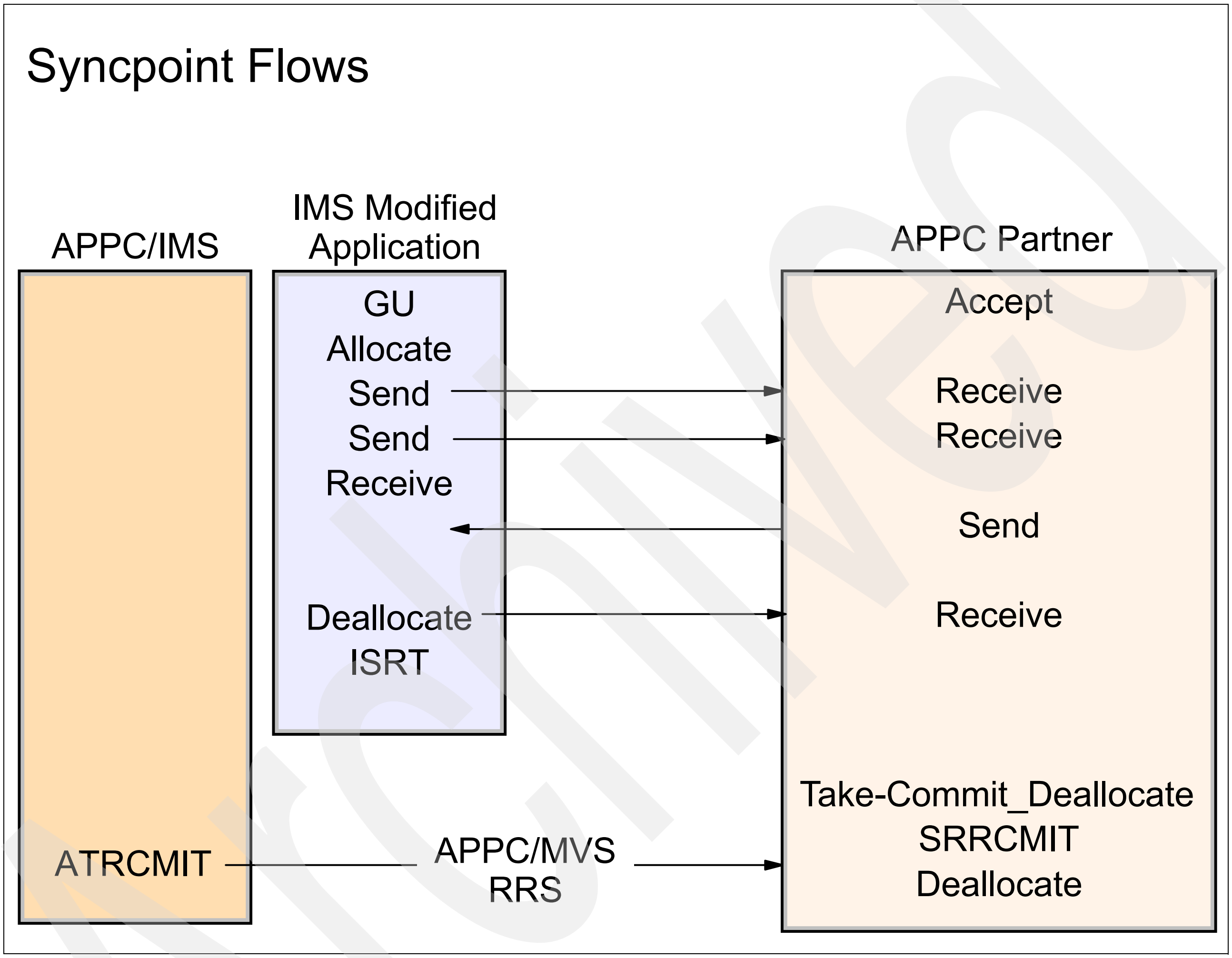


Figure 5-2 Syncpoint flow

IMS only schedules a CPI-C Driven application program and the application program has to issue the X/OPEN CPI-C calls or the APPC/MVS-specific API calls to communicate with the APPC partner program, as illustrated in Figure 5-3 on page 70.

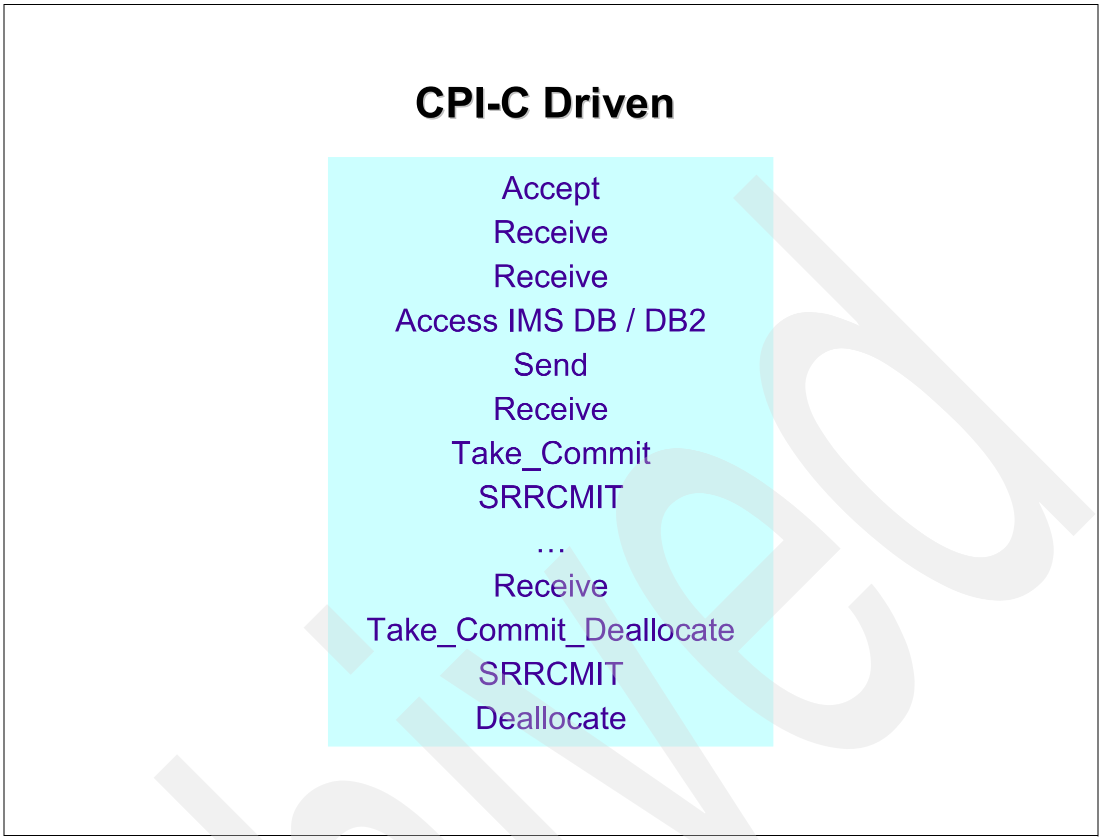


Figure 5-3 CPI-C Driver application flow

An IMS CPI-C Driven application program uses SRRCMIT to participate in an APPC protected conversation coordinated syncpoint process. When the IMS CPI-C Driven application program receives Take_Commit on a Receive verb, it uses SRRCMIT to activate its syncpoint. In the case of Take_Commit_Deallocate the IMS CPI-C Driven application program issues the SRRCMIT call and Deallocates the protected conversation.

Our sample IMS PL/I application programs consisted of the following:

IMSH	IMSI
IMS1TR1 Modified	IMS2IMP Implicit
IMS1TR2 Modified	IMS2EXP Explicit

> **Note:** To re-create the environment we used:
>
> Refer to Appendix A, “Installation definitions for Protected Conversation exploiters” on page 113 for detailed installation instructions required to set up the sample systems used throughout this book.
>
> Refer to Appendix B, “APPC exploiter sample source code” on page 155 for the source code used in our examples.

The flow for Modified to Implicit

The IMS1TR1 transaction running in IMSH uses an APPC protected conversation to schedule the IMS2IMP transaction using APPC/IMS Implicit mode support in IMSI. The input message for IMS1TR1 specifies the manner in which both transactions complete processing.

The input message format for IMS1TR1 is:

```
LLZZIMS1TR1 action type text comment appc option
```

The variables are defined as follows:

- *action type* can be:
  - NULLS - IMS1TR1 and IMS1IMP complete processing - results in Two-Phase Commit
  - LROLB - IMS1TR1 issues IMS ROLB call - results in IMS2IMP U711-1E
  - LROLL - IMS1TR1 issues IMS ROLL call - results in IMS1TR1 U778, IMS2IMP U711-1E
  - LABND - IMS1TR1 ABENDS with U3333 and IMS2IMP U711-1E
  - PROLB - IMS2IMP issues IMS ROLB call results in U711-20 and IMS1TR1 U711-14
  - PROLL - IMS2IMP issues IMS ROLL call results in U778 abend and IMS1TR1 U711-14
  - PABND - IMS2IMP ABENDS U3333 results in IMS1TR1 U711-14
- *text comment* is not processed by the applications but can be used to provide information about the transaction processing.
- *appc option* can be set to NOAPPC. IMS1TR1 will not allocate a protected conversation to schedule IMS2IMP.

The flow for Modified to Explicit

The IMS1TR2 transaction running in IMSH uses an APPC protected conversation to schedule the IMS2EXP transaction using APPC/IMS Explicit mode support in IMSI. The input message for IMS1TR2 specifies the manner in which both transactions complete processing.

The input message format for IMS1TR2 is:

```
LLZZIMS1TR2 action type text comment appc option
```

The variables are defined as follows:

- *action type* can be:
  - NULLS - IMS1TR2 and IMS2EXP complete processing - results in Two-Phase Commit
  - LROLB - IMS1TR2 issues IMS ROLB call - results in U711-1E
  - LROLL - IMS1TR2 issues IMS ROLL call - results in U778
  - LABND - IMS1TR2 ABENDS with U3333
  - PROLB - IMS2EXP issues IMS ROLB call results in IMS1TR2 U711-14
  - PROLL - IMS2EXP issues IMS ROLL call results in IMS1TR2 U711-14
  - PCMIT - IMS2EXP issues SRRCMIT call
  - PBACK - IMS2EXP issues SRRBACK call results in IMS1TR2 U711-14
  - PABND - IMS2EXP ABENDS results in IMS1TR2 U711-14
- *text comment* is not processed by the applications but can be used to provide information about the transaction processing.

- *appc option* can be set to NOAPPC. IMS1TR2 will not allocate a protected conversation to schedule IMS2EXP.

For both Implicit to Implicit and Implicit to Explicit

Lnnnn function is used to create failure scenarios in the originator of the APPC protected conversation to verify actions propagated to the Partner Program.

Pnnn function is used to create failure scenarios in the Partner Program to verify actions propagated to the originator of the APPC protected conversation.

These failure scenarios resulted in IMS, APPC/MVS, and RRS recognizing the work was In-Flight or In-Prepare status and could be backed out.

5.1.1 Additional scenarios

These scenarios are mainly targeted for environments running IMS V8 and earlier versions of the product.

We modified the PL/I application programs to test the start of multiple CPI-C Driven applications or IMS Implicit applications. This gave us the ability to test multiple transactions that could participate in a single unit of work and to verify that IMS could support a Modified Application Program allocating more than one protected conversation. See Figure 5-4.

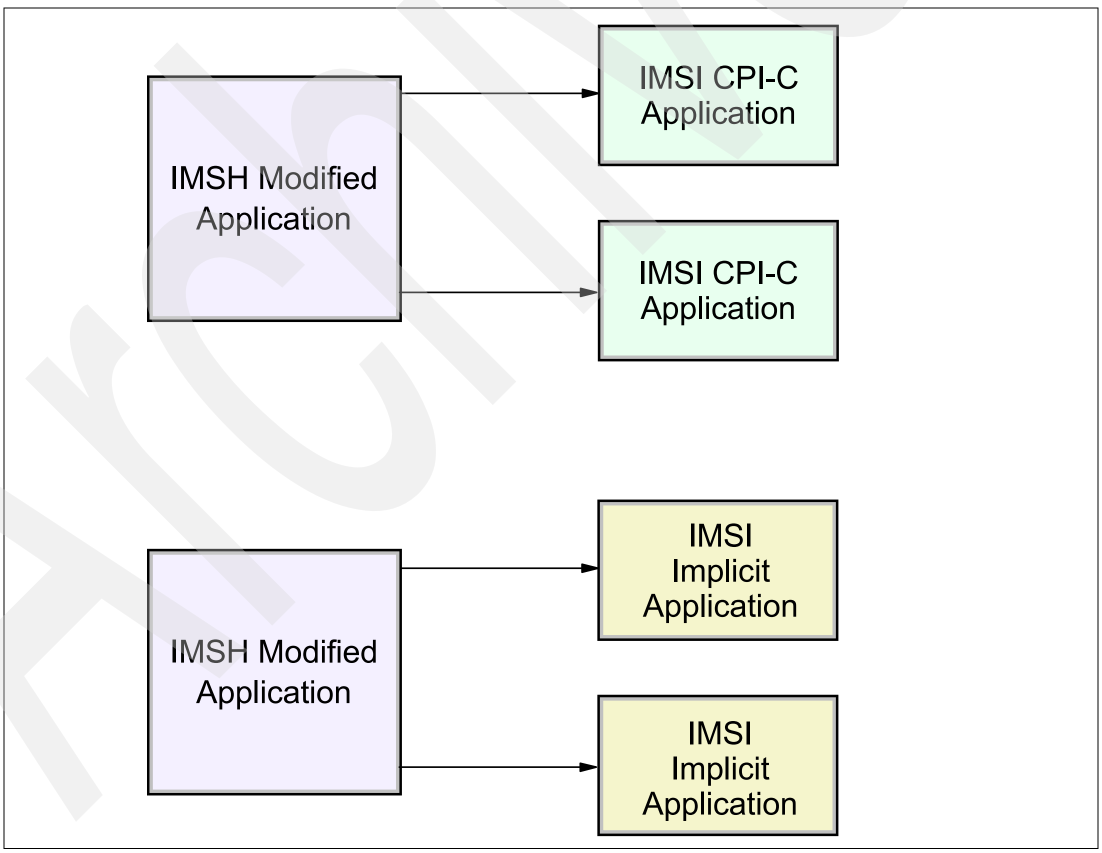


Figure 5-4 Application flows for the scenarios

Another modification we made was to create a chain of transactions to participate in a unit of work. In this scenario we used a CICS application to initiate a protected conversation to an IMS CPI-C Driven application program. The IMS CPI-C Driven program allocates a protected conversation to an IMS Implicit application program. This test validated that a daisy chain application model can participate in a coordinated unit of work.

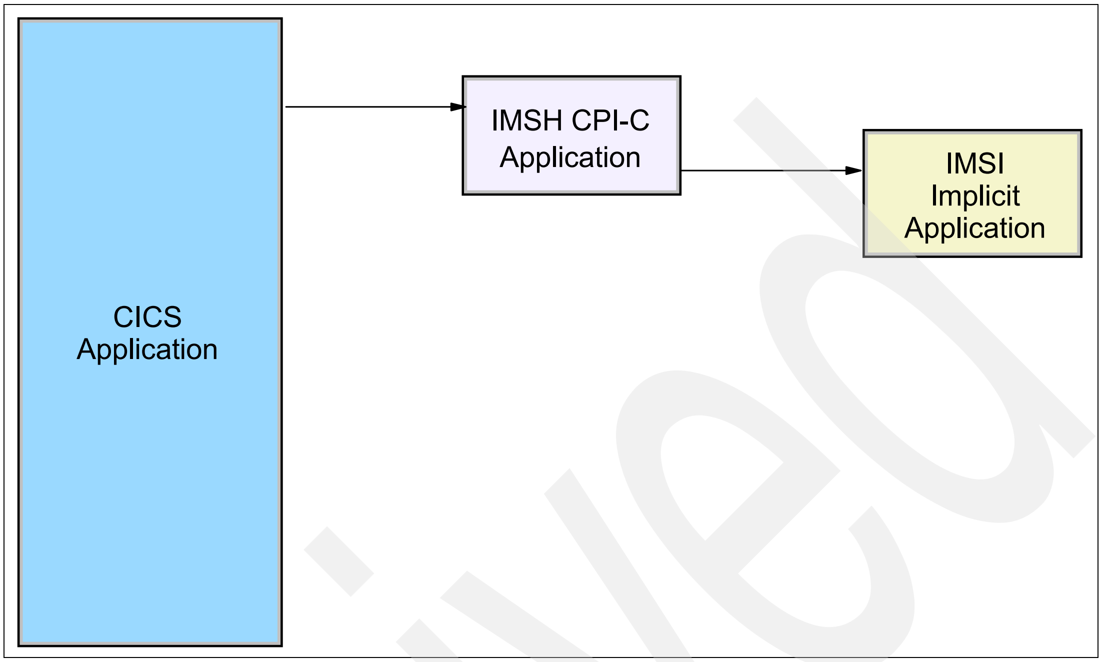


Figure 5-5 Application flow for a chain transaction scenario

These additional scenarios exposed a logic error in the IMS CPI-C Driven application program. The program was not able to support multiple Send/Receive processing. The logic error prevented multiple CPI-C transactions from participating in the unit of work. We modified the application to support multiple scheduling and to recognize a Take_Commit when in Receive Status.

> **Note:** A simple rule of thumb we observed was to have the application program that initiates the protected conversation flow also initiate the syncpoint process.

5.2 How to manage and relate the pieces together

The following discussion documents a case where IMS recognizes a Unit of Work is In-Doubt and relates APPC LUWID with UR information managed by IMS and RRS.

The ATB204I message indicates resynchronization that resulted in a back out. The DFS0693I message from IMS identifies the RRS-URID and associates it with the IMS Token. Figure 5-6 on page 74 shows the ATB204I message, the DFS0693I message, the output from the **/DIS UOR** IMS command and the output from the RRS UR panel display.

```
ATB204I LOGICAL UNIT OF WORK USIBMSC.SC38TC91 174430E524ED 0001 782
WITH CONVERSATION CORRELATOR 60C1C1D940404040 IS BACKED OUT
AT LU USIBMSC.SCSCPA8K BECAUSE OF RESYNCHRONIZATION BETWEEN
LU USIBMSC.SCSIM8IA AND LU USIBMSC.SCSCPA8K.

DFS0693I RIS ESTABLISHED FOR PSB IMS2IMP ,  783
PRTKN=00010040, TOKEN=IMSI    000000AF00000000,
RRS-URID=BC1744337E6E4374000000F2010E0000 IMSI

DFS554A IVP8IM13 00002 REGION   IMS2IMP (1) IMS2IMP  000,0711
        LUNAME:SCSCPA8K IMSI
/DIS UOR

ST  P-TOKEN  PSBNAME        RRS-URID                              IMS-RECTOKN

RI   00010040    IMS2IMP    BC1744337E6E4374000000F2010E0000     000000AF00000000

LUWID=10USIBMSC.SC38TC91174430E524ED0001

RRS output

URID=BC1744337E6E4374000000F2010E0000 JOBNAME=IVP8IM11 USERID=STC
PARENT URID=00000000000000000000000000000000
SURID=N/A
WORK MANAGER NAME=IMS.IMSI____V081.STL.SANJOSE.IBM
SYNCPOINT=Commit  RETURN CODE=0000012C
START=2004/11/08 19:12:31.654862 COMPLETE=2004/11/08 19:13:05.452858
EXITFLAGS=00000000
LUWID=USIBMSC.SC38TC91 174430E524ED 0001
```

Figure 5-6 Syslog with ATB204I message

Figure 5-7 on page 75 shows the IMS Log Records associated by the IMS token produced by running the IMS DFSERA10 Utility using the DFSERA70 exit. Example 5-1 shows the control cards used to find the log records.

Example 5-1 Sample control card to locate the IMS log records

```
CONTROL  CNTL  STOPAFT=EOF
OPTION   PRINT EXITR=DFSERA70,PARM=(XFMT=N,TOKEN=X'000000AF00000000')
END
```

```
08 RECORD – IMS2IMP application program scheduled
 00000000 00700000 0800C9D4  E2F2C9D4 D7400100   00000101 00000000
 00000020 0003C9D4 E2C94040  4040 000000AF0000000

56 RECORD - IMS external subsystem support recovery log record
 00000000 005C0000 56070000  C9D4E2C9 40404040   00000003 C9D4E2F2  C9D4D740 0000000
 00000020 00000000 00000000  00000000 C9D4E2C9   40404040 000000AF00000000

56 RECORD - Interest has been registered with RRS for this UOW
 00000000 01000000 56160000  C9D4E2C9 40404040   00000003 00000000
 00000020 00000000 00000000  00000000 C9D4E2C9   40404040 000000AF00000000

56 RECORD -  Phase 1 commit processing started
 00000000 00680000 56100000  C9D4E2C9 40404040   00000003 00000000
 00000020 00000000 00000000  00000000 C9D4E2C9   40404040 000000AF00000000

56 RECORD - Phase 1 commit processing ended
 00000000 01000000 56110000  C9D4E2C9 40404040   00000003 00000000
 00000020 00000000 00000000  00000000 C9D4E2C9   40404040 000000AF00000000

38 RECORD - A Protected Conversation has been put in doubt
 00000000 00700000 38040000  21728118 C9D4E2C9   40404040 000000AF0000000

56 RECORD - Recoverable in-doubt structure (RIS) created
 00000000 005C0000 56130000  C9D4E2C9 40404040   00000003 00000000
 00000020 00000000 00000000  00000000 C9D4E2C9   40404040 000000AF00000000

07 RECORD - IMS2IMP application program terminated
 00000000 015C0000 07C9D4E2  F2C9D4D7 40C9D4E2   F2C9D4D7 40010100
 00000020 C9E5D7F8 C9D4F1F1  D9C5C7C9 D6D54040   00000001 00000000
 00000100 E2C94040 4040 000000AF0000000 00000   00000000 00000000
```

Figure 5-7 IMS log records

After IMS restart the In-Doubt Unit of Work can be resolved and the RIS can be removed by IMS. Figure 5-8 shows the IMS DFS0699I message written during IMS restart to indicate the Abort processing was completed for the Unit of Work.

```
DFS0699I RESYNC ABORT  COMPLETE FOR PSB IMS2IMP ,  152
 PRTKN=00010040, TOKEN= IMSI    000000AF00000000,
 RRS-URID=BC1744337E6E4374000000F2010E0000 IMSI
```

Figure 5-8 IMS restart message to indicate UOW abort processing has completed

It should be noted that we could have used the IMS **/CHANGE UOR** command to abort the in-doubt UOR since the ATB message indicated back out processing for the LUWID.

5.3 How to handle failure scenarios

In the case where the Unit of Work is In_Flight or In_Prepare status IMS, APPC/MVS and RRS can perform the ABORT processing to back out the changes to resources.

For In-Doubt processing IMS, APPC/MVS, and RRS must work together to provide resolve in_doubt processing. What needs to be determined is should the Unit of Work be committed or aborted. If a resource manager voted no during the prepare stage then the Unit of Work needs to be aborted. However, if all resource managers voted yes during prepare phase then the work needs to be committed. The resolve in_doubt processing requires network flows;

however, if long delays occur resource locks can be held for an extended time, which impacts new work. If a manual intervention is performed on a resource a heuristic mixed in_doubt resolution can occur. This requires manual coordination across the resource managers to ensure all changes were committed or aborted. Example 5-2 shows the IMS recommended steps to gather documentation for problem analysis.

Example 5-2 IMS recommended steps to collect documentation

```
Problem Determination
A common challenge to all client/server implementations is the area of problem
determination. When a persistent error occurs, several steps can be taken to
gather information for analysis. The traces that can be taken include a VTAM
buffer trace, the APPC component trace, and the IMS LU manager trace (LUMI).
· VTAM buffer trace
The VTAM buffer trace traces inbound and outbound message buffers for a
specified LU. The trace data is gathered in a general trace facility (GTF)
external data set. To take the trace:
1. Start the VTAM trace: F
NET,TRACE,TYPE=BUF,ID=......,AMOUNT=FULL.
2. Start GTF with TRACE=USR control card: S GTF.XXX,
where XXX is a user-defined set of characters that are used as a
modifier.
3. Gather data.
4. Stop the VTAM trace: F NET,NOTRACE,TYPE=BUF,ID=......
5. Stop GTF: P GTF.XXX,
where XXX is the modifier that was used to start the GTF trace
6. View the output, using IPCS, and specify GTF USR(FEF)
· APPC component trace
The APPC component trace traces calls into and out of the APPC/MVS
address space. It provides data such as the module flow, the caller¢s
parameters, and return codes. The trace data is gathered in a buffer in
storage. To take the trace:
1. Start the trace: TRACE CT,ON,COMP=SYSAPPC.
2. Respond to the WTOR: R xx,OPTIONS=GLOBAL,END.
3. Verify that the trace is on: D TRACE,COMP=SYSAPPC.
4. Gather data.
5. Stop the trace: TRACE CT,OFF,COMP=SYSAPPC.
When the trace stops, a dump of the APPC address space is
automatically taken.
6. View the output, using IPCS command CTRACE COMP(SYSAPPC) FULL.
· LUMI
The LUMI traces the flows through the APPC/IMS modules and shows APPC
verbs and return codes. The trace data is gathered in one of three places:
the IMS external trace data set, the IMS log, or IMS storage. The actual
location depends on the IMS setup for tracing. To take the trace:
1. Start the trace: /TRACE SET ON TABLE LUMI.
2. Gather data.
3. Stop the trace: /TRACE SET OFF TABLE LUMI.
If data is being gathered in IMS storage, take a dump of IMS.
4. View the output:
- Use IPCS if the data is in the dump:
VERBX IMSDUMP,xxxIMS,¢D,A,R,FMTIMS(ALL)¢
- Use DFSERA10 if the data is in an external data set or the IMS log:
OPTION PRINT O=5,L=2,V=67FA,T=x,E=DFSERA60
```

5.3.1 When IMS is not connected to RRS

Within V6, IMS became able to process protected APPC conversation using RRS. With active RRS, every transaction was registered to RRS even if it is not necessary. If your installation was not interested in APPC protected conversation and you wished to avoid wasting CPU, the only way was to customize IMS.

Within IMS V7, a new proclib parameter, named RRS, was introduced with the meaningful values of Y/N. If you start your IMS with RRS=N, no action is done toward RRS and *IMS can't accept protected inbound conversations*. But what about the outbound conversations? If you run a modified IMS program that allocates an APPC outbound conversation using a synclevel value of syncpoint and your IMS LU has syncpoint capability, no error will be shown. *The problem is that IMS is not participating as resource manager to the UOW and this means you have data exposure.* Because IMS is not registered as RM it is not notified about failures at the inbound side. Actually, IMS can't recognize this kind of error because the APPC API calls are executed out of its own control. If you think you haven't got any problems because your IMS is always started with RRS=Y, then take a look at the syslog output shown in Example 5-3.

Example 5-3 Sample syslog output during a loss of connection between IMS and RRS

```
R 309,/STO REG 3 ABDUMP IMS2EXP
IEE600I REPLY TO 309 IS;/STO REG 3 ABDUMP IMS2EXP
IEA989I SLIP TRAP ID=X47B MATCHED.  JOBNAME=APPC    , ASID=0443.
DFS058I 17:07:06 STOP COMMAND IN PROGRESS   IMSI
ATB213I LOGICAL UNIT OF WORK USIBMSC.SCSIM8HA 22410569C401 0001 223
WITH CONVERSATION CORRELATOR BC1122410558E441
REQUIRED RESYNCHRONIZATION ON 11/03/2004 AT 17:07:06.
TO RESOLVE THE LOGICAL UNIT OF WORK,
RESYNCHRONIZATION HAS STARTED BETWEEN
LOCAL LU USIBMSC.SCSIM8IA AND PARTNER LU USIBMSC.SCSIM8HA.
310 DFS996I *IMS READY*  IMSI
ATR306I RESOURCE MANAGER ATB.USIBMSC.SCSIM8IA.IBM 224
CAUSED A OK-OUTCOME-PENDING CONDITION FOR URID =
BC1122417E6E40000000002A010E0000.
ATR169I RRS HAS UNSET EXITS FOR RESOURCE MANAGER 226
IMS.IMSI____V081.STL.SANJOSE.IBM REASON: BAD RETCODE
ATB214I THE RESYNCHRONIZATION OF LOGICAL 227
UNIT OF WORK USIBMSC.SCSIM8HA 22410569C401 0001
WITH CONVERSATION CORRELATOR BC1122410558E441
IS BEING SUSPENDED ON 11/03/2004 AT 17:07:06.
RESYNCHRONIZATION WAS STARTED BY LOCAL LU
USIBMSC.SCSIM8IA ON 11/03/2004 AT 17:07:06
FOR THE LOGICAL UNIT OF WORK.
THE LOCAL LU WILL TRY AGAIN TO RESYNCHRONIZE WITH LU USIBMSC.SCSIM8HA
TO RESOLVE THE LOGICAL UNIT OF WORK.
DFS554A IVP8IM11 00003 REGION   IMS2EXP (1) IMS2EXP  000,0474 PSB SMB
           2004/308 17:07:06 IMSI
DFS0653I PROTECTED CONVERSATION PROCESSING WITH RRS/MVS ENABLED  IMSI
```

After an **abdump** command against a CPI-C Driven transaction (it's only an example), IMS and RRS lose the connection. During the time period between the ATR169I and DFS0653I messages, IMS is not a resource manager from an RRS point of view—even if at the startup you specified RRS=Y! This time period is as long as needed to resolve the synchronization problem. All the transactions running at the moment of the ATR169I message abend within U0711 while the subsequent ones, up to the DFS0653I message, are exposed to the same problem of an IMS startup with RRS=N. The only way to protect your data from this kind of exposure is to code a cross check in your modified IMS application. One way to code the check is by the macro described in Example 5-4.

Example 5-4 Macro flow to check if your APPC/IMS application is under RRS control

1. alloc APPC conversation within a SYNCLEVEL=SYNCPOINT and retrieve the APPC Conversation ID.
2. CALL ATBEXAI with Extract_code = X'0001' and conversation id = what was retrieved in step 1. Map the response buffer within the ATBEXCOS macro. Field EXCOS_URID is the RRS Unit of Recovery identifier for protected conversation (must be > 0). For further information, refer to *z/OS V1R2.0 - V1R4.0 MVS Writing TPs for APPC/MVS,* SA22-7621.

   Another way to determine the URID is to issue the ATRRURD1 call with states_option=ATR_EXTENDED_STATES. After the call, you need to either examine the URID field or the UR_STATE field. If URID=0 or UR_STATE=ATR_IN_RESET, then an outbound protected conversation should not be allocated. Otherwise, ATRRURD1 returns a URID for input to step 5 (the ATRQUERY URINFO call).
3. Call AIBTDLI within the INQY ENVIRON parameter and find out the IMS identifier and release level fields.
4. Build the IMS RM name as IMS.*????*____V*&&&*.STL.SANJOSE.IBM where *????* is the IMSid field and *&&&* is the release level (081 for our V8).
5. Call ATRQUERY with REQUEST=URINFO and parameters RMNAME= what you built in step 4, URID= what you built in step 2, and COUNT=mycount. Remember that the use of ATRQUERY requires the application user to have ACC(READ) to the resource MVSADMIN.RRS.COMMANDS of the FACILITY class. For further information, refer to *z/OS V1R4.0 MVS Programming: Resource Recovery,* SA22-7616.
6. Check the return code. If the RC=0 from the ATRQUERY URINFO call and mycount > 0, it means IMS has expressed interest in the UR with RRS and therefore IMS will be involved in any syncpoint directives from RRS.

To code some parts of the control you must use assembler language. You might be tempted to simplifying the flow shown in Example 5-4 by checking only that IMS is a registered resource manager to RRS (ATRQUERY REQUEST=RMINFO), but you should consider the time window between your application start and the ATRQUERY call! Remember that IMS registers your UOR to RRS only if, at GU time, RRS is available. You can find a sample routine that checks for RRS availability as part of the additional material; refer to Appendix C, “Additional material” on page 161 for instructions on how to download the sample source code.

Tip: IMS, in V9, will allow you to know if your IMS UOR has expressed interest in the UR by issuing an AIBTDLI INQY ENVIRON call. There is a new char(03) field within the meaningful value of RRS. If the RRS indicator is returned on the INQY ENVIRON call, it means that IMS has expressed interest in the UR.

Example 5-5 and Example 5-6 show a test where at the abend of the inbound program a corresponding abend of the outbound program occurs because IMS is under RRS control (RRS=Y).

Example 5-5 The IMS inbound program abending

```
J E S 2  J O B  L O G  --  S Y S T E M  S C 6 2  --  N O D E  W T S C P L X 1
20.05.34 JOB07359 ---- WEDNESDAY, 03 NOV 2004 ----
20.05.34 JOB07359  IRR010I  USERID STC      IS ASSIGNED TO THIS JOB.
20.05.36 JOB07359  ICH70001I STC      LAST ACCESS AT 20:05:34 ON WEDNESDAY, NOVEMBER 3,2004
```

```
20.05.36 JOB07359  $HASP373 IVP8IM11 STARTED - INIT A    - CLASS A - SYS SC62
20.05.36 JOB07359  IEF403I IVP8IM11 - STARTED - TIME=20.05.36 - ASID=003C - SC62
20.05.36 JOB07359  +DFS0578I - READ SUCCESSFUL FOR DDNAME PROCLIB  MEMBER = DFSINTDC  IMSI
20.06.19 JOB07359  +IMS2IMI: USERID=STC
20.06.19 JOB07359  +IMS2IMI: IMS_GU Status IOPCB.STC_CODE=
20.06.19 JOB07359  +IMS2IMI: IMS_GU Input=  PROLB AAAAAAAAAAAAAAAAAAA
20.06.19 JOB07359  +IMS2IMI: IMS_GU Successful
20.06.19 JOB07359  +IMS2IMI: IMS_ROLB Entry, ROLB_FKT=ROLB
20.06.19 JOB07359  +IMS2IMI: IMS_ROLB Rollback request (PROLB) issue ROLB
20.06.19 JOB07359  IEA995I SYMPTOM DUMP OUTPUT  076
   076              USER COMPLETION CODE=0711 REASON CODE=00000020
   076             TIME=20.06.19  SEQ=04355  CPU=0000  ASID=003C
   076             PSW AT TIME OF ERROR  077C1600   9EDD58DC  ILC 2  INTC 0D
   076               NO ACTIVE MODULE FOUND
   076               NAME=UNKNOWN
   076               DATA AT PSW  1EDD58D6 - 00181610  0A0D9180  B71A4780
   076               GR 0: 84000000   1: 840002C7
   076                  2: 9F43BE20   3: 1EDD523C
   076                  4: 1F704060   5: 1F10F698
   076                  6: 1F3E3D20   7: 00000004
   076                  8: 1F410110   9: 1F704060
   076                  A: 1EFE7B40   B: 00965D80
   076                  C: 1EDD523C   D: 1F704660
   076                  E: 1F10F79C   F: 00000020
   076             END OF SYMPTOM DUMP
20.06.20 JOB07359  +DFS0578I - READ SUCCESSFUL FOR DDNAME PROCLIB  MEMBER = DFSINTDC  IMSI
```

Example 5-6 The IMS outbound program rolling back with RRS=Y

```
J E S 2  J O B  L O G  --  S Y S T E M  S C 6 1  --  N O D E  W T S C P L X 1

20.05.54 JOB07361 ---- WEDNESDAY, 03 NOV 2004 ----
20.05.54 JOB07361  IRR010I  USERID STC      IS ASSIGNED TO THIS JOB.
20.05.55 JOB07361  ICH70001I STC     LAST ACCESS AT 20:05:53 ON WEDNESDAY, NOVEMBER 3, 2004
20.05.55 JOB07361  $HASP373 IVP8HM11 STARTED - INIT A    - CLASS A - SYS SC61
20.05.55 JOB07361  IEF403I IVP8HM11 - STARTED - TIME=20.05.55 - ASID=0028 - SC61
20.05.56 JOB07361  +DFS0578I - READ SUCCESSFUL FOR DDNAME PROCLIB  MEMBER = DFSINTDC  IMSH
20.06.18 JOB07361  +IMS1PI1: PL/I IMS stub routine Begins...
20.06.18 JOB07361  +IMS1PI1: IMS_GU Entry.
20.06.18 JOB07361  +IMS1PI1: IMS_GU Status IOPCB.STC_CODE=
20.06.18 JOB07361  +IMS1PI1: IMS_GU Input= PROLB AAAAAAAAAAAAAAAAAAA
20.06.18 JOB07361  +IMS1PI1: IMS_GU Successful
20.06.18 JOB07361  +IMS1PS1: Outbound routine begins...
20.06.18 JOB07361  +IMS1PS1: Allocate Entry.
20.06.18 JOB07361  +IMS1PS1: Allocate User_ID=         , Password=         .
20.06.18 JOB07361  +IMS1PS1: Allocate Return_Code=0
20.06.18 JOB07361  +IMS1PS1: Allocate Conversation_ID=224BB3F800000333
20.06.18 JOB07361  +IMS1PS1: Allocate RC is OK
............. omissis .............
20.06.19 JOB07361  +IMS1PS1: Receive_And_Wait RCV1 Entry, Conversation_ID=224BB3F800000333
20.06.19 JOB07361  +IMS1PS1: Receive_And_Wait RCV1 Return_Code=21
20.06.19 JOB07361  +IMS1PS1: Receive_And_Wait RCV1 Receive_Length=256
20.06.19 JOB07361  +IMS1PS1: Receive_And_Wait RCV1 Receive_Buffer=
20.06.19 JOB07361  +IMS1PS1: Receive_And_Wait RCV1 Data_Received=0
20.06.19 JOB07361  +IMS1PS1: Receive_And_Wait RCV1 Request_to_Send_Received=0
20.06.19 JOB07361  +IMS1PS1: Receive_And_Wait RCV1 Status_Received is
atb_no_status_received
20.06.19 JOB07361  +IMS1PS1: Receive_And_Wait RCV1 Failed
20.06.19 JOB07361  +IMS1PS1: Error_Extract Entry, Conversation_ID=224BB3F800000333
20.06.19 JOB07361  +IMS1PS1: Error_Extract Return_Code=0
```

```
20.06.19 JOB07361  +IMS1PS1: Error_Extract Service_Name=ATBRCVW
20.06.19 JOB07361  +IMS1PS1: Error_Extract Service_Reason_Code=100
20.06.19 JOB07361  +IMS1PS1: Error_Extract: Message_Text follows:
20.06.19 JOB07361  +ATB80100I From VTAM macro APPCCMD: Primary error return code: 0030,
secondary error return code: 0000, sense
                            code: 08890000.
20.06.19 JOB07361  +IMS1PS1: Error_Extract Exit.
20.06.19 JOB07361  +IMS1PS1: Receive_And_Wait RCV1 RC is PROGRAM_ERROR_NO_TRUNC
20.06.19 JOB07361  +IMS1PS1: Outbound routine ends...
20.06.19 JOB07361  +IMS1PI1: IMS1PS1 IMS1PS1_Return_Code=21
20.06.19 JOB07361  +IMS1PI1: IMS1PS1 call Failed, DATETIME=20041103200619448
20.06.19 JOB07361  +IMS1PI1: IMS_ISRT Entry.
20.06.19 JOB07361  +IMS1PI1: IMS_ISRT Output=IMS1TR1 Completed OK,
DATETIME=20041103200619459
20.06.19 JOB07361  +IMS1PI1: IMS_ISRT Status IOPCB.STC_CODE=
20.06.19 JOB07361  +IMS1PI1: IMS_ISRT Successful
20.06.19 JOB07361  +IMS1PI1: IMS_GU Entry.
20.06.19 JOB07361  IEA995I SYMPTOM DUMP OUTPUT  350
   350              USER COMPLETION CODE=0711 REASON CODE=00000014
   350             TIME=20.06.19  SEQ=03194  CPU=0000  ASID=0028
   350             PSW AT TIME OF ERROR  077C1600   9E277408  ILC 2  INTC 0D
   350               NO ACTIVE MODULE FOUND
   350               NAME=UNKNOWN
   350               DATA AT PSW  1E277402 - 00181610  0A0DC4C6  E2D9D9E2
   350               GR 0: 84000000   1: 840002C7
   350                  2: 000000C8   3: 1E3A8810
   350                  4: 1E39D060   5: 1EE73698
   350                  6: 1EF730E0   7: 1E26F048
   350                  8: 1E26F048   9: 1EF72F70
   350                  A: 1E39D060   B: 0097CD80
   350                  C: 9E272E50   D: 1E39D6A8
   350                  E: 1E2773EE   F: 00000014
   350             END OF SYMPTOM DUMP
20.06.20 JOB07361  +DFS0578I - READ SUCCESSFUL FOR DDNAME PROCLIB  MEMBER = DFSINTDC  IMSH
```

Example 5-7 and Example 5-8 show a test where the abend of the inbound program doesn't correspond to an abend of the outbound program because outbound IMS is not under RRS control (RRS=N).

Example 5-7 The IMS inbound program abending

```
J E S 2  J O B  L O G  --  S Y S T E M  S C 6 2  --  N O D E  W T S C P L X 1

19.59.39 JOB07321 ---- WEDNESDAY, 03 NOV 2004 ----
19.59.39 JOB07321  IRR010I  USERID STC      IS ASSIGNED TO THIS JOB.
19.59.40 JOB07321  ICH70001I STC     LAST ACCESS AT 19:59:39 ON WEDNESDAY, NOVEMBER 3, 2004
19.59.40 JOB07321  $HASP373 IVP8IM11 STARTED - INIT A    - CLASS A - SYS SC62
19.59.40 JOB07321  IEF403I IVP8IM11 - STARTED - TIME=19.59.40 - ASID=003C - SC62
19.59.40 JOB07321  +DFS0578I - READ SUCCESSFUL FOR DDNAME PROCLIB  MEMBER = DFSINTDC  IMSI
20.00.33 JOB07321  +IMS2IMI: USERID=STC
20.00.33 JOB07321  +IMS2IMI: IMS_GU Status IOPCB.STC_CODE=
20.00.33 JOB07321  +IMS2IMI: IMS_GU Input=  PCMIT YYYYYYYYYYYYYYYYY
20.00.33 JOB07321  +IMS2IMI: IMS_GU Successful
20.00.33 JOB07321  +IMS2IMI: IMS_ISRT Entry.
20.00.33 JOB07321  +IMS2IMI: IMS_ISRT Output=IMS2IMP Completed OK,
DATETIME=20041103200033558
20.00.33 JOB07321  +IMS2IMI: IMS_ISRT Status IOPCB.STC_CODE=
20.00.33 JOB07321  +IMS2IMI: IMS_ISRT Successful
20.00.33 JOB07321  IEA995I SYMPTOM DUMP OUTPUT  970
   970              USER COMPLETION CODE=0711 REASON CODE=0000001E
```

```
   970             TIME=20.00.33  SEQ=04351  CPU=0000  ASID=003C
   970             PSW AT TIME OF ERROR  077C1600   9EDD5636  ILC 2  INTC 0D
   970               NO ACTIVE MODULE FOUND
   970               NAME=UNKNOWN
   970               DATA AT PSW  1EDD5630 - 00181610  0A0D17FF  58E094D4
   970               AR/GR 0: 00000000/84000000   1: 00000000/840002C7
   970                     2: 00000000/00000032   3: 00000000/1EDD523C
   970                     4: 00000000/1F704060   5: 00000000/1F10F698
   970                     6: 00000000/1F3E3D20   7: 00000000/00000003
   970                     8: 00000000/1F410110   9: 00000000/1F704060
   970                     A: 00000000/1EFE7B40   B: 00000000/00965D80
   970                     C: 00000000/1EDD523C   D: 00000000/1F704660
   970                     E: 00000000/1F10F79C   F: 00000001/0000001E
   970             END OF SYMPTOM DUMP
20.00.36 JOB07321  +DFS0578I - READ SUCCESSFUL FOR DDNAME PROCLIB  MEMBER = DFSINTDC  IMSI
```

Example 5-8 The IMS outbound program not rolling back with RRS=N

```
J E S 2  J O B  L O G  --  S Y S T E M  S C 6 1  --  N O D E  W T S C P L X 1

19.59.10 JOB07312 ---- WEDNESDAY, 03 NOV 2004 ----
19.59.10 JOB07312  IRR010I  USERID STC      IS ASSIGNED TO THIS JOB.
19.59.11 JOB07312  ICH70001I STC     LAST ACCESS AT 19:59:09 ON WEDNESDAY, NOVEMBER 3, 2004
19.59.11 JOB07312  $HASP373 IVP8HM11 STARTED - INIT A    - CLASS A - SYS SC61
19.59.11 JOB07312  IEF403I IVP8HM11 - STARTED - TIME=19.59.11 - ASID=0028 - SC61
19.59.11 JOB07312  +DFS0578I - READ SUCCESSFUL FOR DDNAME PROCLIB  MEMBER = DFSINTDC  IMSH
20.00.33 JOB07312  +IMS1PI1: PL/I IMS stub routine Begins...
20.00.33 JOB07312  +IMS1PI1: IMS_GU Entry.
20.00.33 JOB07312  +IMS1PI1: IMS_GU Status IOPCB.STC_CODE=
20.00.33 JOB07312  +IMS1PI1: IMS_GU Input= PCMIT YYYYYYYYYYYYYYYY
20.00.33 JOB07312  +IMS1PI1: IMS_GU Successful
20.00.33 JOB07312  +IMS1PS1: Outbound routine begins...
20.00.33 JOB07312  +IMS1PS1: Allocate Entry.
20.00.33 JOB07312  +IMS1PS1: Allocate User_ID=         , Password=         .
20.00.33 JOB07312  +IMS1PS1: Allocate Return_Code=0
20.00.33 JOB07312  +IMS1PS1: Allocate Conversation_ID=224BB3F80000032E
20.00.33 JOB07312  +IMS1PS1: Allocate RC is OK
............. omissis ............
20.00.33 JOB07312  +IMS1PS1: Receive_And_Wait RCV1 Entry, Conversation_ID=224BB3F80000032E
20.00.33 JOB07312  +IMS1PS1: Receive_And_Wait RCV1 Return_Code=0
20.00.33 JOB07312  +IMS1PS1: Receive_And_Wait RCV1 Receive_Length=68
20.00.33 JOB07312  +IMS1PS1: Receive_And_Wait RCV1 Receive_Buffer=IMS2IMP Completed OK,
DATETIME=20041103200033558
20.00.33 JOB07312  +IMS1PS1: Receive_And_Wait RCV1 Data_Received=2
20.00.33 JOB07312  +IMS1PS1: Receive_And_Wait RCV1 Request_to_Send_Received=0
20.00.33 JOB07312  +IMS1PS1: Receive_And_Wait RCV1 Status_Received is
atb_confirm_send_received
20.00.33 JOB07312  +IMS1PS1: Receive_And_Wait RCV1 Succeeded
20.00.33 JOB07312  +IMS1PS1: Receive_And_Wait RCV1 RC is OK
20.00.33 JOB07312  +IMS1PS1: Check_Received_Data Entry.
20.00.33 JOB07312  +IMS1PS1: Check_Received_Data Exit.
20.00.33 JOB07312  +IMS1PS1: Confirmed Entry, Conversation_ID=224BB3F80000032E
20.00.33 JOB07312  +IMS1PS1: Confirmed Return_Code=0
20.00.33 JOB07312  +IMS1PS1: Confirmed Succeeded
20.00.33 JOB07312  +IMS1PS1: Confirmed RC is OK
20.00.33 JOB07312  +IMS1PS1: Deallocate Entry, Conversation_ID=224BB3F80000032E
20.00.33 JOB07312  +IMS1PS1: Deallocate Return_Code=0
20.00.33 JOB07312  +IMS1PS1: Deallocate Succeeded
20.00.33 JOB07312  +IMS1PS1: Deallocate RC is OK
20.00.33 JOB07312  +IMS1PS1: Outbound routine ends...
```

```
20.00.33 JOB07312  +IMS1PI1: IMS1PS1 IMS1PS1_Return_Code=0
20.00.33 JOB07312  +IMS1PI1: IMS1PS1 call Succeeded, DATETIME=20041103200033623
20.00.33 JOB07312  +IMS1PI1: IMS_ISRT Entry.
20.00.33 JOB07312  +IMS1PI1: IMS_ISRT Output=IMS1TR1 Completed OK,
DATETIME=20041103200033627
20.00.33 JOB07312  +IMS1PI1: IMS_ISRT Status IOPCB.STC_CODE=
20.00.33 JOB07312  +IMS1PI1: IMS_ISRT Successful
20.00.33 JOB07312  +IMS1PI1: IMS_GU Entry.
20.01.50 JOB07312  +IMS1PI1: IMS_GU Status IOPCB.STC_CODE=QC
20.01.50 JOB07312  +IMS1PI1: IMS_GU Input=
20.01.50 JOB07312  +IMS1PI1: IMS_GU Successful
20.01.50 JOB07312  +IMS1PI1: PL/I IMS stub routine Ends.....
```

6

Sample scenario: IMS to CICS

This chapter describes some basic scenarios that we use to show the flow, cause, and results of executing APPC events in a protected conversation.

For these examples we used an IMS, CICS, and DB2 subsystem. The next section contains a description of the system architecture used with these examples.

We describe three scenarios:

- A successful syncpoint and commit conversation
- A CICS transaction abend requiring rollback
- A generic error during a conversation and rollback

In addition, we discuss some design considerations related to the CICS APPC implemented architecture.

6.1 Description

For IMS to CICS protected conversations scenarios we use IMS as the outbound conversation partner and CICS as the inbound conversation partner, which updates records in a DB2 table.

6.1.1 Architecture

Figure 6-1 shows the architecture we used to test the sample scenarios.

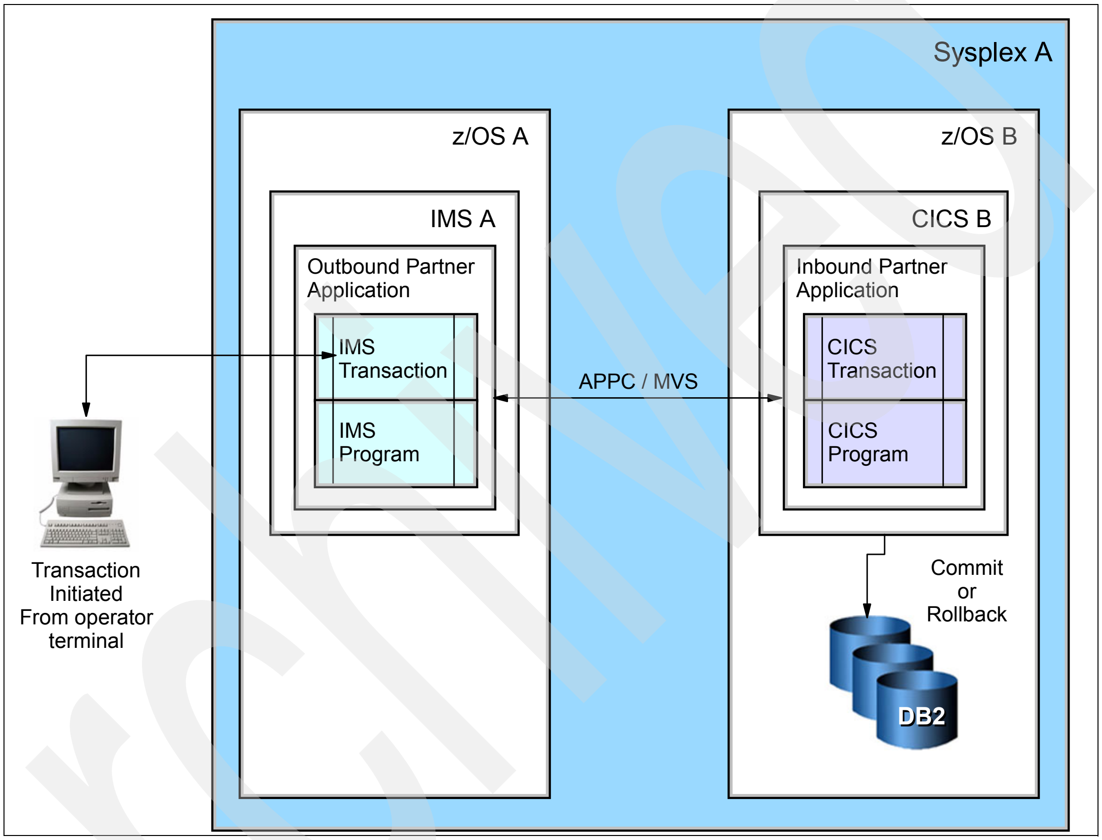


Figure 6-1 System Overview

Note: To re-create the environment we used:

Refer to Appendix A, “Installation definitions for Protected Conversation exploiters” on page 113 for detailed instructions for setting up the sample systems used throughout this book.

Refer to Appendix B, “APPC exploiter sample source code” on page 155 for the source code used in our examples.

6.1.2 Scenarios

In the following examples we pass data from an IMS transaction to a CICS subsystem which in turn executes a CICS transaction. The data passed to the CICS transaction determines the program flow for each of the examples documented in this chapter.

- Implied sync-point commit - no data passed via outbound partner. We only enter the transaction name in the IMS subsystem.
- Requested application sync-point commit. We enter the IMS transaction name followed by the commit keyword PCMIT and a user-defined Text String.
- Requested CICS transaction abend. We enter the IMS transaction name followed by the abend keyword PAEND and a user-defined Text String.
- Requested application initiated rollback. We enter the IMS transaction name followed by the rollback keyword PBACK and a user-defined Text String.

Note: The data passed from the IMS transaction (keyword and user text string) is used to create a record by the CICS transaction which has the following format:

```
timestamp keyword user_text_string
```

This record is inserted into a DB2 table. This allows us to clearly show records that have either been committed or rolled back as a result of our example scenarios.

6.2 How to manage and relate the pieces together

In this section we describe the two partners of the protected conversation and their independent program flow. The diagrams depict the flow of the APPC logic within the programs. Only the logic flow relating to the interaction of the APPC conversations is shown.

6.2.1 The outbound program

The outbound partner consists of an IMS transaction and an IMS program executing on any IMS within a sysplex. Figure 6-2 shows the outbound partner program logic we used to test each of our example scenarios.

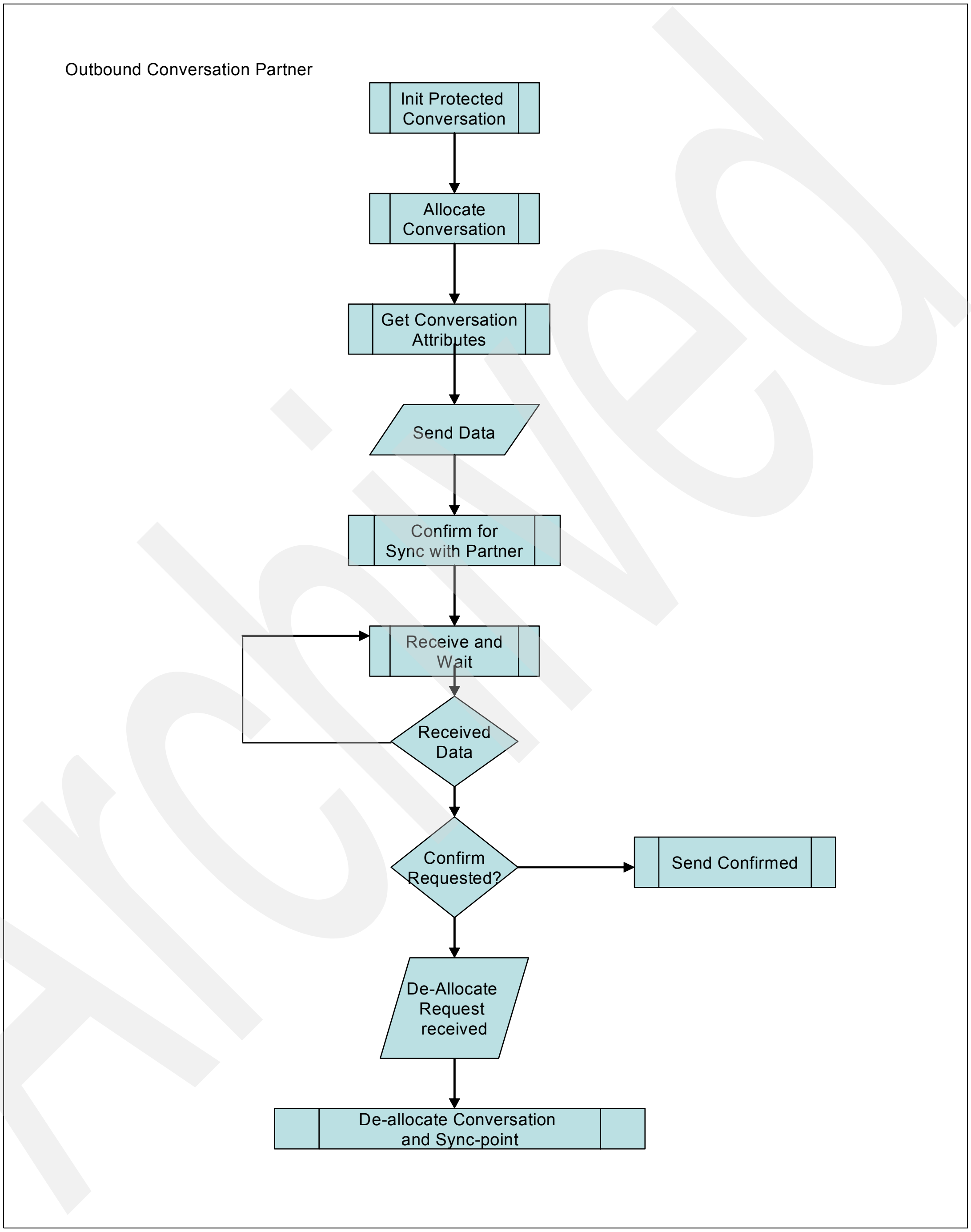


Figure 6-2 Outbound flow

6.2.2 The inbound program

The inbound partner consists of a CICS transaction and a CICS program that writes to DB2. The application can execute on any CICS region within a sysplex environment. Figure 6-3 shows the inbound partner program logic we used to test each of our example scenarios.

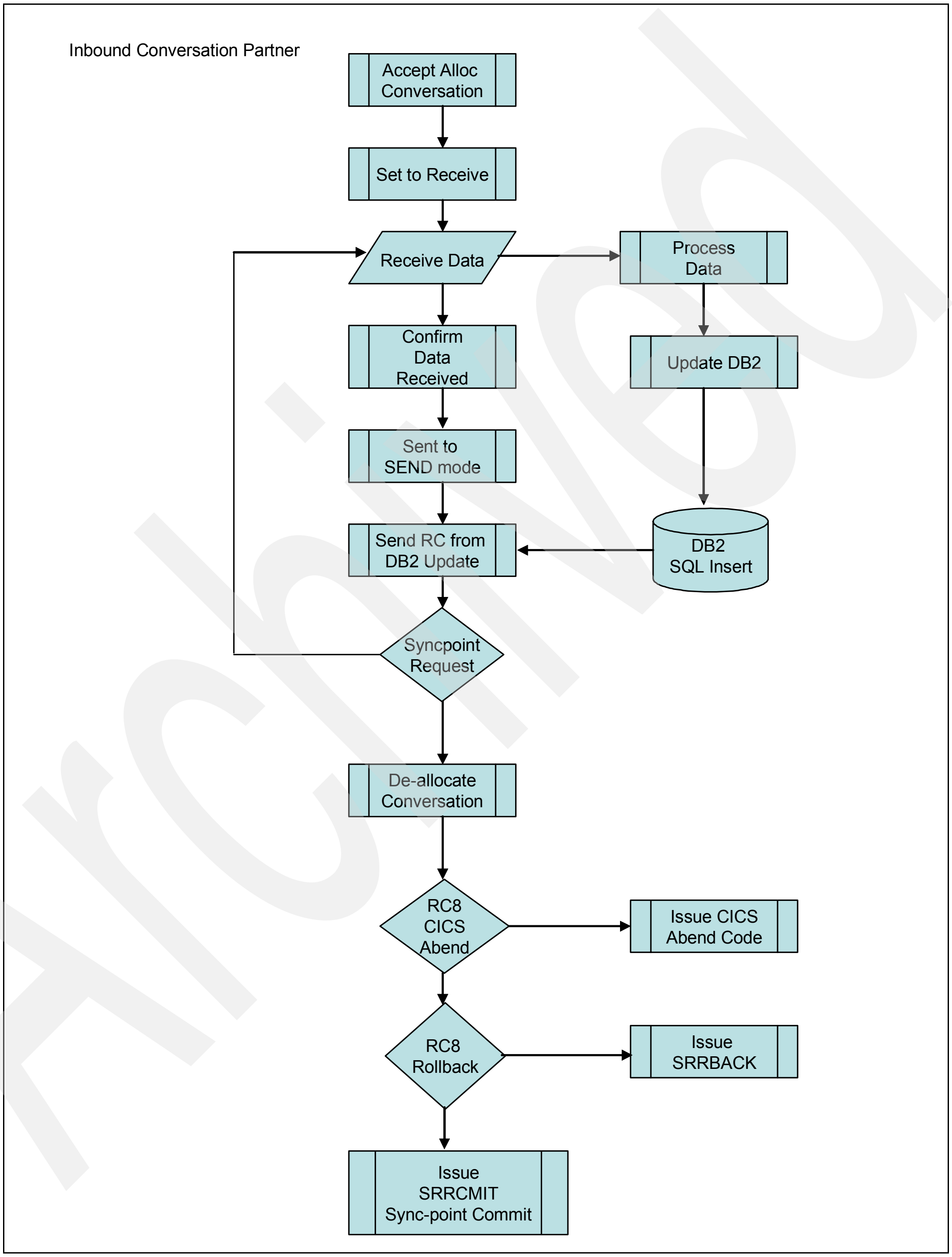


Figure 6-3 Inbound flow

6.3 Outbound and inbound conversation

Here we look at the protected conversation between our IMS outbound program and the CICS inbound program. We discuss the normal actions used in a sync-point commit. The following example applies to either an implied or explicit commit.

6.3.1 Example PCMIT: A successful sync-point and commit conversation

A protected conversation between an IMS outbound program and a CICS inbound program. In this example we examine the process involved during the execution of the conversation. A record of data is written to a DB2 table which can be displayed following execution to confirm sync-point commit success. Refer to the figure notes for further discussion regarding the application exchange.

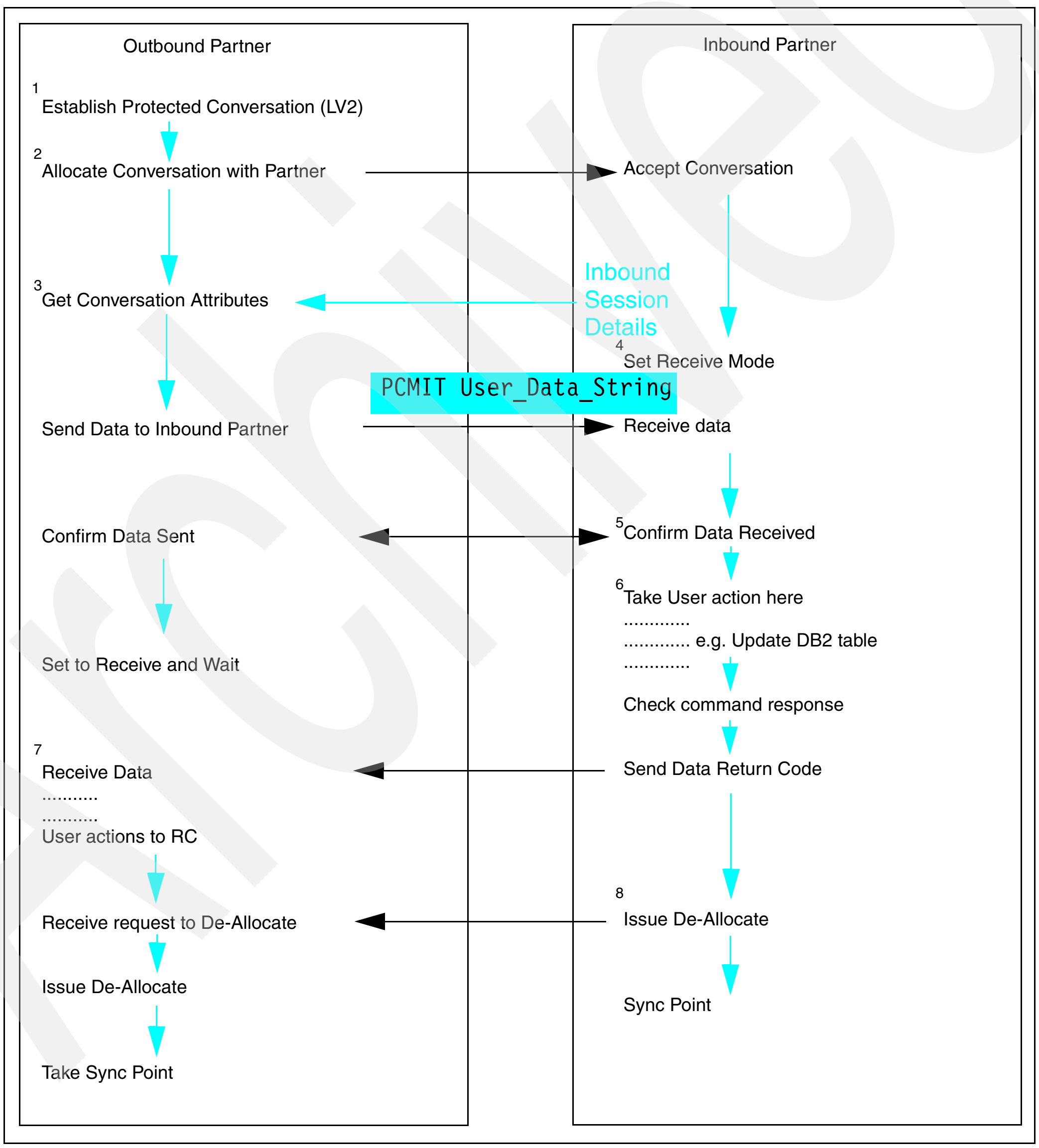


Figure 6-4 PCMIT example

Figure 6-4 notes:

1. IMS transaction executes the outbound program which sets Sync Level 2 for protected conversation mode.
2. The outbound partner issues an APPC ALLOCATE to initiate a conversation between IMS (outbound) and CICS (inbound). The CICS transaction executes the inbound CICS program and responds by issuing an APPC ACCEPT for the conversation.
3. The GET ATTRIBUTES returns important information with which the conversation was allocated. We use this command to get detailed information about our inbound partner.
4. The inbound partner switches to RECEIVE mode and waits for data from our outbound partner.
5. A CONFIRM and subsequent confirmation is exchanged to synchronize the partners and indicate the successful delivery of data.
6. The inbound program takes the data and carries out the user designed response. In this case the data will contain a PCMIT keyword and Text string which will subsequently be used to insert a row into a DB2 table.
7. The outbound program now has the opportunity of responding to the results of actions taken against the DB2 insertion. We expect a successful completion.
8. The inbound conversation issues a DE-ALLOCATE prior to taking the SYNC-POINT and exiting. The outbound program receives a DE-ALLOCATION request and then SYNC-POINTs prior to exiting.

In our test environment the IMS transaction passed the following data:

```
PCMIT user_text_string
```

This data is used to create the following DB2 record:

```
2004-11-03-11.23.12.948011  PCMIT     IMPAPPC: 03/11/04 11:23:13 SPG1
```

The record inserted in the DB2 table is retained as a result of the successful commit processing.

Example: Sync-Point Commit processing

Example 6-1shows the IMS transaction as entered at the host terminal. Example 6-2 displays the exchange of messages and events produced by the outbound IMS program and the inbound CICS program during the execution of the conversation for this scenario.

Example 6-1 IMS transaction screen

```
IMS1TR3 PCMIT TEST DATA TO SYNCPOINT

IMS1TR3 Completed OK, DATETIME=20041105094235278
```

Example 6-2 PCMIT messages and events

```
This is the results of the IMS outbound program in a Sync-point commit scenario:

IMS1PI3: IMS_GU Input= PCMIT TEST DATA FOR SYNCPOINT COMMIT
IMS1PI3: IMS_GU Successful
IMS1PS3: Outbound routine begins...
IMS1PS3: Allocate Entry.
IMS1PS3: Allocate User_ID=NO_USERID , Password=NO_PASSWRD.
IMS1PS3: Allocate Return_Code=0
```

```
IMS1PS3: Allocate Conversation_ID=224BB3F800000337
IMS1PS3: Allocate RC is OK
IMS1PS3: Get_Attributes Entry, Conversation_ID=224BB3F800000337
IMS1PS3: Get_Attributes Return_Code=0
IMS1PS3: Get_Attributes Return_Code=0
IMS1PS3: Get_Attributes Partner_LU_name=USIBMSC.SCSCPA8K
IMS1PS3: Get_Attributes Mode_name=APPCHOST
IMS1PS3: Get_Attributes Sync_level=2
IMS1PS3: Get_Attributes Conversation_correlator=
IMS1PS3: Get_Attributes LUW_id= USIBMSC.SCSIM8HA    Cv
IMS1PS3: Get_Attributes TP_name_length=0
IMS1PS3: Get_Attributes TP_name=
IMS1PS3: Get_Attributes Local_LU_name=SCSIM8HA
IMS1PS3: Get_Attributes Conversation_type=1
IMS1PS3: Get_Attributes User_id=
IMS1PS3: Get_Attributes Profile=
IMS1PS3: Get_Attributes User_token=
IMS1PS3: Get_Attributes Conversation_state=3
IMS1PS3: Get_Attributes Succeeded
IMS1PS3: Get_attributes RC is OK
IMS1PS3: Send_Data Entry, Conversation_ID=224BB3F800000337
IMS1PS3: Send_Data Return_Code=0
IMS1PS3: Send_Data Request_to_Send_Received=0
IMS1PS3: Send_Data Succeeded
IMS1PS3: Send_Data RC is OK
IMS1PS3: Confirm Entry, Conversation_ID=224BB3F800000337
IMS1PS3: Confirm Return_Code=0
IMS1PS3: Confirm Request_to_Send_Received=0
IMS1PS3: Confirm Succeeded
IMS1PS3: Confirm RC is OK
IMS1PS3: Receive_And_Wait RCV1Entry,  Conversation_ID=224BB3F800000337
IMS1PS3: Receive_And_Wait RCV1 Return_Code=0
IMS1PS3: Receive_And_Wait RCV1 Receive_Length=80
IMS1PS3: Receive_And_Wait RCV1 Receive_Buffer=0
IMS1PS3: Receive_And_Wait RCV1 Data_Received=2
IMS1PS3: Receive_And_Wait RCV1 Request_to_Send_Received=0
IMS1PS3: Receive_And_Wait RCV1 Status_Received is atb_no_status_received
IMS1PS3: Receive_And_Wait RCV1 Succeeded
IMS1PS3: Receive_And_Wait RCV1 RC is OK
IMS1PS3: Check_Received_Data Entry.
IMS1PS3: Check_Received_Data Exit.
IMS1PS3: Receive_And_Wait RCV2Entry,  Conversation_ID=224BB3F800000337
IMS1PS3: Receive_And_Wait RCV2 Return_Code=0
IMS1PS3: Receive_And_Wait RCV2 Receive_Length=256
IMS1PS3: Receive_And_Wait RCV2 Receive_Buffer=
IMS1PS3: Receive_And_Wait RCV2 Data_Received=0

This is the results of the CICS inbound program in a Sync-point commit scenario:

IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     000-MAIN.
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     100-APPC-ACCEPT.
IMPAPPC: 05/11/04 09:42:35 SPG1: CMACCP RC  =      CM-OK
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     200-APPC-RECEIVE.
IMPAPPC: 05/11/04 09:42:35 SPG1: APPC-CMRCV  : CONTENTS OF DATA-BUFFER:
 PCMIT TEST DATA FOR SYNCPOINT COMMIT
IMPAPPC: 05/11/04 09:42:35 SPG1: CMRCV RC   =      CM-OK
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     400-CONFIRM-DATA-RECEIVED
IMPAPPC: 05/11/04 09:42:35 SPG1: CMCFMD RC  =      CM-OK
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     200-APPC-RECEIVE.
IMPAPPC: 05/11/04 09:42:35 SPG1: APPC-CMRCV  : CONTENTS OF DATA-BUFFER:
```

```
IMPAPPC: 05/11/04 09:42:35 SPG1: CMRCV RC   =      CM-OK
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     210-RECEIVE-LOOP.
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     250-PARSE-CMD-LINE.
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     260-GET-CMDSTR.
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     800-APPC-UPDATE-DB2.
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     300-SEND-RETURN-CODE.
IMPAPPC: 05/11/04 09:42:35 SPG1: CMSST RC   =      CM-OK
IMPAPPC: 05/11/04 09:42:35 SPG1: APPC-CMSEND : CONTENTS OF DATA-BUFFER:
IMPAPPC: 05/11/04 09:42:35 0
IMPAPPC: 05/11/04 09:42:35 SPG1: CMSEND RC  =      CM-OK
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     500-DEALLOCATE-CONVERSATION.
IMPAPPC: 05/11/04 09:42:35 SPG1: CMDEAL RC  =      CM-OK
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     450-APPC-ISSUE-SRRCMIT.
IMPAPPC: 05/11/04 09:42:35 SPG1: SRRCMIT RC =      CM-OK
IMPAPPC: 05/11/04 09:42:35 SPG1: Called  subr:     900-LETS-EXIT.
```

6.4 How to handle failure scenarios

In this section we look at two scenarios which can encounter common error conditions when a conversation exists between IMS, CICS and DB2. These scenarios are:

- A CICS transaction abend is issued prior to the end of a protected conversation.
- A generic error condition is encountered during the execution of a protected conversation which results in a Rollback being issued.

The example scenarios shown here may be applied to many application designs which required rollback error protection for data.

6.4.1 Example PAEND: A CICS transaction abend requiring rollback

Here we look at the protected conversation between our IMS outbound program and the CICS inbound program. In this example we encounter a CICS transaction abend during the execution of the conversation. As a result of the transaction abend, any data written to the DB2 table will be rolled back during recovery of the CICS unit of work. Refer to the figure notes for a detailed further description of the application exchange.

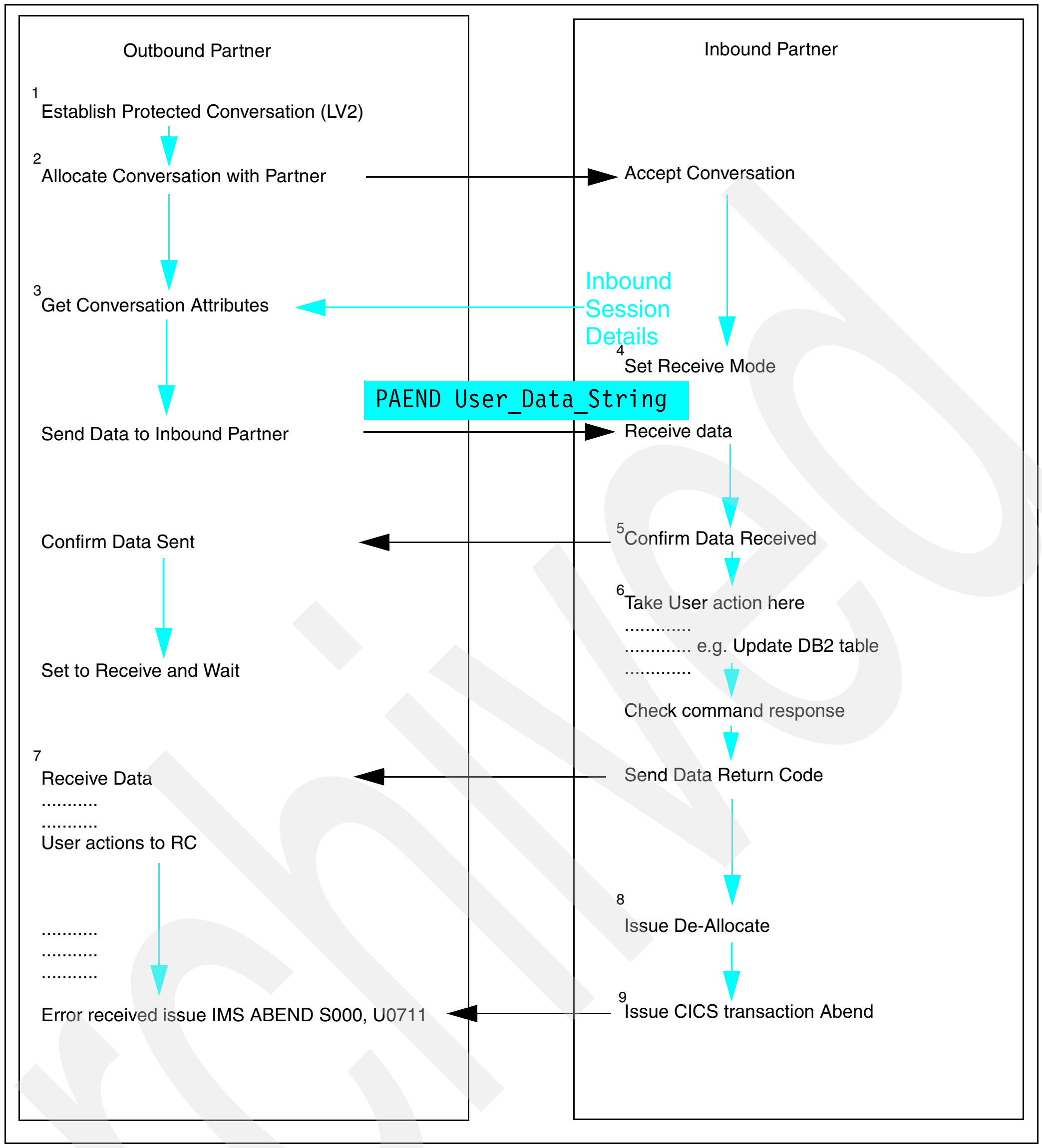


Figure 6-5 PAEND example

Figure 6-5 notes:

1. IMS transaction executes the outbound program which sets the Sync Level 2 for protected conversation mode.
2. The outbound partner issues an APPC ALLOCATE to initiate a conversation between IMS (outbound) and CICS (inbound). The CICS transaction executes the inbound CICS program and responds by issuing an APPC ACCEPT for the conversation.
3. The GET ATTRIBUTES returns important information with which the conversation was allocated. We use this command here to get detailed information about our inbound partner.
4. The inbound partner switches to RECEIVE mode and waits for data from our outbound partner.
5. A CONFIRM and subsequent confirmation is exchanged to synchronize the partners and indicate the successful delivery of data.
6. The inbound program takes the data and carries out the user-designed response. In this case the data will contain a PAEND keyword and Text string which will subsequently be used to insert a row into a DB2 table.
7. The outbound program now has the opportunity of responding to the results of actions taken against the DB2 insertion. We expect a successful completion for the DB2 update at this point. Note that we have not committed the data as yet.
8. The inbound conversation issues a DE-ALLOCATE and then encounters a problem in the CICS environment.
9. The inbound program issues a CICS transaction ABEND and passes control to CICS which rolls back any units of work and ends the conversation. The outbound program receives an error from the inbound program and issues an IMS ABEND prior to exiting.

In our test environment the IMS transaction passed the following data:

```
PAEND user_text_string
```

This data is used to create the following DB2 record:

```
2004-11-03-11.23.12.948011  PAEND IMPAPPC: 03/11/04 11:23:13 SPG1
```

The record inserted in the DB2 table is backed out as a result of the abend.

Example: CICS ABEND processing

Example 6-3 shows the IMS transaction as entered at the host terminal.

Example 6-3 IMS transaction screen

```
IMS1TR3 PAEND TEST DATA TO ABEND CICS TRANSACTION

 DFS555I TRAN IMS1TR3  ABEND S000,U0711 ; MSG IN PROCESS:
 IMS1TR3 PAEND TEST DATA TO ABEND CICS TRANSACTION
  2004/310   9:52:58
```

Example 6-4 displays the exchange of messages and events produced by the outbound IMS program and the inbound CICS program during the execution of the conversation for this scenario.

Example 6-4 PAEND messages and events

```
This is the results of the IMS outbound program in a CICS transaction ABEND scenario:

IMS1PI3: IMS_GU Input= PAEND TEST DATA TO ABEND CICS TRANSACTION
IMS1PI3: IMS_GU Successful
IMS1PS3: Outbound routine begins...
IMS1PS3: Allocate Entry.
IMS1PS3: Allocate User_ID=NO_USERID , Password=NO_PASSWRD.
IMS1PS3: Allocate Return_Code=0
IMS1PS3: Allocate Conversation_ID=224BB3F800000338
IMS1PS3: Allocate RC is OK
IMS1PS3: Get_Attributes Entry, Conversation_ID=224BB3F800000338
IMS1PS3: Get_Attributes Return_Code=0
IMS1PS3: Get_Attributes Return_Code=0
IMS1PS3: Get_Attributes Partner_LU_name=USIBMSC.SCSCPA8K
IMS1PS3: Get_Attributes Mode_name=APPCHOST
IMS1PS3: Get_Attributes Sync_level=2
IMS1PS3: Get_Attributes Conversation_correlator=    +¨K
IMS1PS3: Get_Attributes LUW_id= USIBMSC.SCSIM8HA  +ö
IMS1PS3: Get_Attributes TP_name_length=0
IMS1PS3: Get_Attributes TP_name=
IMS1PS3: Get_Attributes Local_LU_name=SCSIM8HA
IMS1PS3: Get_Attributes Conversation_type=1
IMS1PS3: Get_Attributes User_id=
IMS1PS3: Get_Attributes Profile=
IMS1PS3: Get_Attributes User_token=
IMS1PS3: Get_Attributes Conversation_state=3
IMS1PS3: Get_Attributes Succeeded
IMS1PS3: Get_attributes RC is OK
IMS1PS3: Send_Data Entry, Conversation_ID=224BB3F800000338
IMS1PS3: Send_Data Return_Code=0
IMS1PS3: Send_Data Request_to_Send_Received=0
IMS1PS3: Send_Data Succeeded
IMS1PS3: Send_Data RC is OK
IMS1PS3: Confirm Entry, Conversation_ID=224BB3F800000338
IMS1PS3: Confirm Return_Code=0
IMS1PS3: Confirm Request_to_Send_Received=0
IMS1PS3: Confirm Succeeded
IMS1PS3: Confirm RC is OK
IMS1PS3: Receive_And_Wait RCV1Entry,  Conversation_ID=224BB3F800000338
IMS1PS3: Receive_And_Wait RCV1 Return_Code=0
IMS1PS3: Receive_And_Wait RCV1 Receive_Length=80
IMS1PS3: Receive_And_Wait RCV1 Receive_Buffer=8
IMS1PS3: Receive_And_Wait RCV1 Data_Received=2
IMS1PS3: Receive_And_Wait RCV1 Request_to_Send_Received=0
IMS1PS3: Receive_And_Wait RCV1 Status_Received is atb_no_status_received
IMS1PS3: Receive_And_Wait RCV1 Succeeded

This is the results of the CICS inbound program in a CICS transaction ABEND scenario:

IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     000-MAIN.
IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     100-APPC-ACCEPT.
IMPAPPC: 05/11/04 09:52:58 SPG1: CMACCP RC  =      CM-OK
IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     200-APPC-RECEIVE.
IMPAPPC: 05/11/04 09:52:58 SPG1: APPC-CMRCV  : CONTENTS OF DATA-BUFFER:
 PAEND TEST DATA TO ABEND CICS TRANSACTION
IMPAPPC: 05/11/04 09:52:58 SPG1: CMRCV RC   =      CM-OK
IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     400-CONFIRM-DATA-RECEIVED
IMPAPPC: 05/11/04 09:52:58 SPG1: CMCFMD RC  =      CM-OK
IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     200-APPC-RECEIVE.
```

```
IMPAPPC: 05/11/04 09:52:58 SPG1: APPC-CMRCV  : CONTENTS OF DATA-BUFFER:

IMPAPPC: 05/11/04 09:52:58 SPG1: CMRCV RC   =      CM-OK
IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     210-RECEIVE-LOOP.
IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     250-PARSE-CMD-LINE.
IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     260-GET-CMDSTR.
IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     800-APPC-UPDATE-DB2.
IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     300-SEND-RETURN-CODE.
IMPAPPC: 05/11/04 09:52:58 SPG1: CMSST RC   =      CM-OK
IMPAPPC: 05/11/04 09:52:58 SPG1: APPC-CMSEND : CONTENTS OF DATA-BUFFER:
IMPAPPC: 05/11/04 09:52:58 8
IMPAPPC: 05/11/04 09:52:58 SPG1: CMSEND RC  =      CM-OK
IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     500-DEALLOCATE-CONVERSATION.
IMPAPPC: 05/11/04 09:52:58 SPG1: CMDEAL RC  =      CM-OK
IMPAPPC: 05/11/04 09:52:58 SPG1: Called  subr:     470-APPC-ISSUE-ABEND.
DFHAC2236 11/05/2004 09:52:58 SCSCPA8K Transaction IMP1 abend GD01 in program CICSPG1 term
-AAL. Updates to local recoverable resources will be backed out.
```

6.4.2 Example generic error during a conversation and rollback

Again we look at the protected conversation between our IMS outbound program and the CICS inbound program. In this example we will encounter an error during the execution of the conversation. As a result of the error, any data written to the DB2 table will be rolled back. Refer to the figure notes for a detailed description of the application exchange.

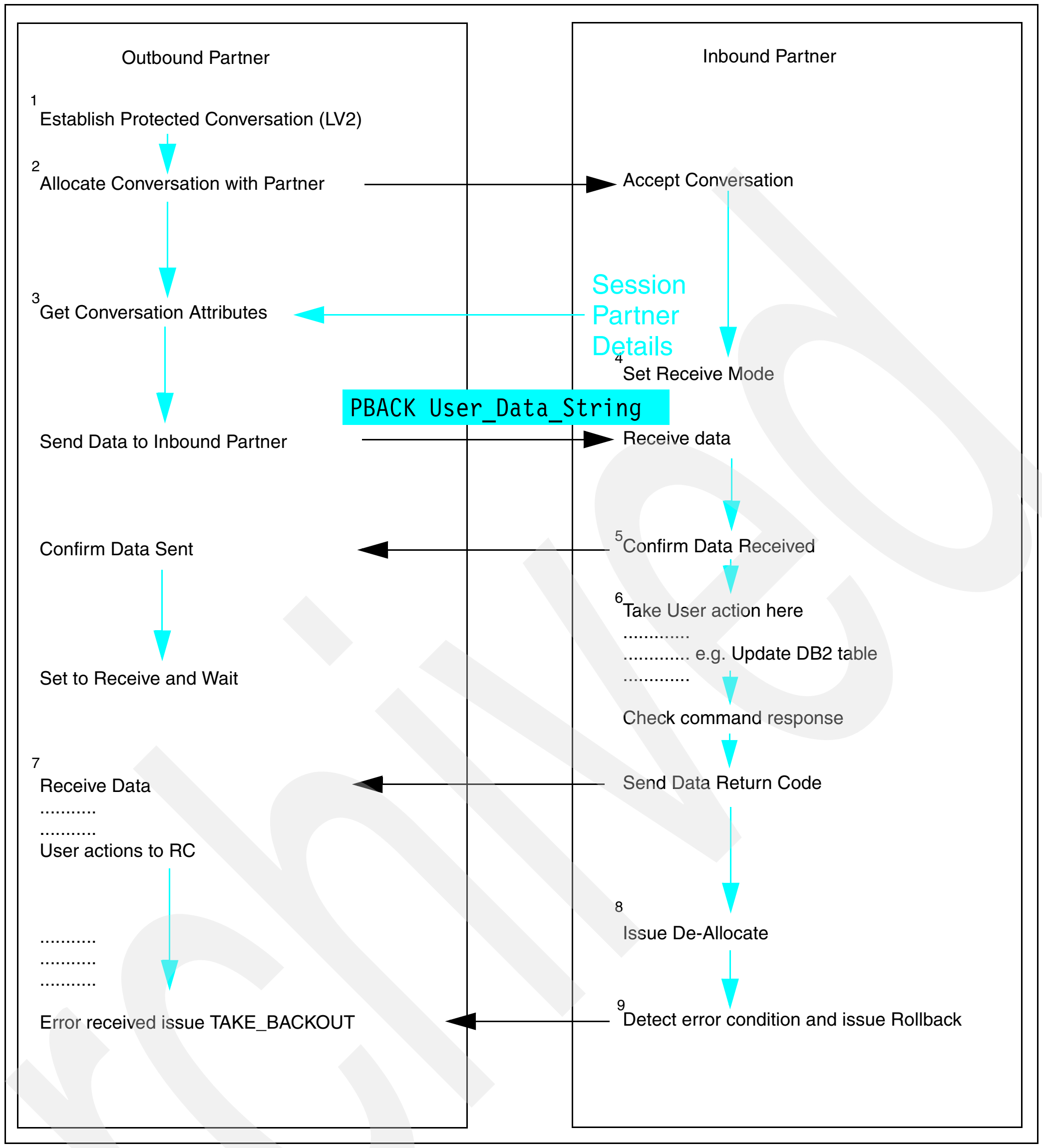


Figure 6-6 PBACK example

Figure 6-6 notes:

1. IMS transaction executes the outbound program which sets the Sync Level 2 for protected conversation mode.
2. The outbound program issues an APPC ALLOCATE to initiate a conversation between IMS (outbound) and CICS (inbound). The CICS transaction executes the inbound CICS program and responds by issuing an APPC ACCEPT for the conversation.
3. The GET ATTRIBUTES returns important information with which the conversation was allocated. We use this command here to get detailed information about our inbound partner.
4. The inbound partner switches to RECEIVE mode and waits for data from our outbound partner.
5. A CONFIRM and subsequent confirmation is exchanged to synchronize the partners and indicate the successful delivery of data.
6. The inbound program takes the data and carries out the user-designed response. In this case the data will contain a PBACK keyword and Text string which will subsequently be used to insert a row into a DB2 table.
7. The outbound program now has the opportunity of responding to the results of actions taken against the DB2 insertion. We expect a successful completion for the DB2 update at this point. Note that we have not committed the data as yet.
8. The inbound conversation issues a DE-ALLOCATE and then encounters an error condition.
9. The inbound program issues a SRRBACK and ends the conversation. The outbound program receives an error from the inbound partner and issues a TAKE_BACK and ends the conversation.

In our test environment the IMS transaction passed the following data:

```
PBACK user_text_string
```

This data is used to create the following DB2 record:

```
2004-11-03-11.23.12.948011  PBACK IMPAPPC: 03/11/04 11:23:13 SPG1
```

The record is subsequently backed out as a result of the SRRBACK condition.

Example: APPC Rollback processing

Figure 6-5 shows the IMS transaction as entered at the host terminal.

Example 6-5 IMS transaction screen

```
IMS1TR3 PBACK TEST DATA FOR BACKOUT RECOVERY

DFS555I TRAN IMS1TR3  ABEND S000,U0711 ; MSG IN PROCESS:
IMS1TR3 PBACK TEST DATA FOR BACKOUT RECOVERY
 2004/310  10:02:38
```

Example 6-6 displays the exchange of messages and events produced by the outbound IMS program and the inbound CICS program during the execution of the conversation for this scenario.

Example 6-6 PBACK messages and events

```
This is the results of the IMS outbound program in an application roll back scenario:

IMS1PI3: IMS_GU Input= PBACK TEST DATA FOR BACKOUT RECOVERY
IMS1PI3: IMS_GU Successful
IMS1PS3: Outbound routine begins...
IMS1PS3: Allocate Entry.
IMS1PS3: Allocate User_ID=NO_USERID , Password=NO_PASSWRD.
IMS1PS3: Allocate Return_Code=0
IMS1PS3: Allocate Conversation_ID=224BB3F800000339
IMS1PS3: Allocate RC is OK
IMS1PS3: Get_Attributes Entry, Conversation_ID=224BB3F800000339
IMS1PS3: Get_Attributes Return_Code=0
IMS1PS3: Get_Attributes Return_Code=0
IMS1PS3: Get_Attributes Partner_LU_name=USIBMSC.SCSCPA8K
IMS1PS3: Get_Attributes Mode_name=APPCHOST
IMS1PS3: Get_Attributes Sync_level=2
IMS1PS3: Get_Attributes Conversation_correlator=     <
IMS1PS3: Get_Attributes LUW_id= USIBMSC.SCSIM8HA   *"f
IMS1PS3: Get_Attributes TP_name_length=0
IMS1PS3: Get_Attributes TP_name=
IMS1PS3: Get_Attributes Local_LU_name=SCSIM8HA
IMS1PS3: Get_Attributes Conversation_type=1
IMS1PS3: Get_Attributes User_id=
IMS1PS3: Get_Attributes Profile=
IMS1PS3: Get_Attributes User_token=
IMS1PS3: Get_Attributes Conversation_state=3
IMS1PS3: Get_Attributes Succeeded
IMS1PS3: Get_attributes RC is OK
IMS1PS3: Send_Data Entry, Conversation_ID=224BB3F800000339
IMS1PS3: Send_Data Return_Code=0
IMS1PS3: Send_Data Request_to_Send_Received=0
IMS1PS3: Send_Data Succeeded
IMS1PS3: Send_Data RC is OK
IMS1PS3: Confirm Entry, Conversation_ID=224BB3F800000339
IMS1PS3: Confirm Return_Code=0
IMS1PS3: Confirm Request_to_Send_Received=0
IMS1PS3: Confirm Succeeded
IMS1PS3: Confirm RC is OK
IMS1PS3: Receive_And_Wait RCV1Entry,  Conversation_ID=224BB3F800000339
IMS1PS3: Receive_And_Wait RCV1 Return_Code=0
IMS1PS3: Receive_And_Wait RCV1 Receive_Length=80
IMS1PS3: Receive_And_Wait RCV1 Receive_Buffer=8
IMS1PS3: Receive_And_Wait RCV1 Data_Received=2
IMS1PS3: Receive_And_Wait RCV1 Request_to_Send_Received=0
IMS1PS3: Receive_And_Wait RCV1 Status_Received is atb_no_status_received
IMS1PS3: Receive_And_Wait RCV1 Succeeded
IMS1PS3: Receive_And_Wait RCV1 RC is OK
IMS1PS3: Check_Received_Data Entry.
IMS1PS3: Check_Received_Data Exit.
IMS1PS3: Receive_And_Wait RCV2Entry,  Conversation_ID=224BB3F800000339
IMS1PS3: Receive_And_Wait RCV2 Return_Code=100
IMS1PS3: Receive_And_Wait RCV2 Receive_Length=256
IMS1PS3: Receive_And_Wait RCV2 Receive_Buffer=
IMS1PS3: Receive_And_Wait RCV2 Data_Received=0
IMS1PS3: Receive_And_Wait RCV2 Request_to_Send_Received=0
IMS1PS3: Receive_And_Wait RCV2 Status_Received is atb_no_status_received
IMS1PS3: Receive_And_Wait RCV2 Failed
IMS1PS3: Error_Extract Entry, Conversation_ID=224BB3F800000339
IMS1PS3: Error_Extract Return_Code=0
```

```
IMS1PS3: Error_Extract Service_Name=ATBRCVW
IMS1PS3: Error_Extract Service_Reason_Code=100
IMS1PS3: Error_Extract: Message_Text follows:
ATB80100I From VTAM macro APPCCMD: Primary error return code: 005C, secondary error return
code: 0001, sense code: 08240000.
IMS1PS3: Error_Extract Exit.
IMS1PS3: Receive_And_Wait RCV2 RC is TAKE_BACKOUT
IMS1PS3: Outbound routine ends...
IMS1PI3: IMS1PS3 IMS1PS3_Return_Code=100

This is the results of the CICS inbound program in an application roll back scenario:

IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     000-MAIN.
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     100-APPC-ACCEPT.
IMPAPPC: 05/11/04 10:02:39 SPG1: CMACCP RC  =      CM-OK
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     200-APPC-RECEIVE.
IMPAPPC: 05/11/04 10:02:39 SPG1: APPC-CMRCV  : CONTENTS OF DATA-BUFFER:
 PBACK TEST DATA FOR BACKOUT RECOVERY
IMPAPPC: 05/11/04 10:02:39 SPG1: CMRCV RC   =      CM-OK
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     400-CONFIRM-DATA-RECEIVED
IMPAPPC: 05/11/04 10:02:39 SPG1: CMCFMD RC  =      CM-OK
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     200-APPC-RECEIVE.
IMPAPPC: 05/11/04 10:02:39 SPG1: APPC-CMRCV  : CONTENTS OF DATA-BUFFER:

IMPAPPC: 05/11/04 10:02:39 SPG1: CMRCV RC   =      CM-OK
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     210-RECEIVE-LOOP.
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     250-PARSE-CMD-LINE.
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     260-GET-CMDSTR.
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     800-APPC-UPDATE-DB2.
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     300-SEND-RETURN-CODE.
IMPAPPC: 05/11/04 10:02:39 SPG1: CMSST RC   =      CM-OK
IMPAPPC: 05/11/04 10:02:39 SPG1: APPC-CMSEND : CONTENTS OF DATA-BUFFER:
IMPAPPC: 05/11/04 10:02:39 8
IMPAPPC: 05/11/04 10:02:39 SPG1: CMSEND RC  =      CM-OK
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     500-DEALLOCATE-CONVERSATION.
IMPAPPC: 05/11/04 10:02:39 SPG1: CMDEAL RC  =      CM-OK
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     460-APPC-ISSUE-SRRBACK.
IMPAPPC: 05/11/04 10:02:39 SPG1: SRRBACK RC =      CM-OK
IMPAPPC: 05/11/04 10:02:39 SPG1: Called  subr:     900-LETS-EXIT.

DFHZN2701 11/05/2004 10:02:38 SCSCPA8K Log data sent on ISC session is DFHAC2223 10:02:38
SCSCPA8K Transaction IMP1 has failed with abend ASP2 due to the links to the remote systems
being in an invalid state. Updates will be backed out.

DFHAC2253 11/05/2004 10:02:38 SCSCPA8K Transaction IMP1 running program CICSPG1 term -AAL
has failed with abend ASP2 due to the links to the remote systems being in an invalid
state. Updates will be backed out.
```

Note: CICS issues an abend ASP2 to indicate the failure of the conversation and complete its own end of LUW processing.

6.4.3 Architecture and program design issues

When developing APPC applications between CICS and a partner application there are some considerations that must be addressed. As the architecture for CICS deviates from the APPC/MVS implementation there is a scenario where the conversation may be lost but *not* abended due to these architectural differences. This section describes the scenario and possible resolutions to help you design a more robust application.

CICS considerations

Its is possible to lose a conversation with a partner LU and still commit processing in what you may normally consider a correctly defined application design. Consider the following test:

Objective

We create an error in the inbound application flow and expect CICS to recover from the problem and issue a transaction abend.

Scenario

A CICS transaction, named GTCX, is executed and in turn executes program GTCICS02. This program starts an APPC conversation with an IMS LU using a synchlevel=2.

Due to an application programming error, after requesting an APPC ISSUE CONFIRMATION, the program performs the FREE of the conversation (this is permitted as specified in *The State tables for APPC mapped conversations DOCNUM=SC34-6236).*

This causes an abend U0711 RC=1E on the inbound side, the IMS application, with the following APPC message:

```
ATB80112I Protocol Violation:  APPC/MVS received deallocation status on a conversation
with Sync_Level of Syncpt, but not during a two-phase commit exchange.
```

You can also review the RRS backout on the inbound system:

```
SC62     2004/11/02 14:56:55.906666 BLOCKID=000000001BFFCC4
  URID=BC0FC3617E6CC000000000A4010E0000 JOBNAME=RRS      USERID=STC
  PARENT URID=00000000000000000000000000000000
  SURID=N/A
  WORK MANAGER NAME=IMS.IMSI____V081.STL.SANJOSE.IBM
  SYNCPOINT=Backout RETURN CODE=00000000
  START=2004/11/02 19:56:55.906251 COMPLETE=2004/11/02 19:56:55.906387
  EXITFLAGS=42000000
  LUWID=USIBMSC.SC38TC56 0FC3617FA30A 0001  TID=            GTID=

  FORMATID=003654612722 (decimal) D9D4F6F2 (hexadecimal)
  GTRID=
F0F0F2F0F6F4F1C3F7C9C2D4F0F2F0F0F0F0F0F0F0F1F0C5C3C210E4E2C9C2D4E2C34BE2C3F3F8E3
C3F5F60FC3617FA30A000100
  BQUAL=
E4E2C9C2D4E2C34BE2C3E2C3D7C1F8D200E4E2C9C2D4E2C34BE2C3E2C9D4F8C9C100
```

However, on the originator CICS side, the program completes successfully and commits the updates.

Analysis

On the surface it would seem that the CICS transaction should abend and rollback any changes as a result of the IMS abend, which does not commit any changes.

What you need to be aware of is that CICS deviates from the APPC architecture in allowing an SL(2) conversation to be deallocated at any time when the conversation is in *send state.* The reasons for this deviation are historical.

Execution of the FREE command results in a request, data (if previously buffered by CICS), and DFC (CEB, RQE1), being passed to VTAM for onward transmission to IMS. CICS breaks the association between the user task and the session allocated for the conversation when the request has been accepted by VTAM. The result is that the negative response sent by IMS is received by CICS but cannot be passed back to the user task. You can see the same results if you end a conversation by executing a SEND command with the LAST and WAIT options specified.

Resolution

When designing your CICS applications for communication between CICS and IMS you need to omit the WAIT option from a SEND command. Do not issue a FREE while in the SEND state until you are sure that the partner LU has completed its processing. The later execution of a SYNCPOINT command will result in the data being sent with DFC acceptable to IMS.

7

Monitoring

This chapter outlines the tools and utilities available to monitor and tune your APPC environment:

- SMF records - collection and tooling
- How to interpret the data
- The ATBTRACE REXX facility
- The RRS REXX batch log processor

7.1 SMF records - collection and tooling

With the SMF type 33 records, it is possible to analyze the topology of the APPC conversations; for example, to understand how many and what type of conversations belong to the single application transaction.

This can be made possible by associating the value of the Logical Unit of Work ID associated to each APPC conversation and available in the SMF 33 records.

You can also use the tool to understand the overall utilization of the APPC conversations.

When APPC/MVS conversations are deallocated by either partner program, SMF writes a type 33 subtype 2 record. For each conversation, SMF provides information such as:

- Conversation ID
- Name of the TP that issued the conversation request
- Local and partner LU name
- Number of sends and receives
- Amount of data sent and received

For inbound conversations that are processed by APPC/MVS servers, rather than transaction schedulers, subtype 2 records also contain information that is specific to server processing. For example, SMF records the specific dates and times that the conversation request was:

- Received by APPC/MVS
- Added to the server's allocate queue
- Received by the server for subsequent processing
- Deallocated

An SMF 33 record is written on the system where the inbound LU is located. If the outbound LU is located on a z/OS system, the target z/OS system will also write an SMF 33 record that describes the same conversation.

To help correlate conversations between partner programs, APPC/MVS applications can write user-specific information to a 255-byte user data field in the subtype 2 record through the Set_Conversation_Accounting_Information service.

For further information refer to *z/OS V1R4.0 MVS System Management Facilities (SMF),* SA22-7630.

7.2 SMF tool

You can use the SMF tool to extract and analyze the SMF records if your installation needs to analyze activity related to APPC protected conversations.

Extract type 33 subtype 2 records

The following JCL can be used to extract type 33 subtype 2 records from the system SMF data sets.

Example 7-1 SMF tool to extract and format type 33subtype 2 records

```
//*
//********************************************************************
//*                                                                  *
//* J1SMF332 - CONVERTS EACH SMF R33.2 IN A ROW OF CSV DATA          *
//*                                                                  *
//********************************************************************
//*
//   SET   CNTL=MARIO.SMF0332.CNTL     /* THE NAME OF THIS JCL LIBRARY
//   SET   SMFDATA=MARIO.SMF0332.DATA  /* DATASET CONTAINING SMF Type33
//   SET   CSVOUT=MARIO.SMF0332.CSVOUT /* THE OUTPUT FILE TO BE CREATED
//   SET   OUTVOL=MBZMBZ               /* OUTPUT FILE RESIDENCY VOLUME
//*
//         JCLLIB ORDER=(&CNTL)
//*
//SMF332   EXEC SMF332GO
//*
```

Notes: The SMF332 program is available in OBJ format. Refer to Appendix C, “Additional material” on page 161 for details on how to retrieve the material.

Define the variables in the JCL skeleton before submitting the job.

Execution statistics are going to be written to the SYSPRINT DDNAME; output data are collected on the CSVOUT DDNAME.

Refer to Table 7-1 for the record format produced.

7.2.1 How to interpret the data

Table 7-1 shows the record format produced using the SMF extract and format tool described previously.

Each record consists in an output row with fixed length single field.

Table 7-1 Record format and fields extracted from SMF 33 (2)

Example field contents	Field name	Description
SC61	smf33sid	System ID
IVP8HM11	smf33jid	Jobname
0	smf33cio	Conversation Inbound or Outbound
1	smf33clr	Conversation Partner LU location
0	smf33ckd	How the conversation was allocated
2	smf33csl	Synchronization Level
SCSIM8HA	smf33cll	Local LU
USIBMSC.SCSCPA8K	smf33cpl	Partner LU
IMSH	smf33csh	Conversation scheduler name
1900.001 00:00:00.000000	smf33crt	Date/time alloc request
1900.001 00:00:00.000000	smf33cqt	Date/time request on queue

Example field contents	Field name	Description
2004.323 15:38:54.146790	smf33cst	Date/time of conversation start
2004.323 15:38:54.323639	smf33cet	Date/time of conversation end
1	smf33csn	The number of SEND requests
83	smf33cds	Amount of data sent
1	smf33cre	The number of RECEIVE requests
84	smf33cdr	Amount of data received
7	smf33cvb	Number of called services
0	smf33crc	Conversation last return code
0	smf33crs	Conversation las reason code
1	smf33csa	Conversation state
2004.323 15:38:54.256495	smf33css	Date/time last service start
2004.323 15:38:54.323639	smf33cse	Date/time last service end
IMP1	smf33tpn_P	Partner TP name
	smf33tpn_L	Local TP name
USIBMSC.SCSIM8HAx.00 0000008C8FDFB2	smf33luw	Logical unit of work

When the job has completed you can download the sequential data to a third party product such as MYSQL, or use a spreadsheet (Excel®) to summarize the information or process the data in the host; for example, in a DB2 table. You might want to filter and consider only records for inbound conversations (smf33cio) to avoid counting the same conversations twice.

Once loaded into a relational database, APPC accounting information can be easily analyzed. It can be summarized to show workloads by system, synclevel, or lu-name; used to calculate minimum, maximum, and average values for conversation duration; used to figure the amount of data sent and received; and so on.

Sample SQL statements to show workload intensity by system and synclevel are provided in Example 7-2; the related output is shown in Table 7-2.

Example 7-2 Sample SQL statements

```
SELECT SMF33SID     AS SYSTEM_ID,
SMF33CSL            AS SYNC_LEVEL,
COUNT(SMF33CSL)     AS CONV_COUNT,

FROM  SG246486.SMF0332

GROUP BY SMF33SID, SMF33CSL

ORDER  BY SMF33SID, SMF33CSL;
```

Table 7-2 Sample Query output

SYSTEM_ID	SYNC_LEVEL	CONV_COUNT
MVSZ	0	4388
MVSZ	1	120
MVSZ	2	1902

Additional samples and information about loading the SMF data into a relational database can be found Appendix C, "Additional material" on page 161.

7.3 The ATBTRACE REXX facility

The ATBTRACE REXX tool that is able to trace every APPC API call between LUs is very useful, especially for programs using APPC verbs directly - IMS programs, for example. Trace writes its records into a user data set that, after stopping the trace, can be accessed.

Example 7-3 Our procedure to start the ATBTRACE

```
//*------------------------------------------------------------------*
//* SAMPLE JCL TO CALL ATBTRACE FACILITY                              *
//* YOU CAN START OR STOP THE TRACE                                   *
//* ON START YOU MUST SPECIFY AT LEAST THE OUTPUT DATASET AND ONE OR *
//*    MORE FILTER FACTORS.                                           *
//* ON START YOU MUST SPECIFY ONLY THE DATASET NAME YOU WANT STOP     *
//*    TRACE FOR.                                                     *
//*------------------------------------------------------------------*
//TRACE   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//SYSEXEC DD   DSN=SYS1.SBLSCLIO,DISP=SHR
//SYSTSPRT DD   SYSOUT=A
//SYSTSIN  DD   *
 %ATBTRACE START DATASET('MASSIMO.ATBTRACE') SYMDEST(IMS2DEST)
/*
```

Example 7-3 is a sample procedure for starting ATBTRACE on all the outcoming APPC transactions with symbolic destination of IMS2DEST; the output data set is MASSIMO.ATBTRACE.

Example 7-4 Our procedure to stop the ATBTRACE

```
//*------------------------------------------------------------------*
//* SAMPLE JCL TO CALL ATBTRACE FACILITY                              *
//* YOU CAN START OR STOP THE TRACE                                   *
//* ON START YOU MUST SPECIFY AT LEAST THE OUTPUT DATASET AND ONE OR  *
//*    MORE FILTER FACTORS.                                           *
//* ON START YOU MUST SPECIFY ONLY THE DATASET NAME YOU WANT STOP     *
//*    TRACE FOR.                                                     *
//*------------------------------------------------------------------*
//TRACE   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//SYSEXEC DD   DSN=SYS1.SBLSCLIO,DISP=SHR
//SYSTSPRT DD   SYSOUT=A
//SYSTSIN  DD   *
 %ATBTRACE STOP DATASET('MASSIMO.ATBTRACE')
/*
```

Example 7-4 is a sample procedure for stopping the ATBTRACE previously started on data set MASSIMO.ATBTRACE. For further information, refer to "Using the ATBTRACE REXX

Exec“ in *z/OS MVS Writing TPs for APPC/MVS*. Example 7-5 shows a sample output from an ATBTRACE REXX session.

Example 7-5 Output from ATBTRACE REXX

```
ATB60051I API TRACE WAS STARTED AT 10/21/2004 13:38:21.011650 FOR:
  LU     : USIBMSC.SCSIM8IA
  TP     : IMS2IMP
  SYMDEST: IMS2DEST
  USERID : *
ATB60055I ENTRY TO THE ATBALLC SERVICE:
 TIMESTAMP : 10/21/2004 13:38:32.696543
 ASID      : 0029
 TCB ADDR  : 006D1A18
 JOB NAME  : IVP8HM11
 LU        : USIBMSC.SCSIM8IA
 TP        : IMS2IMP
 USERID    : APPCUSR
 CONVID    : 0000000000000000
 PARAMETERS:
  CONVERSATION_TYPE: MAPPED_CONVERSATION
  SYM_DEST_NAME    : IMS2DEST
  PARTNER_LU_NAME  : SCSIM8IA
  MODE_NAME        : APPCHOST
  TP_NAME_LENGTH   : 7
  TP_NAME          : IMS2IMP
  RETURN_CONTROL   : WHEN_SESSION_ALLOCATED
  SYNC_LEVEL       : SYNCPOINT
  SECURITY_TYPE    : SECURITY_SAME
  USERID           :
  PASSWORD         :
  PROFILE          :
  USER_TOKEN       : 0040
  NOTIFY_TYPE      : 00000000
  TP_ID            : 0000000000000000
ATB60061I AN FMH-5 WAS SENT TO PARTNER LU USIBMSC.SCSIM8IA.
 TIMESTAMP : 10/21/2004 13:38:32.697611
 ASID      : 0029
 TCB ADDR  : 006D1A18
 JOB NAME  : IVP8HM11
 LU        : USIBMSC.SCSIM8IA
 TP        : IMS2IMP
 USERID    : APPCUSR
 CONVID    : 224BB3F800000080
 FMH-5     : 440502FF8003D1008007C9D4E2F2C9D4D70F0802D4C1E2E2C9D4D60500E2E8E2F1
             1910E4E2C9C2D4E2C34BE2C3E2C9D4F8C8C18E11A7964E25000108BC008E11A774
             B00500000000000000000000000000000000000000000000000000000000000000
             000000000000000000000000000000000000000000000000000000000000000000
             000000000000000000000000000000000000000000000000000000000000000000
             000000000000000000000000000000000000000000000000000000000000000000
             000000000000000000000000000000000000000000000000000000000000000000
             00000000000000000000000000000000000000000000000000
ATB60056I THE ATBALLC SERVICE COMPLETED.
 TIMESTAMP : 10/21/2004 13:38:32.697848
 ASID      : 0029
 TCB ADDR  : 006D1A18
 JOB NAME  : IVP8HM11
 LU        : USIBMSC.SCSIM8IA
TP        : IMS2IMP
USERID    : APPCUSR
CONVID    : 224BB3F800000080
```

```
PARAMETERS:
 CONVERSATION_ID: 224BB3F800000080
 RETURN_CODE    : OK
ATB60055I ENTRY TO THE ATBGTA2 SERVICE:
........ omissis .......
ATB60055I ENTRY TO THE ATBDEAL SERVICE:
 TIMESTAMP : 10/21/2004 13:38:33.294926
 ASID      : 0029
 TCB ADDR  : 006D1A18
 JOB NAME  : IVP8HM11
 LU        : USIBMSC.SCSIM8IA
 TP        : IMS2IMP
 USERID    : APPCUSR
 CONVID    : 224BB3F800000080
 PARAMETERS:
  CONVERSATION_ID: 224BB3F800000080
  DEALLOCATE_TYPE: DEALLOC_SYNC_LEVEL
  NOTIFY_TYPE    : 00000000
ATB60056I THE ATBDEAL SERVICE COMPLETED.
 TIMESTAMP : 10/21/2004 13:38:33.296025
 ASID      : 0029
 TCB ADDR  : 006D1A18
 JOB NAME  : IVP8HM11
 LU        : USIBMSC.SCSIM8IA
 TP        : IMS2IMP
 USERID    : APPCUSR
 CONVID    : 224BB3F800000080
 PARAMETERS:
  RETURN_CODE: OK
ATB60052I ATBTRACE STOP REQUEST ISSUED BY MASSIMO AT
          10/21/2004 13:38:44.371613.
          THE DATA SET CONTAINS TRACE DATA FOR:
API TRACE WAS STARTED AT 10/21/2004 13:38:21.011650 FOR:
  LU     : USIBMSC.SCSIM8IA
  TP     : IMS2IMP
  SYMDEST: IMS2DEST
  USERID : *
```

7.4 The RRS REXX batch log processor

All the RRS information is available through the ISPF RRS interface, a set of ISPF programs that let you query all the RRS data, filtering what you need. When you query the ARCHIVE log stream, your TSO session could be blocked for a long time if you request a large time period of data to analyze, or if the RRS log data sets are not available on disk and need to be recalled. For this kind of query, you can use a batch RRS query interface that allow you to retrieve the same information as from the ISPF interface, but though a batch job. The sample job is available in member ATRBATCH in SYS1.SAMPLIB. Shown in Example 7-6 is a sample customized for our environment and, in Example 7-7, the relative output.

Example 7-6 ATRBATCH JCL customized for our environment.

```
//*********************************************************************
//*
//*01* PROC NAME: ATRBATCH
//*01* DESCRIPTIVE NAME:  Batch mode RRS logstream formatter.
//*01* COMPONENT: MVS/RRS (SCRRS)
//*
/* ! You must change GNAME, find 'Note 1:' for more information.    */
```

```
/*******************************************************************/
LOG     = 'A'              /* U=UR, A=ARCHIVE, D=RMDATA, R=RESTART */
GNAME   = 'WTSCPLX1'       /* Find 'Note 1:' for more information. */
REPORT  = 'S'              /* D=DETAIL, S=SUMMARY                  */
IURID   = ' '
ISURID  =,                 /* 64 char SURID is too long for 1 line */
 ' '
IRMNAME = ' '              /* Can be used only for RMData log      */
AFTERD  = '20041103'       /* USE NO SLASHES IN THE DATE FIELDS    */
AFTERT  = '200459'         /* USE NO COLONS IN THE TIME FIELD      */
BEFORED = '20041103'       /* EXAMPLE: '20031105' MEANS 5 NOV 2003 */
BEFORET = '200500'         /* EXAMPLE: '142022' MEANS 2:20:22 PM   */
```

Example 7-7 Output from the ATRBATCH utility

```
This report was produced by a batch job.
The batch job passed the following data:
Log     = "A"
GName   = "WTSCPLX1"
Report  = "S"
IURID   = "                                "
ISURID  = "                                                                "
IRMName = "                                "
AfterD  = "20041103"
AfterT  = "200459"
BeforeD = "20041103"
BeforeT = "200500"

The batch job interprets the input as follows:
Log     =  Archive
GName   = "WTSCPLX1"
Report  =  Summary
IURID   = "                                "
ISURID  = "                                                                "
IRMName = "                                "
AfterD Year = "2004"
AfterD Month= "11"
AfterD Day  = "03"
AfterT Hour  = "20"
AfterT Minute= "04"
AfterT Second= "59"
BeforeD Year = "2004"
BeforeD Month= "11"
BeforeD Day  = "03"
BeforeT Hour  = "20"
BeforeT Minute= "05"
BeforeT Second= "00"
RRS/MVS LOG STREAM BROWSE SUMMARY REPORT

READING ATR.WTSCPLX1.ARCHIVE       LOG STREAM

SC52     2004/11/03 20:04:59.006474 BLOCKID=0000000002C85EB4
  URID=BC114A1A7E5A20000001ED26010B0000 JOBNAME=WS521S   USERID=ASSR1
  PARENT URID=00000000000000000000000000000000
  SURID=N/A
  WORK MANAGER NAME=BBO.CL521.CLU521.WS521.IBM
  SYNCPOINT=Commit  RETURN CODE=00000000
  START=2004/11/04 01:04:59.003408 COMPLETE=2004/11/04 01:04:59.006295
  EXITFLAGS=00840000
```

```
  LUWID=                                    TID=             GTID=

  FORMATID=003284271494 (decimal) C3C20186 (hexadecimal)
  GTRID=
BC114A1A0EB2E8C0000000010000F6A2BBD395FAEFB32B870000100C00000001090C042A
  BQUAL=
BC114A1A0EB2E8C0000000010000F6A2BBD395FAEFB32B870000100C00000001090C042A00000001

SC52     2004/11/03 20:04:59.589568 BLOCKID=0000000002C86098
  URID=BC114A1A7E5A20000001ED28010B0000 JOBNAME=WS521S   USERID=ASSR1
  PARENT URID=00000000000000000000000000000000
  SURID=N/A
  WORK MANAGER NAME=BBO.CL521.CLU521.WS521.IBM
  SYNCPOINT=Commit  RETURN CODE=00000000
  START=2004/11/04 01:04:59.588632 COMPLETE=2004/11/04 01:04:59.589387
  EXITFLAGS=00840000
  LUWID=                                    TID=             GTID=

  FORMATID=003284271494 (decimal) C3C20186 (hexadecimal)
  GTRID=
BC114A1A9DA71102000000010000F6A3BBD395FAEFB32B870000100C00000001090C042A
  BQUAL=
BC114A1A9DA71102000000010000F6A3BBD395FAEFB32B870000100C00000001090C042A00000001

SC62     2004/11/03 20:04:59.719753 BLOCKID=0000000002C8627C
  URID=BC114A1A7E6E40000000004A010E0000 JOBNAME=IVP8IM11 USERID=STC
  PARENT URID=00000000000000000000000000000000
  SURID=N/A
  WORK MANAGER NAME=IMS.IMSI____V081.STL.SANJOSE.IBM
  SYNCPOINT=Commit  RETURN CODE=00000000
  START=2004/11/04 01:04:59.690788 COMPLETE=2004/11/04 01:04:59.719509
  EXITFLAGS=00800000
  LUWID=USIBMSC.SCSIM8HA 4A1A5AA66245 0001  TID=             GTID=

  FORMATID=003654612722 (decimal) D9D4F6F2 (hexadecimal)
  GTRID=
F0F0F2F0F6F4F1C3F7C9C2D4F0F2F0F0F0F0F0F0F0F0F1F0C5C3C210E4E2C9C2D4E2C34BE2C3E2C9D4
F8C8C14A1A5AA66245000100
  BQUAL=
E4E2C9C2D4E2C34BE2C3E2C9D4F8C8C100E4E2C9C2D4E2C34BE2C3E2C9D4F8C9C100

SC61     2004/11/03 20:04:59.724230 BLOCKID=0000000002C864E3
  URID=BC114A1A7E5EF0000000006A01050000 JOBNAME=IVP8HM11 USERID=STC
  PARENT URID=00000000000000000000000000000000
  SURID=N/A
  WORK MANAGER NAME=SC61.IVP8HM11.0028
  SYNCPOINT=Commit  RETURN CODE=00000000
  START=2004/11/04 01:04:59.683062 COMPLETE=2004/11/04 01:04:59.724014
  EXITFLAGS=00800000
  LUWID=USIBMSC.SCSIM8HA 4A1A5AA66245 0001  TID=             GTID=

  FORMATID=003654931682 (decimal) D9D9D4E2 (hexadecimal)
  GTRID=
F0F0F2F0F6F4F1C3F7C9C2D4F0F2F0F0F0F0F0F0F0F0F1F0C5C3C2BC114A1A5AA67985
  BQUAL=
D9D9D4E24BBC114A1A5AA67D05F0F0F2F0F6F4F1C3F7C9C2D4F0F2F0F0F0F0F0F0F0F0F1F0C5C3C2
```

To relate together RRS, APPC, IMS, refer to 4.1, “How to manage the resources” on page 54.

A

Installation definitions for Protected Conversation exploiters

This appendix provides the system definitions required to install the inbound and outbound partner resources to implement an environment similar to the one we used to produce the examples included in this book.

In this appendix we provide definitions for the following resources:

- General
- CICS
- IMS
- DB2

For details about the inbound and outbound source code, refer to Appendix B, "APPC exploiter sample source code" on page 155.

Overview of installed components

When defining our systems for setting up an APPC protected conversation environment we utilized existing installation verification procedures and available examples where appropriate.

We used the basis of the DB2 and CICS IVP code to establish a connection and update facility for the APPC protected conversation. This allows us to write records to DB2 and view any updates that are committed to the database. We used SPUFI to interrogate DB2. This is not mandatory but is widely available.

The setup was organized into four areas: general, CICS, IMS, and DB2 definitions. Where required, we also provided the JCL, RDO definitions, SQL code, and source to aid you in installing a complete environment.

General definitions

PL/I and COBOL Enterprise compilers

Access to the compilers is needed to allow the five APPC/IMS (PL/I) and one APPC/CICS (COBOL) programs. The compiles are done via batch jobs so access to the PL/I and COBOL compilers via ISPF foreground panels is not required.

a. Enterprise PL/1 V3.3.0 (includes the compiler, samples, and so forth.)

   PL/I libraries established under HLQ: BMZ.EPLI.V3R3M0

```
IBMZ.EPLI.V3R3M0.SIBMZCMP (Load library)
IBMZ.EPLI.V3R3M0.SIBMZPRC (procs)
IBMZ.EPLI.V3R3M0.SIBMZSAMP (samples)
```

b. Enterprise COBOL Cobol v3.3.0 (includes the compiler)

   Cobol can be found under HLQ: IGY.ECOBOL.V3R3M0

```
IGY.ECOBOL.V3R3M0.SIGYCLST
IGY.ECOBOL.V3R3M0.SIGYCOMP (Load library)
IGY.ECOBOL.V3R3M0.SIGYMAC
IGY.ECOBOL.V3R3M0.SIGYPROC (procs)
IGY.ECOBOL.V3R3M0.SIGYSAMP (samples)
```

Version 3.3.0 of PL/I and COBOL is used for the setup exercise.

Network components

Commonly you implement protected conversations in an existing environment. From a network point of view, there are three components that must be addressed:

- General VTAM parameters
- CICS LUs parameters
- IMS LUs parameters

ACF/VTAM LOGMODE table entries

You must specify a Modetable and an entry within the table to be used for LU6.2 conversations. The VTAM APPL parameter MODETAB=LOGMODES indicates the Mode Table. LOGMODES resides in SYSx.VTAMLIB. Table A-1 lists the entries for the Logmodes.

Table A-1 Logmode Names

Logmode Name	Description
SNASVCMG	Logmode table entry for resources capable of acting as LU 6.2 devices, required for LU management.
APPCPCLM	Logmode table entry for resources capable of acting as LU 6.2 devices for PC target.
APPCHOST	Logmode table entry for resources capable of acting as LU 6.2 devices for host target in this example RU size of 4096 is used.

The VTAM APPL parameter DLOGMOD=APPCHOST indicates the entry (MODEENT) within the Mode Table. The Logmode entry used by the IMS system and CICS systems needs to be aligned.

For CICS-to-IMS links that are cross-domain, you must associate the IMS LOGMODE entry with the CICS applid (the generic applid for XRF systems) using the DLOGMOD or MODETAB parameters. Ensure the CICS Mode table entry contains a Logmode table entry (MODEENT) which is the same as the Logmode table entry used by IMS when allocating the conversation.

The Modetable LOGMODES and its associated APPCHOST Logmode is sourced from the ATBLMODE member in SYS1.SAMPLIB.

Example A-1: SYSx.SAMPLIB Logmodes

```
SYSx.SAMPLIB contains the following samples;
     ATBLJOB       JCL, VTAM logmode table link edit job.
     ATBLMODE      VTAM logmode table for APPC.
```

Extract from the ATBLMODE member (VTAM logmode table for APPC).

Example A-2: Extract from the ATBLMODE

```
LOGMODES MODETAB
         EJECT
***********************************************************************
         TITLE 'SNASVCMG'                                   *@R495812*
***********************************************************************
*        LOGMODE TABLE ENTRY FOR RESOURCES CAPABLE OF ACTING          *
*               AS LU 6.2 DEVICES                                     *
*               REQUIRED FOR LU MANAGEMENT                            *
***********************************************************************
SNASVCMG MODEENT LOGMODE=SNASVCMG,FMPROF=X'13',TSPROF=X'07',           *
               PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',            *
               RUSIZES=X'8585',ENCR=B'0000',                           *
               PSERVIC=X'060200000000000000000300'
***********************************************************************
         TITLE 'APPCPCLM'
***********************************************************************
*        LOGMODE TABLE ENTRY FOR RESOURCES CAPABLE OF ACTING          *
*               AS LU 6.2 DEVICES                                     *
*               FOR PC TARGET                                         *
*   IN THIS EXAMPLE THE DEFAULT RU SIZE FOR OS/2 (1024) IS USED  @01C*
***********************************************************************
APPCPCLM MODEENT LOGMODE=APPCPCLM,                                     *
               RUSIZES=X'8787',                                        *
               SRCVPAC=X'00',                                          *
               SSNDPAC=X'01'
```

```
***********************************************************************
         TITLE 'APPCHOST'
***********************************************************************
*        LOGMODE TABLE ENTRY FOR RESOURCES CAPABLE OF ACTING         *
*               AS LU 6.2 DEVICES                                     *
*               FOR HOST TARGET                                       *
*   IN THIS EXAMPLE RU SIZE OF 4096 IS USED                       @01C*
***********************************************************************
APPCHOST MODEENT LOGMODE=APPCHOST,                                     *
               RUSIZES=X'8989',                                        *
               SRCVPAC=X'00',                                          *
               SSNDPAC=X'01'
         MODEEND
         END
```

For further details, refer to *CICS TS Intercommunication Guide Version 2 Release 2.*

For APPC sessions, you can use the MODENAME option of the CICS DEFINE SESSIONS command to identify a VTAM logmode entry that in turn identifies the required entry in the VTAM class-of-service table. Every modename that you supply, when you define a group of APPC sessions to CICS, must be matched by a VTAM LOGMODE name. All that is required in the VTAM LOGMODE table are entries of the following form:

```
MODEENT LOGMODE=modename
MODEEND
```

An entry is also required for the LU services manager modeset (SNASVCMG):

```
MODEENT LOGMODE=SNASVCMG
MODEEND
```

If you plan to use autoinstall for single-session APPC terminals, additional information is required in the MODEENT entry. For programming information about coding the VTAM LOGON mode table, see the *CICS Transaction Server for z/OS V2.3 CICS Customization Guide*.

CICS definitions

The CICS region name we use is CICSPS8K and it is directly connected to the same DB2 subsystem that one of the IMS regions will use. The other IMS region will connect to a different DB2.

Add CICS APPC support

Define a CICS CONNection and a CICS SESSion to a CICS group of your choice. This example shows the definitions being created by the CICS Resource Definition Online method using the CICS transaction CEDA. You can also use the CICS batch utility DFHCSDUP to create the same definitions.

When you have completed the definitions you will have to ADD your groupname to the CICS1 grplist. This will install and activate your new definitions the next time you start CICS. To avoid a CICS restart, you can also issue a **`CEDA INSTALL GROUP(groupname)`** in order to activate your definitions dynamically.

Defining the CICS Connection

Example A-3: CICS RDO Definitions for APPC Connection

```
CEDA  View CONnection( APPC)
   CONnection     : APPC
   Group          : groupname
   DEscription    : APPC PROJECT
   CONNECTION IDENTIFIERS
    Netname        : SCSIM8HA
    INDsys         :
   REMOTE ATTRIBUTES
    REMOTESYSTem   :
    REMOTEName     :
    REMOTESYSNet  :
   CONNECTION PROPERTIES
    ACcessmethod         : Vtam              Vtam | IRc | INdirect | Xm
    PRotocol      : Appc              Appc | Lu61 | Exci
    Conntype      :                   Generic | Specific
    SInglesess     : No                No | Yes
    DAtastream     : User              User | 3270 | SCs | STrfield | Lms
    RECordformat         : U                 U | Vb
    Queuelimit     : No                No | 0-9999
    Maxqtime       : No                No | 0-9999
   OPERATIONAL PROPERTIES
    AUtoconnect    : No                No | Yes | All
    INService      : Yes               Yes | No
   SECURITY
    SEcurityname   :
    ATtachsec      : Local             Local | Identify | Verify | Persistent
                                       | Mixidpe
    BINDPassword         :                   PASSWORD NOT SPECIFIED
    BINDSecurity   : No                No | Yes
    Usedfltuser    : No                No | Yes
   RECOVERY
    PSrecovery     : Sysdefault        Sysdefault | None
    Xlnaction      : Keep              Keep | Force
```

Defining the CICS Session

Example A-4: CICS RDO Definitions for APPC Session

```
   CEDA  View Sessions( CICSPA8K)
    Sessions       : CICSPA8K
    Group          : groupname
    DEscription    : CICSPA8K APPC PROJECT
   SESSION IDENTIFIERS
    Connection     : APPC
    SESSName      :
    NETnameq       :
    MOdename       : APPCHOST
   SESSION PROPERTIES
    Protocol       : Appc              Appc | Lu61 | Exci
    MAximum        : 002 , 002         0-999
    RECEIVEPfx     :
    RECEIVECount         :                   1-999
    SENDPfx        :
    SENDCount      :                   1-999
    SENDSize       : 04096             1-30720
    RECEIVESize          : 04096             1-30720
    SESSPriority   : 000               0-255
    Transaction    :
```

```
OPERATOR DEFAULTS
 OPERId        :
 OPERPriority  : 000              0-255
 OPERRsl       : 0                                              0-24,...
 OPERSecurity        : 1                                               1-64,...
PRESET SECURITY
 USERId        :
OPERATIONAL PROPERTIES
 Autoconnect   : No               No | Yes | All
 INservice     :                  No | Yes
 Buildchain    : Yes              Yes | No
 USERArealen         : 000               0-255
 IOarealen      : 00000 , 00000    0-32767
 RELreq        : No               No | Yes
 DIscreq       : No               No | Yes
 NEPclass      : 000              0-255
RECOVERY
 RECOVOption         : Sysdefault        Sysdefault | Clearconv | Releasesess
                                  | Uncondrel | None
 RECOVNotify        : None               None | Message | Transaction
```

Update CICS VTAM APPL definition

CICS LUs need to be configured by specifying APPC, MODETAB and DLOGMOD parameters. APPL definition requires a MODETAB and an entry (MODEENT) within the MODETAB that APPC uses as default for LU6.2 conversations. Example A-5 shows the definition we used in our test. It is for a CICS LU with protected conversation capability.

Example A-5: Definition macro for the CICS LU

```
SCSC8KVT       APPL  ACBNAME=SCSC8KVT,                                   X
               APPC=NO,                                                  X
               AUTH=(ACQ,VPACE,PASS),                                    X
               VPACING=0,                                                X
               EAS=5000,                                                 X
               PARSESS=YES,                                              X
               MODETAB=LOGMODES,                                         X
               DLOGMOD=APPCHOST,                                         X
               SONSCIP=YES
```

Example A-5 shows the definition for a CICS LU with protected conversation capability.

The DLOGMOD parameter can also be defined at CICS level, within the MODENAME parameter in session definition, and specified at application level, within the MODE_NAME parameter on ATBALLC API call.

Attention: The logmode entry used by the IMS and CICS systems needs to be aligned.

Note: Although it seems to contradict the aim of this exercise, APPC=NO must be coded on the CICS VTAM APPL.

For further information, refer to *CICS Intercommunication Guide*, *z/OS V1R2.0 MVS Writing TPs for APPC/MVS*, *CICS Resource Definition Guide*.

ACF/VTAM definition for CICS

You define your CICS system to ACF/VTAM, including the following operands in the VTAM APPL statement. The VTAM APPL parameter indicates that the mode table is LOGMODES (it usually resides in SYSx.VTAMLIB).

- **MODETAB=logon-mode-table-name**

  Specifies the VTAM logon mode table that contains your customized logon mode entries. You can omit this operand if you choose to add your MODEENT entries to the IBM default logon mode table (without renaming it).

- **AUTH=(ACQ,SPO,VPACE[,PASS])**

  ACQ is required to allow CICS to acquire LU type 6 sessions. SPO is required to allow CICS to issue the **`MODIFY vtamname USERVAR`** command (For further information about the significance of USERVARs, see the *CICS/ESA® 3.3 CICS XRF Guide*). VPACE is required to allow pacing of the intersystem flows. PASS is required if you intend to use the EXEC CICS ISSUE PASS command, which passes existing terminal sessions to other VTAM applications.

- **VPACING=number**

  This operand specifies the maximum number of normal-flow requests that another logical unit can send on an intersystem session before waiting to receive a pacing response. Take care when selecting a suitable pacing count. Too low a value can lead to poor throughput because of the number of line turnarounds required. Too high a value can lead to excessive storage requirements.

- **EAS=number**

  This operand specifies the number of network-addressable units that CICS can establish sessions with. The number must include the total number of parallel sessions for this CICS system.

- **PARSESS=YES**

  This option specifies LU type 6 parallel session support.

- **SONSCIP=YES**

  This operand specifies session outage notification (SON) support. SON enables CICS, in particular cases, to recover a failed session without requiring operator intervention.

- **APPC=NO**

  For ACF/VTAM Version 3.2 and above, you need to direct CICS to use VTAM macros. CICS does not issue the APPCCMD macro. The use of APPC=NO drives CICS to use VTAM macros instead of APPCCMD macros.

For further information about the VTAM APPL statement, refer to *OS/390 eNetwork Communications Server: SNA Resource Definition Reference*.

Defining the CICS Cobol inbound program to the CICS region

This example shows the definitions being created by the CICS Resource Definition Online method using the CICS transaction CEDA. You can also use the CICS batch utility DFHCSDUP to create the same definitions.

Example A-6: CEDA Definition for Inbound Program

```
CEDA  DEFine PROGram( CICSPG1  )
   PROGram        : CICSPG1
   Group          : groupname
   DEscription    : TEST PROGRAM FOR APPC CONVERSATION
   Language       :                    CObol | Assembler | Le370 | C | Pli
```

```
   RELoad         : No                No | Yes
   RESident       : No                No | Yes
   USAge         : Normal            Normal | Transient
   USElpacopy     : No                No | Yes
   Status         : Enabled           Enabled | Disabled
   RSl            : 00                0-24 | Public
   CEdf           : Yes               Yes | No
   DAtalocation   : Below            Below | Any
   EXECKey        : User              User | Cics
   COncurrency    : Quasirent         Quasirent | Threadsafe
   REMOTE ATTRIBUTES
    DYnamic        : No                No | Yes
    REMOTESystem   :
    REMOTEName     :
    Transid        :
    EXECUtionset   : FullapiFullapi | Dplsubset
   JVM ATTRIBUTES
    JVM            : No       No | Yes
    JVMClass       :
                     :
             :
             :
             :
   JVMProfile     : DFHJVMPR
   JAVA PROGRAM OBJECT ATTRIBUTES
    Hotpool        : No                No | Yes
```

Defining the CICS inbound transaction to the CICS region

This example shows the definitions being created by the CICS Resource Definition Online method using the CICS transaction CEDA. You can also use the CICS batch utility DFHCSDUP to create the same definitions.

Example A-7: Inbound transaction definition entry for CICS

```
CEDA  DEFine TRANSaction( IMP1 )
   TRANSaction    : IMP1
   Group          : groupname
   DEscription    : TEST TRAN FOR APPC RED BOOK PROCESSING
   PROGram        : CICSPG1
   TWasize        : 00000             0-32767
   PROFile        : DFHCICST
   PArtitionset   :
   STAtus         : Enabled           Enabled | Disabled
   PRIMedsize     : 00000             0-65520
   TASKDATALoc    : Below             Below | Any
   TASKDATAKey    : User              User | Cics
    STOrageclear   : No                No | Yes
    RUnaway        : System            System | 0 | 500-2700000
    SHutdown       : Disabled          Disabled | Enabled
    ISolate        : Yes               Yes | No
    Brexit         :
   REMOTE ATTRIBUTES
    DYnamic        : No                No | Yes
    ROutable       : No                No | Yes
    REMOTESystem   :
    REMOTEName     :
    TRProf         :
    Localq         :                   No | Yes
   SCHEDULING
```

```
  PRIOrity       : 001               0-255
  TClass         : No                No | 1-10
  TRANClass      : DFHTCL00
 ALIASES
  ALias          :
  TASKReq        :
  XTRanid        :
  TPName         :
                 :
  XTPname        :
              :
                 :
 RECOVERY
  DTimout        : No                No | 1-6800 (MMSS)
  RESTart        : No                No | Yes
  SPurge         : No                No | Yes
  TPUrge         : No                No | Yes
  DUmp           : Yes               Yes | No
  TRACe          : Yes               Yes | No
  COnfdata       : No                No | Yes
  Otstimeout     : No                No | 0-240000 (HHMMSS)
 INDOUBT ATTRIBUTES
  ACtion         : Backout           Backout | Commit
  WAIT           : Yes               Yes | No
  WAITTime       : 00 , 00 , 00      0-99 (Days,Hours,Mins)
  INdoubt        : Backout           Backout | Commit | Wait
 SECURITY
  RESSec         : No                No | Yes
  CMdsec         : No                No | Yes
  Extsec         : No                No | Yes
  TRANSec        : 01                1-64
  RSl            : 00                0-24 | Public
```

Defining the CICS Cobol outbound program to the CICS region

This example shows the definitions being created by the CICS Resource Definition Online method using the CICS transaction CEDA. You can also use the CICS batch utility DFHCSDUP to create the same definitions.

Example A-8: CEDA Definition for Outbound Program

```
CEDA  View PROGram( GTCICS02 )
 PROGram        : GTCICS02
 Group          : IVP8CX02
 DEscription    : TEST OUTBOUND PROGRAM
 Language       :                  CObol | Assembler | Le370 | C | Pli
 RELoad         : No               No | Yes
 RESident       : No               No | Yes
 USAge          : Normal           Normal | Transient
 USElpacopy     : No               No | Yes
 Status         : Enabled          Enabled | Disabled
 RSl            : 00               0-24 | Public
 CEdf           : Yes              Yes | No
 DAtalocation   : Below            Below | Any
 EXECKey        : User             User | Cics
 COncurrency    : Quasirent        Quasirent | Threadsafe
REMOTE ATTRIBUTES
 DYnamic        : No               No | Yes
```

```
 REMOTESystem   :                                                           REMOTEName
:
 Transid        :
 EXECUtionset   : Fullapi           Fullapi | Dplsubset
JVM ATTRIBUTES
 JVM            : No                No | Yes
 JVMClass       :
 (Mixed Case)   :
                :
                :
                :
 JVMProfile     : DFHJVMPR          (Mixed Case)
JAVA PROGRAM OBJECT ATTRIBUTES
 Hotpool        : No                No | Yes
```

Defining the CICS outbound transaction to the CICS region

This example shows the definitions being created by the CICS Resource Definition Online method using the CICS transaction CEDA. You can also use the CICS batch utility DFHCSDUP to create the same definitions.

Example A-9: Outbound transaction definition entry for CICS

```
CEDA  View TRANSaction( GTCX )
 TRANSaction    : GTCX
 Group          : IVP8CX02
 DEscription    : TEST TRANSACTION
 PROGram        : GTCICS02
 TWasize        : 00000             0-32767
 PROFile        : DFHCICST
 PArtitionset   :
 STAtus         : Enabled           Enabled | Disabled
 PRIMedsize     : 00000             0-65520
 TASKDATALoc    : Below             Below | Any
 TASKDATAKey    : User              User | Cics
 STOrageclear   : No                No | Yes
 RUnaway        : System            System | 0 | 500-2700000
 SHutdown       : Disabled          Disabled | Enabled
 ISolate        : Yes               Yes | No
 Brexit         :
REMOTE ATTRIBUTES                                             DYnamic       : No
No | Yes
 ROutable       : No                No | Yes
 REMOTESystem   :
 REMOTEName     :
 TRProf         :
 Localq         :                   No | Yes
SCHEDULING
 PRIOrity       : 001               0-255
 TClass         : No                No | 1-10
 TRANClass      : DFHTCL00
ALIASES
 ALias          :
 TASKReq        :
 XTRanid        :
 TPName         :
                :
 XTPname      :                                         :
                :
```

```
RECOVERY
 DTimout        : No                 No | 1-6800 (MMSS)
 RESTart        : No                 No | Yes
 SPurge         : No                 No | Yes
 TPUrge         : No                 No | Yes
 DUmp           : Yes                Yes | No
 TRACe          : Yes                Yes | No
 COnfdata       : No                 No | Yes
 Otstimeout     : No                 No | 0-240000 (HHMMSS)
INDOUBT ATTRIBUTES
 ACtion         : Backout            Backout | Commit
 WAIT           : Yes                Yes | No
 WAITTime       : 00 , 00 , 00       0-99 (Days,Hours,Mins)
 INdoubt        : Backout            Backout | Commit | Wait
SECURITY                                                RESSec        : No
No | Yes
 CMdsec         : No                 No | Yes
 Extsec         : No                 No | Yes
 TRANSec        : 01                 1-64
 RSl            : 00                 0-24 | Public
```

Dynamically installing the CICS group

This example shows the definitions being created by the CICS Resource Definition Online method using the CICS transaction CEDA. You can also use the CICS batch utility DFHCSDUP to create the same definitions.

To dynamically install the CICS group issue the following command:

```
CEDA INSTALL GROUP(groupname)
```

This will activate the definitions in the group until CICS is recycled. In order to ensure that the definitions are included during the next start of CICS you must add the CICS group to the relevant CICS list.

Adding the CICS Group to the CICS List

This example shows the definitions being created by the CICS Resource Definition Online method using the CICS transaction CEDA. You can also use the CICS batch utility DFHCSDUP to create the same definitions.

Each CICS region can be assigned up to 3 CICS group lists. These lists are defined in the System Initialization Table.

To add a CICS Group to the CICS list which will be included during the next re-cycle of a CICS region issue the following command:

```
CEDA ADD GROUP(groupname) TO LIST(cicsname)
```

Optional: Completing tasks 1 through 5 using the DFHCSDUP utility

Rather than issue CICS commands dynamically, you can create the definitions for the APPC CICS Application using the CICS-supplied batch utility DFHCSDUP. Example A-10 is a sample of the JCL.

Example A-10: DFHCSDUP Sample

```
//Jobname .....
//STEP1   EXEC PGM=DFHCSDUP
//STEPLIB  DD  DSN=CICS.DFHLOAD,DISP=SHR
//DFHCSD   DD  UNIT=3390,DISP=SHR,DSN=CICS.DFHCSD
```

```
//OUTDD    DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
*
* Define Inbound Program
*
   DEFINE PROGRAM(CICSPG1) GROUP(groupname)
*
* Define Inbound Transaction
*
   DEFINE TRANS(IMP1) PROG(CICSPG1) GROUP(groupname)
*
* Define Outbound Program
*
   DEFINE PROGRAM(GTCICS02) GROUP(groupname)
*
* Define Inbound Transaction
*
   DEFINE TRANS(GTCX) PROG(GTCICS02) GROUP(groupname)

*
* ADD Group to CICS Group List
*
   ADD GROUP(groupname) TO LIST(cicslist)
/*
```

For further information regarding DFHCSDUP, refer to *CICS TS Operations & Utilities Guide.*

Compiling the source CICS sample Cobol program

The JCL in Example A-11 can be used as the basis for compiling, linking, and binding the supplied sample CICS Inbound Cobol Program source. Refer to Appendix B, “APPC exploiter sample source code” on page 155 for program source.

Example A-11: Cobol compile sample JCL

```
//COBLPGMS JOB (999,POK),'CONWAY',CLASS=A,MSGCLASS=H,
// NOTIFY=&SYSUID,TIME=1440,REGION=0M
/*JOBPARM S=SC61
//**---------------------------------------------------------------*/
//** INSTRUCTIONS:                                                 */
//** 1. IF YOU USE A PROGRAM NAME OF YOUR CHOSING THEN YOU WILL NEED */
//**    TO CHANGE THE SET MEM=PGMNAME TO POINT TO YOUR PROGRAM NAME. */
//**                                                               */
//** 2. IF YOU WISH TO USE A SEPARATE BIND STEP THEN DELETE OR      */
//**    COMMENT OUT THE FINAL STEP IN THIS JOB.                    */
//**                                                               */
//** 3. IF USING THE BIND01 STEP THEN ENSURE THAT THE PLAN AND MEMBER*/
//**    NAMES MATCH THAT OF YOUR CHOSEN PROGRAM NAME.              */
//**                                                               */
//** 4. UPDATE THE BIND STEP TO POINT TO YOUR DB2 SYSID.           */
//**                                                               */
//** 5. UPDATE THE FOLLOWING LIBRARY NAMES TO MATCH YOUR SITE       */
//**    NAMING STANDARDS:                                          */
//**           IGY.SIGYCOMP                                        */
//**           RC62.BARIAPPC.DBRMLIB.DATA                          */
//**           DB8KU.RUNLIB.LOAD                                   */
//**           DB8K8.SDSNLOAD                                      */
//**           CICSTS23.CICS.SDFHCOB                               */
```

```
//**           CICSTS23.CICS.SDFHLOAD                                */
//**           RC62.GRAHAMD.APPC.DATA                                */
//**           SYS1.CSSLIB                                           */
//**           SYS1.LINKLIB                                          */
//**           CEE.SCEELKED                                          */
//**-----------------------------------------------------------------*/
/*JOBPARM   L=100
//*
//*       DSNHCOB2 - COMPILE AND LINKEDIT A COBOL PROGRAM
//*
// SET WSPC=500
// SET MEM=CICSPG1
//*
//*  ---------------------------------------------------------
//*            PRECOMPILE THE COBOL PROGRAM
//*
//PC      EXEC PGM=DSNHPC,PARM='HOST(COB2),SOURCE,APOST,QUOTE,QUOTESQL'
//DBRMLIB   DD DSN=RC62.BARIAPPC.DBRMLIB.DATA(&MEM),DISP=OLD
//STEPLIB   DD DISP=SHR,DSN=DB8K8.SDSNLOAD
//          DD DSN=IGY.SIGYCOMP,DISP=SHR
//SYSCIN    DD DSN=&&DSNHOUT,DISP=(MOD,PASS),UNIT=SYSDA,
//             SPACE=(800,(&WSPC,&WSPC))
//SYSLIB    DD DISP=SHR,DSN=CICSTS23.CICS.SDFHCOB
//*         DD DSN=CRTWDATA.CRTW.DCLGEN,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSTERM   DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSUT1    DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=SYSDA
//SYSUT2    DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=SYSDA
//SYSIN     DD DISP=SHR,DSN=RC62.GRAHAMD.APPC.DATA(&MEM)
//*
//*  ---------------------------------------------------------
//*
//CICSTRN EXEC PGM=DFHECP1$,PARM='NOSOURCE,SP'
//STEPLIB   DD DSN=CICSTS23.CICS.SDFHLOAD,DISP=SHR
//          DD DISP=SHR,DSN=CICSTS23.CICS.SDFHCOB
//          DD DSN=IGY.SIGYCOMP,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DSN=&&DSNHOUT,DISP=(OLD,DELETE)
//SYSPUNCH  DD DSN=&&SYSCIN,DISP=(,PASS),UNIT=SYSDA,DCB=BLKSIZE=400,
//             SPACE=(CYL,(1,1))
//*
//*  ---------------------------------------------------------
//*
//*            COMPILE THE COBOL PROGRAM IF THE PRECOMPILE
//*            RETURN CODE IS 4 OR LESS
//*
//COB     EXEC PGM=IGYCRCTL,COND=(4,LT,PC),
//           PARM='NODYNAM,LIB,OBJECT,RENT,MAP,XREF'
//STEPLIB   DD DSN=IGY.SIGYCOMP,DISP=SHR
//          DD DSN=CICSTS23.CICS.SDFHLOAD,DISP=SHR
//          DD DSN=CICSTS23.CICS.SDFHCOB,DISP=SHR
//SYSLIN   DD  DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
//             SPACE=(800,(&WSPC,&WSPC))
//SYSIN     DD DSN=&&SYSCIN,DISP=(OLD,DELETE)
//SYSLIB    DD DSN=CICSTS23.CICS.SDFHCOB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSUT1   DD  SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=SYSDA
//SYSUT2   DD  SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=SYSDA
```

```
//SYSUT3   DD  SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=SYSDA
//SYSUT4   DD  SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=SYSDA
//SYSUT5   DD  SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=SYSDA
//SYSUT6   DD  SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=SYSDA
//SYSUT7   DD  SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=SYSDA
//*
//*  --------------------------------------------------------
//*
//*            LINKEDIT IF THE PRECOMPILE AND COMPILE
//*            RETURN CODES ARE 4 OR LESS
//*
//LKED    EXEC PGM=IEWL,PARM='XREF',
//             COND=((4,LT,COB),(4,LT,PC))
//*  NOTE THIS LIBRARY CONTAINS THE IGY MEMBERS.
//SYSLIB   DD DSN=SYS1.CSSLIB,DISP=SHR
//         DD DSN=SYS1.LINKLIB,DISP=SHR
//         DD DSN=IGY.SIGYCOMP,DISP=SHR
//         DD DISP=SHR,DSN=CICSTS23.CICS.SDFHLOAD
//*        DD DISP=SHR,DSN=CICS.TS22.SYSPLEXB.TBSMLOAD
//         DD DISP=SHR,DSN=DB8K8.SDSNLOAD
//         DD DISP=SHR,DSN=CEE.SCEELKED
//SYSLIN   DD  DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD DDNAME=SYSIN
//         DD DSN=CICSTS23.CICS.SDFHCOB(DFHEILIC),DISP=SHR
//SYSLMOD DD DSN=RC62.BARIAPPC.LOAD(&MEM),
//        DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSUT1   DD  SPACE=(1024,(50,50)),UNIT=SYSDA
//SYSIN DD *
  INCLUDE SYSLIB(DSNCLI)
  INCLUDE SYSLIB(DFHCPLC)
  INCLUDE SYSLIB(DFHCPLRR)
//*
//BIND01   EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB  DD DSN=DB8K8.SDSNLOAD,DISP=SHR
//DBRMLIB  DD DSN=RC62.BARIAPPC.DBRMLIB.DATA,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*

//SYSOUT   DD SYSOUT=*
//REPORT   DD SYSOUT=*
//SYSIN    DD *
 GRANT       EXECUTE ON PLAN CICSPG1 TO PUBLIC;
//SYSTSIN DD *
 DSN SYSTEM(DB8K)
 BIND PLAN(CICSPG1) MEMBER(CICSPG1,GTCICS02) -
 ACT(REP) ISO(CS) ENCODING(EBCDIC)
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA81) -
      LIB('DB8KU.RUNLIB.LOAD')
 END
/*
```

Note: Modify the JCL to meet your site standards. Follow the instructions contained within the header of the JCL job.

IMS definitions

Two IMS subsystems are established with the ability to converse using APPC Protected Mode conversations. The IMS subsystems will be connected to one DB2 subsystem. Within this section the IMS subsystems are referred to as IMSH and IMSI.

IMS Gen process to generate two IMS TM/DB systems supporting APPC/IMS

a. APPC/IMS support is provided by "FMID JMK8802 - Transaction Manager, APPC/LU Manager." FMID JMK8802 is a required FMID for a TM/DB system.

> **Reference:** See "Table 3. FMID Installation Requirements" in *IMS Installation Volume 1: Installation Verification Version 8*.

b. Generate the IMSH and IMSI systems as TM/DB subsystems.

c. Our two IMS systems share a common CQS. (This is optional and it may be much simpler to set up IMS systems without a CQS).

Build and start the IMSH and IMSI TM/DB subsystems

We refer to these two IMS subsystems as IMSH and IMSI. These will be directly connected to the same DB2 subsystem.

a. Set APPC/IMS options in the IMS PROCLIB member DFSPBxxx.

   APPC=Y (turns on APPC)

   APPCSE=N (turns off security, otherwise RACF profiles will be required)

b. Set the APPC/IMS Timeout option in the IMS PROCLIB member DFSDCxxx.

   APPCIOT=02(set the timeout to 2 minutes so it does not wait indefinitely)

c. Set RRS/IMS option in IMS PROCLIB member DFSPBxxx.

   RRS=Y (turns on RRS)

Install and test the standard IMS IVP

a. Install the IMS Sample Application.

   Complete the steps required to install the IMS Sample Application, specifically the PART transaction.

> **Reference:** For additional information, refer to *IMS Installation Volume 1: Installation Verification Version 8*, Section 3.3, IMS Sample Application.

b. Install the IVPREXX application.

   Complete the steps required to install the IVPREXX function of the IMS Adapter for REXX.

Reference:

For further details, refer to *IMS Application Programming: Transaction Manager Version 8*, Section 3.0, IMS Adapter for REXX (an extract follows).

3.2.5 IVPREXX: MPP/IFP Front End for General Exec Execution.

> The IVPREXX exec is a front-end generic exec that is shipped with IMS as part of the IVP. It runs other execs by passing the exec name to execute after the TRANCODE (IVPREXX). For further details on IVPREXX, see "IVPREXX Sample Application" in topic 3.1.2.2.

For the latest version of the IVPREXX source code, see the IMS.ADFSEXEC distribution library; member name is IVPREXX.

c. Test the PART transaction in both IMSH and IMSI.

Execute the PART transaction as indicated in the following reference.

Reference:

IMS Installation Volume 1: Installation Verification Version 8, 3.3.3 Sample Transactions

Test the IVPREXX function in both IMSH and IMSI The following Subtopics are simple tests of the IVPREXX functionality.

IMS Application Programming: Transaction Manager Version 8, Document Number SC27-1289.

3.1.2.2 IVPREXX Sample Application

Subtopics

- 3.1.2.2.1 IVPREXX Example 1
- 3.1.2.2.2 IVPREXX Example 2
- 3.1.2.2.3 IVPREXX Example 3
- 3.1.2.2.4 IVPREXX Example 4

Install and test the IMS section of the DB2 IVP

a. Install the IMS DB2 IVP into the DB2 subsystem. Note the following hints/tips extracted from:

http://www-1.ibm.com/support/docview.wss?uid=swg21024061

DB2 IVP Hints and Tips:

1. Always run the IVP jobs with a user ID that has SYSADM install authority. This ensures that you are authorized to perform even very restricted operations, such as creating a DB2 stogroup for the sample database.
2. Use the same user ID to run each of the sample jobs. Some of the IVP jobs, notably DSNTEJ1, create synonyms for sample objects that will be referenced in other jobs. If the user ID changes, these synonyms do not resolve correctly.
3. Always begin with IVP job DSNTEJ1. Don't run job DSNTEJ0, which drops the sample objects, unless you are starting over.

The following are the jobs that are required for the IVP - IMS:

- DSNTEJ1 - Create sample tables. If job fails, run DSNTEJ0 for cleanup.
- DSNTEJ2C - Cobol common modules. Required for Phases 4/5.
- DSNTEJ2P - PLI common modules. Required for Phases 4/5.

  Check that member DSN8MPG contains the PQ44916 change, otherwise it will need to be modified to remove the OPTIONS (MAIN). Refer to PQ44916 (or get the source for this module from the DB2 V8 SDNSSAMP library.

  ```
  * 05/22/03 - FIX CODE HOLE CLOSED BY VA AND ENTERPRISE PL/I   PQ44916*
  DSN8MPG: PROC(MODULE, ICODE, OUTMSG) OPTIONS (MAIN);
  DSN8MPG: PROC(MODULE, ICODE, OUTMSG);
  ```

- DSNTEJ4C - Cobol modules for IMS application.
- DSNTEJ4P - PLI modules for IMS application.

Modify the Linkedit Parms in step PS08 to use AMODE=31,RMODE=ANY

```
//PH04PS08 EXEC PGM=IEWL,PARM='LIST,XREF,LET,AMODE=24,RMODE=24',
//PH04PS08 EXEC PGM=IEWL,PARM='LIST,XREF,LET,AMODE=31,RMODE=ANY',
```

Note 1: Ensure the DB2 IMS IVP PL/I compiles use the Enterprise PL/I Compiler (we are using V3.3.0).

Note 2: Ensure the DB2 IMS IVP COBOL compiles use the Enterprise COBOL Compiler (we are using V3.3.0).

b. Establish the IMSH and IMSI systems for test:

   i. Install the required IMS components and connections in both IMSH and IMSI.

      Add the DB2 IVP applications and transactions to IMS via an IMS Modblks gen.

      Member DSN8FIMS in prefix.SDSNSAMP contains information to assist in the definition step.

   ii. Perform the required steps to allow IMSH and IMSI to establish an ESAF connection with the DB2 system.

      - Define DB2 to IMSH and IMSI via the SSM= parm in DFSPBxxx member.

      - Add DB2 load libraries to the STEPLIB DD of the IMS control region.

      - Add the IVP programs library to STEPLIB DD of their MPR JCL.

      - Add DB2 load libraries to DFSESL DD of the MPR JCL.

c. Test the DB2 IVP in both IMSH and IMSI to ensure data from the DB2 tables can be viewed.

Reference:

DB2 Universal Database for OS/390 and z/OS, Installation Guide, Version 7

2.8.10 Phase 4: Testing the IMS environment

Phase 4 installs the sample IMS transactions for both COBOL and PL/I. In the PL/I version, the phone application discussed in Phase 2 is also installed as an online transaction. For more information on the phone application, refer to "The phone application scenario" in topic 2.8.15.3.

Subtopics:

- 2.8.10.1 Jobs DSNTEJ4C and DSNTEJ4P
- 2.8.10.2 Starting an application in an IMS environment
- 2.8.10.3 Using the phone application in IMS

Allocate TP Profile data set

- SYS1.APPCTP

Note: Sample ATBTPVSM in SYSx.SAMPLIB contains the jobs to create the VSAM data set that holds APPC TP-profiles.   This needs to be tailored to create the IMS TP Profiles.

Allocate SIDEINFO data set

- SYS1.APPCSI

Note: Sample ATBSIVSM in SYSx.SAMPLIB contains the job to create the VSAM data set that holds APPC side-information. This needs to be tailored to create the IMS SideInfo.

Add IMS APPC LUs to VTAM

You have to modify your IMS LUs specifying ATNLOSS and SYNCLVL parameters. APPL definitions specify a MODETAB and an entry (MODEENT) within the MODETAB that APPC uses as default for LU6.2 conversations. The DLOGMOD parameter can also be specified at the application level, within the MODE_NAME parameter on a ATBALLC API call. Example A-12 show a definition we used in our test.

Example A-12: Definition macro for the IMS LU

```
SCSIM8I        VBUILD TYPE=APPL
SCSIMS8I       APPL  AUTH=(ACQ),EAS=1200,ACBNAME=SCSIMS8I
SCSIM8IA       APPL  ACBNAME=SCSIM8IA,                         X
               APPC=YES,                                       X
               ATNLOSS=ALL,                                    X
               AUTH=(ACQ),                                     X
               DLOGMOD=APPCHOST,                               X
               MODETAB=LOGMODES,                               X
               PARSESS=YES,                                    X
               SECACPT=NONE,                                   X
               SYNCLVL=SYNCPT
```

Example A-12 shows the definition for an IMS LU with protected conversation capability.

Notes:

1. The label for the APPC APPL e.g. SCSIMS8I, must be the same as the ACBNAME subparameter.
2. APPCLU62 and APPCMODE must be available in SYSx.VTAMLIB.

   These are added to SYSx.VTAMLIB during the APPC installation process. They are supplied in the SYSx.SAMPLIB member ATBLMODE.

   SYSx.SAMPLIB contains the following samples:

   – ATBLJOB JCL, VTAM logmode table link edit job.
   – ATBLMODE, VTAM logmode table for APPC.
3. ATNLOSS=ALL

   This is required to support IMS protected conversations.

IMS Administration Guide: Transaction Manager Version 8, 6.1.2.2.1 APPC as the Communications Manager

When APPC is the communications manager, RRS/MVS support is activated when a conversation is allocated with SYNCLVL=SYNCPT. This type of conversation is a protected conversation.

When SYNCLVL=SYNCPT is specified, APPC acquires a private context on behalf of IMS. IMS provides its resource manager name to APPC in its identity call. APPC provides the private context to IMS as the message header. IMS, using this context, then assumes the role of a participant in the two-phase commit process with the sync-point manager, RRS/MVS.

In addition to the SYNCLVL=SYNCPT, the keyword ATNLOSS=ALL must be specified in the VTAM definition file for whichever LUS the user wishes to enable for protected conversations.

For further information, refer to *z/OS V1R2.0 MVS Writing TPs for APPC/MVS*.

Add IMSH and IMSI APPC LUs to APPC

Add the following example shows the LUADD statements for APPCPMxx in the z/OS SYSx.PARMLIB for IMSH and IMSI.

Example A-13: LUADD IMS APPC

```
LUADD
      ACBNAME(SCSIMS8H)             /* APPC LU for IMS IMSH System   */
      SCHED(IMSH)                   /* IMSH transaction scheduler    */
      BASE                          /* Base LU                       */
      TPDATA(SYS1.APPCTP)      /* Repository for TP profiles    */
      TPLEVEL(SYSTEM)               /* Search level for TP profiles  */
      SIDEINFO                      /* Holds the side information    */
         DATASET(SYS1.APPCSI)
LUADD
      ACBNAME(SCSIMS8I)             /* APPC LU for IMS IMSI System   */
      SCHED(IMSI)                   /* IMSI transaction scheduler    */
      BASE                          /* Base LU                       */
      TPDATA(SYS1.APPCTP)      /* Repository for TP profiles    */
      TPLEVEL(SYSTEM)               /* Search level for TP profiles  */
      SIDEINFO                      /* Holds the side information    */
         DATASET(SYS1.APPCSI)
```

Reference:

IMS Administration Guide: Transaction Manager, Version 8

- 6.2.4.4 Side information - Outbound

  APPC/MVS side information supplies destination information, such as the name of the partner program, the name of the LU at the partner's node, and the logon mode name. CPI Communications provides a way to use system-defined values for these required fields; these system-defined values are the side information. This information can be used by an IMS application program allocating (establishing) an APPC conversation using CPI Communications, an IMS LU 6.2 descriptor, a DL/I change call (CHNG), or a DFSAPPC message switch.

  System programmers supply and maintain the side information for CPI Communications programs.

- 6.2.4.5 PARMLIB Member

  The APPC address space uses the APPCPMxx member of SYS1.PARMLIB. Define IMS as a local APPC component LU that is controlled by the APPC address space. The scheduler name is the same IMSID used in the IMSCTRL macro. When IMS identifies to APPC, it passes its IMSID as the scheduler name SCHED(IMSH) in APPC member APPCPMxxx.

Define the IMS transactions and programs to the IMSH system

Add the following definitions to the IMSH system via a MODBLKS Gen.

Example A-14: IMS transaction and program definitions

```
*+++ I M S H  definitions +++++++++++++++++++++++++++++++++++++++++++++

*        ----------------------------------------------
*    (1) IMN1 PLI TRANSACTION TO ALLOCATE A CONVERSATION
*        WITH IMN5 AND INVOKE IMS2IMP IMPLICITLY.
*           - MODIFIED STANDARD TRANSACTION
*       ----------------------------------------------
         APPLCTN GPSB=IMS1PG1,PGMTYPE=TP
         TRANSACT CODE=IMS1TR1,MSGTYPE=(,RESPONSE),INQUIRY=NO,MODE=SNGL
*
*        ----------------------------------------------
*    (2) IMN1 PLI TRANSACTION TO ALLOCATE A CONVERSATION
*        WITH IMN5 AND INVOKE IMS2EXP EXPLICITLY.
*           - MODIFIED STANDARD TRANSACTION
*        ----------------------------------------------
         APPLCTN GPSB=IMS1PG2,PGMTYPE=TP
         TRANSACT CODE=IMS1TR2,MSGTYPE=(,RESPONSE),INQUIRY=NO,MODE=SNGL
*
*        ----------------------------------------------
*    (3) IMN1 PLI TRANSACTION TO ALLOCATE A CONVERSATION
*        WITH CICS AND INVOKE CICS TRAN EXPLICITLY.
*           - MODIFIED STANDARD TRANSACTION
*        ----------------------------------------------
         APPLCTN GPSB=IMS1PG3,PGMTYPE=TP
         TRANSACT CODE=IMS1TR3,MSGTYPE=(,RESPONSE),INQUIRY=NO,MODE=SNGL
*
*+++ I M S I  definitions +++++++++++++++++++++++++++++++++++++++++++++
*
*        ----------------------------------------------
*    (4) IMN5 PLI TRAN/PROGRAM SCHEDULED BY A
```

```
*       CONVERSATION INITIATED BY IMN1TR1 in IMN1
*          - STANDARD TRANSACTION
*       -----------------------------------------------
        APPLCTN GPSB=IMS2IMP,PGMTYPE=TP
        TRANSACT CODE=IMS2IMP,MSGTYPE=(,RESPONSE),INQUIRY=NO,MODE=SNGL
*
*       -----------------------------------------------
*    (5) IMN5 PLI EXPLICIT PROGRAM SCHEDULED IN RESPONSE
*       TO A CONVERSATION INITIATED BY IMS1TR2 in IMN1
*          - CPI DRIVEN PROGRAM (NO TRAN DEFINED IN GEN)
*       -----------------------------------------------
        APPLCTN GPSB=IMS2EXP
*
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Add IMS2EXPD to the SIDEINFO data set used by IMSH

Tailor the JCL; refer to the member prolog for details. When complete, submit the JCL and check the return code for a successful completion.

Example A-15: Sample JCL for IMSH sideinfo

```
//@APPCEXP JOB 'ADD IMS2EXPD SIDEINFO',
//         CLASS=A,
//         MSGCLASS=X,
//         NOTIFY=&SYSUID
//*
//*-----------------------------------------------------------------/
//*                                                                 |
//* Add IMS2EXPD to the IMSH's SIDEINFO dataset.                    |
//*                                                                 |
//*-----------------------------------------------------------------+
//*                                                                 |
//* Tailor the JCL;                                                 |
//*                                                                 |
//*  1) Modify the STEPLIB to point to your IMS RESLIB dataset.     |
//*                                                                 |
//*  2) Modify the SYSSDLIB ddname to point to the SIDEINFO dataset |
//*     used by IMSH's APPC Scheduler.                              |
//*                                                                 |
//*  3) Modify the SYSIN input PARTNER_LU parameter value SCSIM8IA  |
//*     to the name of the VTAM LU of the IMSI APPC Scheduler.      |
//*                                                                 |
//*-----------------------------------------------------------------+
//*                                                                 |
//* Use the APPC ATBSDFMU utility to add the IMS2EXPD symbolic      |
//* destination to IMSH's SIDEINFO dataset. The symbolic destination |
//* IMS2EXPD is coded as a subparameter on the APPC allocate        |
//* conversation verb in the module IMS1PS2. It determines the      |
//* Partner LU in the conversation and the TP Profile profile to    |
//* be used to schedule the transaction in the IMSI system.         |
//*                                                                 |
//* NOTE: as we want IMSI to use EXPLICT scheduling we must also    |
//*       add the TP Profile IMS2EXP_PROFILE to IMSI's TP Profile   |
//*       dataset. (member @APPCTP1 has the job to do this)         |
//*                                                                 |
//* Where:                                                          |
//*                                                                 |
//*  TPNAME(IMS2EXP_PROFILE) - name of the TP Profile in IMSI's     |
//*                            TP Profile dataset. It contains the  |
//*                            information needed to EXPLICITLY     |
```

```
//*                            schedule a transaction.             |
//*  MODENAME(APPCHOST)      - APPCHOST is the mode table entry      |
//*                            within the LOGMOMES Mode Table.       |
//*  PARTNER_LU(SCSIM8IA)    - The VTAM LU of IMSI's APPC Scheduler. |
//*                                                                  |
//*------------------------------------------------------------------+
//*                                                                  |
//* MVS command to determine which SIDEINFO dataset the IMSH APPC    |
//* Scheduler is using;                                              |
//*                                                                  |
//* D APPC,LU,LLUN=SCSIM8HA                                          |
//*                                                                  |
//* ATB121I  14.04.02  APPC DISPLAY 798                              |
//*   ACTIVE LU'S    OUTBOUND LU'S    PENDING LU'S  TERMINATING LU'S |
//*      00001           00000           00000           00000       |
//*   SIDEINFO=SYS1.APPCSI                                           |
//*                                                                  |
//*------------------------------------------------------------------+
//STEP     EXEC PGM=ATBSDFMU
//STEPLIB  DD   DISP=SHR,DSN=IMS.V810.SDFSRESL  <<= IMS Reslib
//SYSPRINT DD   SYSOUT=*
//*
//SYSSDLIB DD   DISP=SHR,DSN=SYS1.APPCSI        <<= SIDEINFO dataset
//*
//SYSSDOUT DD   SYSOUT=*
//*  SIADD
//SYSIN    DD   *
  SIDELETE
     DESTNAME(IMS2EXPD)
  SIADD
     DESTNAME(IMS2EXPD)
     TPNAME(IMS2EXP_PROFILE)
     MODENAME(APPCHOST)
     PARTNER_LU(SCSIM8IA)
/*
//*
```

Add IMS2DEST to the SIDEINFO data set used by IMSH

Tailor the member @APPCIMP; refer to the member prolog for details. When complete, submit the JCL and check the return code for a successful completion.

Example A-16: Sample JCL for IMSH destination sideinfo

```
//@APPCIMP JOB 'ADD CICS SIDEINFO',
//         CLASS=A,
//         MSGCLASS=X,
//         NOTIFY=&SYSUID
//*
//*------------------------------------------------------------------/
//*                                                                  |
//* Add IMS2DEST to the IMSH's SIDEINFO dataset.                     |
//*                                                                  |
//*------------------------------------------------------------------+
//*                                                                  |
//* Tailor the JCL;                                                  |
//*                                                                  |
//*  1) Modify the STEPLIB to point to your IMS RESLIB dataset.      |
//*                                                                  |
//*  2) Modify the SYSSDLIB ddname to point to the SIDEINFO dataset  |
//*     used by IMSH's APPC Scheduler.                               |
```

```
//*                                                                  |
//*  3) Modify the SYSIN input PARTNER_LU parameter value SCSIM8IA   |
//*     to the name of the VTAM LU of the IMSI APPC Scheduler.       |
//*                                                                  |
//*------------------------------------------------------------------+
//*                                                                  |
//* Use the APPC ATBSDFMU utility to add the IMS2DEST symbolic       |
//* destination to IMSH's SIDEINFO dataset. The symbolic destination |
//* IMS2DEST is coded as a subparameter on the APPC allocate         |
//* conversation verb in the module IMS1PS1. It determines the       |
//* Partner LU in the conversation and the transaction name which    |
//* will run in the partner IMSI system.                             |
//*                                                                  |
//* Where:                                                           |
//*                                                                  |
//*  TPNAME(IMS2IMP)      - IMS2IMP is the name of the IMSI          |
//*                         transaction which will be IMPLICITLY     |
//*                         scheduled in IMSI by the IMSI APPC       |
//*                         scheduler.                               |
//*  MODENAME(APPCHOST)   - APPCHOST is the mode table entry within  |
//*                         the LOGMOMES Mode Table.                 |
//*  PARTNER_LU(SCSIM8IA) - The VTAM LU of IMSI's APPC Scheduler.    |
//*                                                                  |
//* NOTE: as we want IMSI to use IMPLICIT scheduling for the         |
//*       IMS transaction IMS2IMP, we DO NOT add a TP Profile        |
//*       called IMS2DEST to IMSI's TP Profile dataset.              |
//*       (no TP Profile => IMPLICIT scheduling - very confusing).   |
//*                                                                  |
//*------------------------------------------------------------------+
//* MVS command to determine which SIDEINFO dataset the IMSH APPC    |
//* Scheduler is using;                                              |
//*                                                                  |
//* D APPC,LU,LLUN=SCSIM8HA                                          |
//*                                                                  |
//* ATB121I  14.04.02  APPC DISPLAY 798                              |
//*   ACTIVE LU'S    OUTBOUND LU'S    PENDING LU'S  TERMINATING LU'S |
//*      00001           00000           00000           00000       |
//*   SIDEINFO=SYS1.APPCSI                                           |
//*                                                                  |
//*------------------------------------------------------------------+
//STEP     EXEC PGM=ATBSDFMU
//STEPLIB  DD   DISP=SHR,DSN=IMS.V810.SDFSRESL  <<= IMS Reslib
//SYSPRINT DD   SYSOUT=*
//*
//SYSSDLIB DD   DISP=SHR,DSN=SYS1.APPCSI        <<= SIDE INFO DATASET
//*
//SYSSDOUT DD   SYSOUT=*
//*  SIADD
//SYSIN    DD   *
  SIDELETE
     DESTNAME(IMS2DEST)

  SIADD
     DESTNAME(IMS2DEST)
     TPNAME(IMS2IMP)
     MODENAME(APPCHOST)
     PARTNER_LU(SCSIM8IA)
/*
//*
```

Add CICSDEST to the SIDEINFO data set used by IMSH

Tailor the member @APPCICS - refer to the member prolog for details. When complete submit the JCL and check return code for a successful completion.

Example A-17: Sample JCL for CICS sideinfo

```
//@APPCICS JOB 'ADD CICS SIDEINFO',
//         CLASS=A,
//         MSGCLASS=X,
//         NOTIFY=&SYSUID
//*
//*------------------------------------------------------------------/
//*                                                                  |
//* Add CICSDEST to the IMSH's SIDEINFO dataset.                     |
//*                                                                  |
//*------------------------------------------------------------------+
//*                                                                  |
//* Tailor the JCL;                                                  |
//*                                                                  |
//*  1) Modify the STEPLIB to point to your IMS RESLIB dataset.      |
//*                                                                  |
//*  2) Modify the SYSSDLIB ddname to point to the SIDEINFO dataset  |
//*     used by IMSH's APPC Scheduler.                               |
//*                                                                  |
//*  3) Modify the SYSIN input PARTNER_LU parameter value SCSC8KVT   |
//*     to the name of the VTAM LU of the CICSPA8K Region.           |
//*                                                                  |
//*------------------------------------------------------------------+
//*                                                                  |
//* Use the APPC ATBSDFMU utility to add the CICSDEST symbolic       |
//* destination to IMSH's SIDEINFO dataset. The symbolic destination |
//* CICSDEST is coded as a subparameter on the APPC allocate         |
//* conversation verb in the module IMS1PS3. It determines the       |
//* Partner LU in the conversation and the transaction name which    |
//* will run in the partner CICSPA8K region.                         |
//*                                                                  |
//* Where:                                                           |
//*                                                                  |
//*  TPNAME(IMP1)          - IMP1 is the name of the CICS transaction |
//*                          will be scheduled by the CICS Region.   |
//*  MODENAME(APPCHOST)    - APPCHOST is the mode table entry within |
//*                          the LOGMOMES Mode Table.                |
//*  PARTNER_LU(SCSC8KVT)  - The VTAM LU of the CICSPA8K Region.     |
//*                                                                  |
//*------------------------------------------------------------------+
//*                                                                  |
//* MVS command to determine which SIDEINFO dataset the IMSH APPC    |
//* Scheduler is using;                                              |
//*                                                                  |
//* D APPC,LU,LLUN=SCSIM8HA                                          |
//*                                                                  |
//* ATB121I  14.04.02  APPC DISPLAY 798                              |
//*   ACTIVE LU'S    OUTBOUND LU'S    PENDING LU'S  TERMINATING LU'S |
//*      00001           00000           00000           00000       |
//*   SIDEINFO=SYS1.APPCSI                                           |
//*                                                                  |
//*------------------------------------------------------------------+
//STEP     EXEC PGM=ATBSDFMU
//STEPLIB  DD   DISP=SHR,DSN=IMS.V810.SDFSRESL
//SYSPRINT DD   SYSOUT=*
```

```
//*
//SYSSDLIB DD   DISP=SHR,DSN=SYS1.APPCSI          <<= SIDE INFO DATASET
//*
//SYSSDOUT DD   SYSOUT=*
//*  SIADD
//SYSIN    DD   *
  SIDELETE
     DESTNAME(CICSDEST)
  SIADD
     DESTNAME(CICSDEST)
     TPNAME(IMP1)
     MODENAME(APPCHOST)
     PARTNER_LU(SCSC8KVT)
/*
//*
```

APPC IMS Compile/Link instructions

This section contains details on how to compile and linkedit the IMS modules and their sub-modules. There are five load modules used:

- IMS1PG1
- IMS1PG2
- IMS1PG3
- IMS2IMI
- IMS2EXP

These are created from five main programs and fivew sub-modules (ten modules in all). Following are the details of how to change, compile, and linkedit each of the 10 modules.

Note: Before running any of the Compile and Link-Edit jobs, they must be tailored to your environment. Follow the instructions in the prolog of each member. For examples, refer to:

- “Example IMS PL/I Compile JCL” on page 142
- “Example IMS Link Edit JCL” on page 143

For *all* references to the source for *all* programs see Appendix B, “APPC exploiter sample source code” on page 155.

Refresh IMS regions

When you have completed your modifications you may have to refresh your IMS regions.

Attention:

To refresh the IMSH Regions issue the following from SDSF:

```
/IMSHDIS TRAN IMS1TR1
```

Note the transaction class nn of the IMS1TR1 transaction.

```
/IMSHDIS A
```

Displays IMS active regions. Note the Region name (for example, IMSHMP1) and numbers (for example, 999) of the regions that are able to transaction class nn.

```
/IMSHSTO REG 999
```

Stop each of the regions using the Stop command.

```
/IMSHSTA REG IMSHMP1
```

Restart each region using the Start command. For example, this command will cause IMS to submit the jcl for the message region IMSHMP1 to JES2.

Note: If you have more than one IMS region capable of processing the IMS1TR1 transaction class you will need to stop and start each of these regions.

Module IMS1PG1 structure (processes tran IMS1TR1)

The IMS1TR1 transaction is processed by the IMS1PG1 program. The IMS1PG1 load module consists of three modules:

1. IMS1PI1 is the Main program. It performs all the IMS calls and it also calls the IMS1PS1 module.
2. IMS1PS1 is called by the main module to perform the APPC calls required to initiate and conduct a conversation with the IMSI Partner.
3. IMS1DB2 is called by both the other modules. It performs all the DB2 calls required to insert the message trace into the IMS1_TABLE DB2 table.

Changing IMS1PS1 (APPC module)

Follow these steps to change the APPC calls within the module IMS1PS1:

a. Modify the IMS1PS1 source.

   Modify the member IMS1PS1.

b. Compile IMS1PS1.

   Run a compile of IMS1PS1. (Refer to“Example IMS PL/I Compile JCL” on page 142.)

c. Re-link and DB2 Bind of IMS1PG1

   Run a linkedit of IMS1PG1 to pick up the changed submodule IMS1PS1 and re-bind the DB2 plan IMS1PG1. (Refer to“Example IMS Link Edit JCL” on page 143.)

   (Note: A re-bind is not actually necessary here, but we do it to keep things simple.)

d. Refresh IMS1 MPP Regions.

   Stop/Start the IMS1 MPP regions to pick up the new IMS1PG1 load module. See “Refresh IMS regions” on page 137.

Changing IMS1PI1 (IMS Main Module)

To change the IMS calls within the module IMS1PI1:

a. Modify the IMS1PI1 source.

   Modify the member IMS1PI1.

b. Compile IMS1PI1.

   Run a compile of IMS1PI1.

c. Run a linkedit of IMS1PG1 to pick up the changed submodule IMS1PI1 and re-bind the DB2 plan IMS1PG1.

d. Refresh IMSH MPP Regions

   Stop/Start the IMSH MPP regions to pick up the new IMS1PG1 load module.

Module IMS1PG2 Structure (processes tran IMS1TR2)

The IMS1TR2 transaction is processed by the IMS1PG2 program. The IMS1PG2 load module consists of three modules:

1. IMS1PI2 is the Main program. It performs all the IMS calls and it also calls the IMS1PS2 module.
2. IMS1PS2 is called by the main module to perform the APPC calls required to initiate and conduct a conversation with the IMSI Partner.
3. IMS1DB2 is called by both the other modules. It performs all the DB2 calls required to insert the message trace into the IMS1_TABLE DB2 table.

Changing IMS1PS2 (APPC module)

To change the APPC calls within the module IMS1PS2:

a. Modify the IMS1PS2 source.

   Modify the member IMS1PS2.

b. Compile IMS1PS2.

   Run a compile of IMS1PS2.

c. Re-link & DB2 Bind of IMS1PG2.

   Run a linkedit of IMS1PG2 to pick up the changed submodule IMS1PS2 and re-bind the DB2 plan IMS1PG2.

d. Refresh IMSH MPP Regions.

   Stop/Start the IMSH MPP regions to pick up the new IMS1PG2 load module.

Changing IMS1PI2 (IMS Main Module)

To change the IMS calls within the module IMS1PI2:

a. Modify the IMS1PI2 source.

   Modify the member IMS1PI2.

b. Compile IMS1PI2,

   Run a compile of IMS1PL2.

c. Re-link & DB2 Bind of IMS1PI2.

   Run a linkedit of IMS1Pl2 to pick up the changed submodule IMS1Pl2 and re-bind the DB2 plan IMS1PG2.

d. Refresh IMSH MPP Regions

   Stop/Start the IMSH MPP regions to pick up the new IMS1PG2 load module.

Module IMS1PG3 Structure (processes tran IMS1TR3)

The IMS1TR3 transaction is processed by the IMS1PG3 program. The IMS1PG3 load module consists of three modules:

1. IMS1PI3 is the Main program. It performs all the IMS calls and it also calls the IMS1PS3 module.
2. IMS1PS3 is called by the main module to perform the APPC calls required to initiate and conduct a conversation with the CICS Partner.
3. IMS1DB2 is called by both the other modules. It performs all the DB2 calls required to insert the message trace into the IMS1_TABLE DB2 table.

Changing IMS1PS3 (APPC module)

To change the APPC calls within the module IMS1PS3:

a. Modify the IMS1PS3 source.

   Modify the member IMS1PS3.

b. Compile IMS1PS3.

   Run a compile of IMS1PS3.

c. Re-link and DB2 Bind of IMS1PG3.

   Run a linkedit of IMS1PG3 to pick up the changed submodule IMS1PS3 and re-bind the DB2 plan IMS1PG3.

d. Refresh IMSH MPP Regions.

   Stop/Start the IMSH MPP regions to pick up the new IMS1PG3 load module.

Changing IMS1PI3 (IMS Main Module)

To change the IMS calls within the module IMS1PI3:

a. Modify the IMS1PI3 source.

   Modify the member IMS1PI3.

b. Compile IMS1PI3.

   Run a compile IMS1PI3.

c. Re-link and DB2 Bind of IMS1PG3.

   Run a linkedit of IMS1PG3 to pick up the changed submodule IMS1PI3 and re-bind the DB2 plan IMS1PG3.

d. Refresh IMSH MPP Regions.

   Stop/Start the IMSH MPP regions to pick up the new IMS1PG3 load module.

Module IMS2IMP Structure (processes transaction IMS2IMP)

The IMS2IMP transaction is processed by the IMS2IMP program. The IMS2IMP load module consists of two modules:

1. IMS2IMI is the Main program. It performs all the IMS calls. Since this is a Standard IMS Program there are no APPC calls and hence no APPC sub-module.
2. IMS2DB2 is called to perform the DB2 calls required to insert the message trace into the IMS2_TABLE DB2 table.

Changing IMS2IMP (Main Module)

To change the IMS calls within the module IMS2IMP:

a. Modify the IMS2IMI source.

Modify the member IMS2IMI.

b. Compile IMS2IMI.

Run a compile of IMS2IMI.

c. Re-link and DB2 Bind IMS2IMP.

Run a linkedit of IMS2IMP to pick up the changed submodule IMS2IMI and re-bind the DB2 plan IMS2IMP.

d. Refresh IMSI MPP Regions

Stop/Start the IMSI MPP regions to pick up the new IMS2IMP load module.

Module IMS2EXP Structure (processes transaction IMS2EXP)

The IMS2EXP transaction is processed by the IMS2EXP program. The IMS2EXP load module consists of two modules:

1. IMS2EXP is the Main program. It performs all the IMS calls (APSB/DPSB), and APPC calls required to accept and conduct a conversation with the IMSH Partner. It also issues RRS calls (SRRCMIT /SRRBACK) needed to commit or backout updates.
2. IMS2DB2 is called to perform the DB2 calls required to insert the message trace into the IMS2_TABLE DB2 table.

Changing IMS2EXP (Main Module)

To change the IMS, APPC or RRS calls within the module IMS2EXP:

a. Modify the IMS2EXP source.

Modify the member IMS2EXP.

b. Compile IMS2EXP.

Run a compile and link of IMS2EXP.

c. Re-link and DB2 Bind of IMS2EXP

Run a linkedit of IMS2EXP to pick up the changed submodule IMS2EXP and re-bind the DB2 plan IMS2EXP.

d. Refresh IMSI MPP regions.

Stop/Start the IMSI MPP regions to pick up the new IMS2EXP load module.

Module IMS1DB2 Structure (processes IMSH DB2 inserts)

This module performs the DB2 calls required to insert the message trace into the IMS1_TABLE DB2 table in IMSH. It is used by the IMS1PG1, IMS1PG2, and IMS1PG3, so any change to IMS1DB2 requires all three of these modules to be re-linked and their plans to be re-bound.

Changing IMS1DB2 (IMSH DB2 Insert Module)

To change the DB2 Inserts within the module IMS1DB2:

a. Modify the IMS1DB2 source.

Modify the member IMS1DB2.

b. Compile IMS1DB2.

Run a compile and link of IMS1DB2.

c. Re-link and DB2 Bind of IMS1PG1, IMS1PG2, IMS1PG3.

i. Run a linkedit of IMS1PG1 to pick up the changed submodule IMS1DB2 and re-bind the DB2 plan IMS1PG1.

   ii. Run a linkedit of IMS1PG2 to pick up the changed submodule IMS1DB2 and re-bind the DB2 plan IMS1PG2.

   iii. Run a linkedit of IMS1PG3 to pick up the changed submodule IMS1DB2 and re-bind the DB2 plan IMS1PG3.

   See "Example IMS Link Edit JCL" on page 143.

d. Refresh IMSI MPP regions.

   Stop/Start the IMSI MPP regions to pick up the new IMS1PG1, IMS1PG2 and IMS1PG3 load modules. See "Refresh IMS regions" on page 137.

Module IMS2DB2 Structure (processes IMSI DB2 inserts)

This module performs the DB2 calls required to insert the message trace into the IMS2_TABLE DB2 table. It is used by IMS2IMI and IMS2EXP, so any change to IMS2DB2 requires both these modules to be re-linked and their plans to be re-bound.

Changing IMS2DB2 (IMSI DB2 Insert Module)

To change the DB2 Inserts within the module IMS2DB2:

a. Modify the IMS2DB2 source.

   Modify the member IMS2DB2.

b. Compile IMS2DB2.

   Run a compile of IMS2DB2.

c. Re-link and DB2 Bind of IMS2IMI, IMS2EXP.

   i. Run a linkedit of IMS2IMP to pick up the changed submodule IMS2DB2 and re-bind the DB2 plan IMS2IMP.

   ii. Run a linkedit of IMS2EXP to pick up the changed submodule IMS2DB2 and re-bind the DB2 plan IMS2EXP.

   See "Example IMS Link Edit JCL" on page 143.

d. Refresh IMSI MPP regions.

   Stop/Start the IMSI MPP regions to pick up the new IMS2IMI and IMS2EXP load modules. See "Refresh IMS regions" on page 137.

Example IMS PL/I Compile JCL

Example A-18: IMS Program Compile JCL

```
//JOBCARD.....
//*
// JCLLIB ORDER=IBMZ.SIBMZPRC
//*
//*-------------------------------------------------------------------+
//*                                                                   |
//* COMPILE IMS1PI1                                                   |
//*                                                                   |
//*-------------------------------------------------------------------+
//*                                                                   |
//* TAILOR THE JCL.                                                   |
//*                                                                   |
//*  1) Amend the JCLLIB ORDER statement.                             |
//*     - point to the SIBMZPRC PL/I Enterprise Compiler dataset.     |
//*                                                                   |
//*  2) Amend the LNGPRFX parm of the IBMZCB procedure.               |
//*     - set to the HLQ of the PL/I Enterprise Compiler libraries.   |
//*                                                                   |
```

```
//*  3) Amend the overrides;                                         |
//*     - point to your SYSx.SIEAHDR.H dataset.                      |
//*     - point to your version of USER.PROGRAM.APPCPLI. (Source)    |
//*     - point to your CSSLIB dataset.                              |
//*   - point to your version of USER.PROGRAM.APPCLOAD (LOADLIB)     |
//*----------------------------------------------------------------+
//*
//S1  EXEC IBMZCB,
//         LNGPRFX='IBMZ',
//         PARM.BIND='NCAL,LIST,XREF,LET,AMODE=31,RMODE=ANY'
//PLI.SYSLIB   DD DISP=SHR,DSN=SYS1.SIEAHDR.H
//PLI.SYSIN    DD DISP=SHR,DSN=USER.PROGRAM.APPCPLI(IMS1PI1)
//BIND.SYSLIB  DD
//             DD DISP=SHR,DSN=SYS1.CSSLIB
//BIND.SYSLMOD DD DISP=SHR,DSN=USER.PROGRAM.LOAD
//BIND.SYSIN   DD *
 NAME IMS1PI1(R)
/*
```

Note: To taylor the compile job refer to the comments in the JCL prolog.

Example IMS Link Edit JCL

Example A-19: IMS Link Edit JCL

```
//JOBCARD....
//*----------------------------------------------------------------+
//*                                                                |
//* IMS1PG1 LINK EDIT AND BIND OF ITS DB2 PLAN.                    |
//*                                                                |
//*----------------------------------------------------------------+
//*                                                                |
//* TAILOR THE JCL.                                                |
//*                                                                |
//* For the LKED step;                                             |
//*  1) Amend the STEPLIB datasets;                                |
//*     - point to your CSSLIB dataset.                            |
//*     - point to your SCEELKED dataset.                          |
//*     - point to your DB2 SDSNLOAD dataset.                      |
//*                                                                |
//*  2) Amend the RESLIB to point to your IMS RESLIB dataset.      |
//*                                                                |
//*  3) Amend the SYSLMOD dataset to point to your LOAD library.   |
//*                                                                |
//* For the BIND step;                                             |
//*  1) Amend the STEPLIB to point to your DB2 SDSNLOAD dataset.   |
//*                                                                |
//*  2) Amend the DBRMLIB to point to your DB2 DBRMLIB dataset.    |
//*                                                                |
//*  3) Amend the SYSTSIN input parameters;                        |
//*     - SYSTEM(@7C), replace @7C with your DB2 system's SID.     |
//*     - LIB('DB2710C.D7C.RUNLIB.LOAD'), update the library name. |
//*                                                                |
//*----------------------------------------------------------------+
//*                                                               */
//* IMS1PG1 LOAD MODULE CONSISTS OF;                              */
//*      IMS1PI1   - MAIN MODULE WHICH PERFROMS THE IMS CALLS.    */
//*      IMS1PS1   - MODULE WHICH PERFROMS THE APPC CALLS.        */
//*      IMS1DB2   - MODULE INSERTS INTO DB2 TABLE IMS1_TABLE.    */
//*      DFSLI000  - IMS LANGUAGE INTERFACE.                      */
```

```
//*                                                                 */
//* ENTRY POINT;                                                    */
//*      CEESTART                                                   */
//*                                                                 */
//* NOTES:                                                          */
//*    1) DFSLI000 FROM V810 SDFSRESL.                              */
//*    2) CEESTART FROM LE LIBRARY (CEE.SCEELKED)                   */
//*    3) APPC, CPI-C VERBS (SYS1.CSSLIB)                           */
//*                                                                 */
//*******************************************************************/
//*
//LKED     EXEC PGM=IEWL,
//         PARM='LIST,XREF,LET,AMODE=31,RMODE=ANY',
//         COND=(4,LT)
//SYSLIB   DD DISP=SHR,DSN=SYS1.CSSLIB                <= APPC
//         DD DISP=SHR,DSN=CEE.SCEELKED               <= LE
//         DD DISP=SHR,DSN=DB8K8.SDSNLOAD             <= DB2
//SYSLIN   DD DDNAME=SYSIN
//RESLIB   DD DSN=IMS810H.SDFSRESL,DISP=SHR           <= IMS RESLIB
//SYSLMOD  DD DSN=USER.PROGRAM.LOAD,DISP=SHR          <= MPP LOADLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(1024,(50,50))
//SYSIN    DD *
  INCLUDE SYSLMOD(IMS1PI1)
  INCLUDE SYSLMOD(IMS1DB2)
  INCLUDE SYSLMOD(IMS1PS1)
  INCLUDE RESLIB(DFSLI000)
  ENTRY   CEESTART
  NAME    IMS1PG1(R)
/*
//*
//BIND    EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB  DD  DISP=SHR,DSN=DB8K8.SDSNLOAD
//DBRMLIB  DD  DISP=SHR,DSN=USER.PROGRAM.DBRMLIB.DATA
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//REPORT   DD  SYSOUT=*
//SYSIN    DD  *
 GRANT       EXECUTE ON PLAN IMS1PG1  TO PUBLIC;
//SYSTSIN  DD  *
 DSN SYSTEM(DB8K)
 FREE PLAN(IMS1PG1)
 BIND PLAN(IMS1PG1) MEMBER(IMS1DB2) ACT(REP) ISO(CS) RETAIN +
      VALIDATE (BIND) OWNER (APPCOWN)
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA81) -
       LIB('DB8KU.RUNLIB.LOAD')
 END
/*
```

Note: To taylor the compile job refer to the comments in the JCL prolog.

DB2 Definitions

The following tasks are required when implementing the DB2 component of the scenarios employed throughout this redbook. Note that DB2 administrator access is required. Install and test the CICS section of the DB2 IVP.

Install the CICS DB2 IVP into the DB2 subsystem

The CICS DB2 IVP Job and Application source can be found in the DB2.SDSNSAMP library.

To prepare the sample applications to be used in a CICS-DB2 environment, run the job DSNTEJ5C (Cobol Version). Job DSNTEJ5C installs the sample application transactions in COBOL and prepares the organization application. Customize this job to match your data center standards.

This job will:

- Compile and link-edit the CICS online applications.
- Bind the CICS online applications.
- Create the BMS maps for the online applications.

Update your CICS1 DFHSITxx to include DB2 startup definitions

```
DB2CONN=YES,
INITPARM=(DFHD2INI=',DB@id',CMRFSET2=',"SUBSYS=BBCI"'),
```

Update the CICS1 Startup JCL (either STC or JOB)

The following libraries must be added to your CICS STEPLIB in the order shown:

```
CEE.SCEERUN2
CEE.SCEERUN
DB2.V7R1M0.SDSNLOAD
```

The following libraries must be added to the CICS DFHRPL in the order shown:

```
CEE.SCEECICS
CEE.SCEERUN2
CEE.SCEERUN
DB2.V7R1M0.SDSNLOAD
DB2710C.D7C.RUNLIB.LOAD
```

(B2710C.D7C.RUNLIB.LOAD is used as application library for CICS / DB2 sample IVP programs.)

Install the required CICS components and connections

Define a DB2 CONNection in your CICS group for a DB2 Subsystem that you want to connect to. This example shows the definitions being created by the CICS Resource Definition Online method using the CICS transaction CEDA. You can also use the CICS batch utility DFHCSDUP to create the same definitions.

> **Note:** When you have completed the definitions you will have to ADD your groupname to the CICS1 grplist. This will install and activate your new definitions the next time you start CICS. To avoid a CICS restart you can also issue a CEDA INSTALL GROUP(groupname) in order to activate your definitions dynamically.

Example A-20: CEDA Define DB2Conn

```
CEDA  View DB2Conn( DB2id      )
 DB2Conn        : DB2id
 Group          : groupname
 DEscription    : CONNECT DB2id TO CICSPA8K
CONNECTION ATTRIBUTES
 CONnecterror   : Sqlcode            Sqlcode | Abend
 DB2Groupid     :
 DB2Id          : DB2id
 MSGQUEUE1             : CDB2
 MSGQUEUE2             :
 MSGQUEUE3             :
 Nontermrel     : Yes                Yes | No
 PUrgecycle     : 00 , 30            0-59
 Resyncmember          : Yes                Yes | No
 SIgnid         :
 STANdbymode         : Reconnect        Reconnect | Connect | Noconnect
 STATsqueue          : CDB2
 TCblimit       : 0012               4-2000
 THREADError           : N906D              N906D | N906 | Abend
POOL THREAD ATTRIBUTES
 ACcountrec     : TXid               None | TXid | TAsk | Uow
 AUTHId         :
 AUTHType       : Userid             Userid | Opid | Group | Sign | TErm
                                     | TX
 DRollback      : Yes                Yes | No
 PLAN           :
 PLANExitname          : DSNCUEXT
 PRiority       : High               High | Equal | Low
 THREADLimit          : 0003               3-2000
 THREADWait          : Yes                Yes | No
COMMAND THREAD ATTRIBUTES
 COMAUTHId           :
 COMAUTHType          : Userid             Userid | Opid | Group | Sign | TErm
                                     | TX
 COMThreadlim          : 0001              0-2000
```

Implement the LE environment and IVP application

Add Group (CEE) to the CICS1 region group list *(required for LE services).* Ensure the DB2 IMS IVP COBOL compiles using the Enterprise COBOL Compiler (we are using V3.3.0).

Defining the DB2 tables

We created these DB2 tables in the DB2 subsystem. They are used by the five IMS programs and the CICS program to insert a log of their activities. There are three tables, one each for the IMS1, IMS2, and CICS systems, and they are named:

- IMS1_TABLE
- IMS2_TABLE
- CICS_TABLE

These tables provide the means of verifying whether DB2 Updates performed by the participants within a protected conversation are backed out during a failure or program induced backout.

Acquire DB2 SYSADM authority for the DB2 system

SYSADM access is required to create the additional resources in the DB2 Subsystem.

Create the DB2 STORAGE Group APPCSGP

a. Review the STOGROUP definitions

Example A-21: DB2 STOGROUP Definitions

```
--
--  CREATE STORAGE GROUP FOR IMS/CICS APPC..
--
-- DROP   STOGROUP APPCSGP  ;
-- COMMIT;

   CREATE STOGROUP APPCSGP
         VOLUMES ('*')
         VCAT    DB2710C ;
   GRANT USE OF STOGROUP APPCSGP  TO PUBLIC ;
   COMMIT ;
```

b. Modify the value DB2710C specified for the VCAT subparameter to the value to be used for your DB2 VSAM datasets. The VCAT subparameter determines the VSAM High Level Qualifier under which the DB2 Tablespace datasets will be created.

c. Using SPUFI under ISPF process the STOGROUP definitions. Refer to "SPUFI and SQL" on page 151 for instructions on how to use SPUFI.

Create the DB2 Database APPCDB

Use SPUFI under ISPF to process the DATABASE definitions.

Example A-22: Creating the DB2 Databases

```
--
--  CREATE DATABASE FOR IMS/CICS APPC..
--
  DROP   DATABASE APPCDB ;
  COMMIT ;
--
  CREATE DATABASE APPCDB
    BUFFERPOOL BP0
    STOGROUP APPCSGP ;
  COMMIT ;
--
  GRANT DBADM ON DATABASE APPCDB    TO PUBLIC ;
```

Create the DB2 IMS1TS tablespace and IMS1_TABLE table

Use SPUFI under ISPF to process the IMS1TAB definitions.

Example A-23: DB2 IMS1TS defining tables.

```
--
--  CREATE TABLESPACE, TABLE, INDEX FOR IMSH APPC ...
--
--DROP   TABLESPACE APPCDB.IMS1TS;
--COMMIT ;
--
  CREATE TABLESPACE IMS1TS
   IN APPCDB
   USING STOGROUP APPCSGP
             PRIQTY 48
             SECQTY 48
             ERASE NO
   LOCKSIZE ROW  LOCKMAX SYSTEM
```

```
    BUFFERPOOL BP0
    CLOSE NO ;
--
  CREATE TABLE APPCOWN.IMS1_TABLE
                 (MSG_TIMESTAMP TIMESTAMP   NOT NULL WITH DEFAULT,
MSG_NAME     CHAR(8)     NOT NULL,
                  MSG_TEXT     CHAR(136)   NOT NULL )
          IN APPCDB.IMS1TS ;
--
  CREATE UNIQUE INDEX APPCOWN.IMS1_INDEX
                 ON APPCOWN.IMS1_TABLE
                     (MSG_TIMESTAMP  ASC)
                 USING STOGROUP APPCSGP
                           PRIQTY 48
                           SECQTY 48
                           ERASE NO
                 BUFFERPOOL BP0   ;
--

  GRANT ALL PRIVILEGES ON APPCOWN.IMS1_TABLE TO PUBLIC ;
```

Create the DB2 IMS2TS tablespace and IMS2_TABLE table

Use SPUFI under ISPF to process the IMS2TAB definitions.

Example A-24: DB2 IMS2TS defining tables.

```
--
--  CREATE TABLESPACE, TABLE, INDEX FOR IMSI APPC ...
--
--DROP   TABLESPACE APPCDB.IMS2TS;
--COMMIT ;
--
  CREATE TABLESPACE IMS2TS
   IN APPCDB
   USING STOGROUP APPCSGP
             PRIQTY 48
             SECQTY 48
             ERASE NO
   LOCKSIZE ROW  LOCKMAX SYSTEM
   BUFFERPOOL BP0
   CLOSE NO ;
--
  CREATE TABLE APPCOWN.IMS2_TABLE
                 (MSG_TIMESTAMP TIMESTAMP   NOT NULL WITH DEFAULT,
MSG_NAME     CHAR(8)     NOT NULL,
                  MSG_TEXT     CHAR(136)   NOT NULL )
          IN APPCDB.IMS2TS ;
--
  CREATE UNIQUE INDEX APPCOWN.IMS2_INDEX
                 ON APPCOWN.IMS2_TABLE
                     (MSG_TIMESTAMP  ASC)
                 USING STOGROUP APPCSGP
                           PRIQTY 48
                           SECQTY 48
                           ERASE NO
                 BUFFERPOOL BP0   ;
--

  GRANT ALL PRIVILEGES ON APPCOWN.IMS2_TABLE TO PUBLIC ;
```

Create the DB2 CICSTS tablespace and CICS_TABLE table

Use SPUFI under ISPF to process the CICSTAB definitions.

Example A-25: DB2 CICSTS defining tables.

```
--
--  CREATE TABLESPACE, TABLE, INDEX FOR CICS APPC ...
--
--DROP   TABLESPACE APPCDB.CICSTS;
--COMMIT ;
--
  CREATE TABLESPACE CICSTS
   IN APPCDB
   USING STOGROUP APPCSGP
             PRIQTY 48
             SECQTY 48
             ERASE NO
   LOCKSIZE ROW  LOCKMAX SYSTEM
   BUFFERPOOL BP0
   CLOSE NO ;
--
  CREATE TABLE APPCOWN.CICS_TABLE
                 (MSG_TIMESTAMP TIMESTAMP   NOT NULL WITH DEFAULT,
MSG_NAME     CHAR(8)     NOT NULL,
                  MSG_TEXT     CHAR(136)   NOT NULL )
          IN APPCDB.CICSTS ;
--
  CREATE UNIQUE INDEX APPCOWN.CICS_INDEX
                 ON APPCOWN.CICS_TABLE
                     (MSG_TIMESTAMP  ASC)
                 USING STOGROUP APPCSGP
                           PRIQTY 48
                           SECQTY 48
                           ERASE NO
                 BUFFERPOOL BP0   ;
--

  GRANT ALL PRIVILEGES ON APPCOWN.CICS_TABLE TO PUBLIC ;
```

Example IMS1_TABLE Table contents

The output written to the JOBLOG is also written to the IMS1_TABLE. This can be viewed via SPUFI. Note that if an abend or backout occurs, the information will not be available in the IMS1_TABLE because it will have been backed-out.

Example A-26: Sample IMS1_TABLE

```
---------+---------+---------+---------+---------+---------+---------+---------+
 -- select * from APPCOWN.IMS1_TABLE                                    00030008
 -- select * from APPCOWN.IMS2_TABLE                                    00031008
 -- select * from APPCOWN.CICS_TABLE                                    00032009
    select * from APPCOWN.IMS1_TABLE                                    00070015
---------+---------+---------+---------+---------+---------+---------+---------+
MSG_TIMESTAMP              MSG_NAME  MSG_TEXT
---------+---------+---------+---------+---------+---------+---------+---------+
2004-09-28-12.26.14.730098  IMS10100  IMS10100 MESSAGE TEXT
2004-09-28-12.44.06.647092  IMN1MP1   IMS1PI2: Call to Insert Msg in IMS1_TABLE
2004-09-29-14.55.17.381845  IMS1MP1   IMS1PI1: PL/I IMS stub routine Begins...
2004-09-29-14.55.17.462183  IMS1MP1   IMS1PI1: IMS_GU Entry.
2004-09-29-14.55.17.530170  IMS1MP1   IMS1PI1: IMS_GU Status IOPCB.STC_CODE=
2004-09-29-14.55.17.566920  IMS1MP1   IMS1PI1: IMS_GU Input= ZZZZZZZZZZZZZZZZZZZ
```

```
2004-09-29-14.55.17.605054  IMS1MP1   IMS1PI1: IMS_GU Successful
2004-09-29-14.55.17.644629  IMS1MP1   IMS1PS1: Outbound routine begins...
2004-09-29-14.55.17.680078  IMS1MP1   IMS1PS1: Allocate Entry.
2004-09-29-14.55.17.712252  IMS1MP1   IMS1PS1: Allocate User_ID=NO_USERID , Pass
2004-09-29-14.55.18.408505  IMS1MP1   IMS1PS1: Allocate Return_Code=0
2004-09-29-14.55.18.483362  IMS1MP1   IMS1PS1: Allocate Conversation_ID=0EEBB3F8
2004-09-29-14.55.18.523391  IMS1MP1   IMS1PS1: Allocate RC is
2004-09-29-14.55.18.555063  IMS1MP1   IMS1PS1: OK
```

Example IMS2_TABLE table contents

The output written to the JOBLOG is also written to the IMS1_TABLE. This can be viewed via SPUFI. Note that if an abend or backout occurs, the information will not be available in the IMS2_TABLE because it will have been backed-out.

Example A-27: Sample IMS2_TABLE

```
---------+---------+---------+---------+---------+---------+---------+---------+
 -- select * from APPCOWN.IMS1_TABLE                                    00030008
 -- select * from APPCOWN.IMS2_TABLE                                    00031008
 -- select * from APPCOWN.CICS_TABLE                                    00032009
    select * from APPCOWN.IMS2_TABLE                                    00060014
---------+---------+---------+---------+---------+---------+---------+---------+
MSG_TIMESTAMP              MSG_NAME  MSG_TEXT
---------+---------+---------+---------+---------+---------+---------+---------+
2004-09-29-13.40.21.672225  IMS2MP1   IMS2IMP: Call DB2 to Insert Msg in IMS2_TABLE
2004-09-29-13.47.06.391961  IMS2MP1   IMS2IMP: Call DB2 to Insert Msg in IMS2_TABLE
2004-09-29-14.09.49.619018  IMS2MP1   IMS2IMI: PL/I IMS stub routine Begins...
2004-09-29-14.09.49.839332  IMS2MP1   IMS2IMI: IMS_GU Entry.
2004-09-29-14.09.49.885937  IMS2MP1   IMS2IMI: IMS_GU Status IOPCB.STC_CODE=
2004-09-29-14.09.49.937093  IMS2MP1   IMS2IMI: IMS_GU IMS_GU Input= XXXXXXXXXXXXXX
2004-09-29-14.09.50.043876  IMS2MP1   IMS2IMI: IMS_GU Successful
2004-09-29-14.09.50.081853  IMS2MP1   IMS2IMI: IMS_ISRT Entry.
2004-09-29-14.09.50.124947  IMS2MP1   IMS2IMI: IMS_ISRT Output=IMS2IMP Completed
2004-09-29-14.09.50.165485  IMS2MP1   IMS2IMI: IMS_ISRT Status IOPCB.STC_CODE=
2004-09-29-14.09.50.202251  IMS2MP1   IMS2IMI: IMS_ISRT Successful
2004-09-29-14.09.50.241474  IMS2MP1   IMS2IMI: IMS_GU Entry.
```

Example CICS_TABLE table contents

The output written to the CICS MSGOUT DDname is also written to the CICS_TABLE. This can be viewed via SPUFI. Note that if an abend or backout occurs, the information will not be available in the CICS_TABLE because it will have been backed-out.

Example A-28: Sample CICS_TABLE

```
---------+---------+---------+---------+---------+---------+---------+---------+
 -- select * from sysibm.systables                                      00010005
 -- select * from dsn8710.dept                                          00011004
 -- select * from dsn8710.emp                                           00020004
 -- select * from APPCOWN.IMS1_TABLE                                    00030008
 -- select * from APPCOWN.IMS2_TABLE                                    00031008
 -- select * from APPCOWN.CICS_TABLE                                    00032009
 -- INSERT INTO APPCOWN.IMS1_TABLE (MSG_NAME,MSG_TEXT)                  00040007
 --        VALUES ('IMS10100','IMS10100 MESSAGE TEXT') ;                00050007
    select * from APPCOWN.CICS_TABLE                                    00080016
---------+---------+---------+---------+---------+---------+---------+---------+
MSG_TIMESTAMP              MSG_NAME  MSG_TEXT
---------+---------+---------+---------+---------+---------+---------+---------+
2004-09-27-15.46.35.858884  SPG1:     CICSPG1: 27/09/04 15:46:29
```

```
2004-09-28-10.16.52.193888  C: ##     IMPAPPC: 28/09/04 10:16:52 SPG1: Called  s
2004-09-28-10.20.20.684459  TEST      IMPAPPC: 28/09/04 10:20:21 SPG1: Called  s
2004-09-28-14.12.24.272093  TEXT      IMPAPPC: 28/09/04 14:12:24 SPG1: Called  s
2004-09-28-14.27.57.460981  AVAST     IMPAPPC: 28/09/04 14:27:57 SPG1: Called  s
2004-09-28-14.30.42.050725  NEW       IMPAPPC: 28/09/04 14:30:42 SPG1: Called  s
2004-09-28-14.34.13.625246  SHOW      IMPAPPC: 28/09/04 14:34:14 SPG1: Called  s
2004-09-28-14.35.16.709240  CHECK     IMPAPPC: 28/09/04 14:35:17 SPG1: Called  s
```

SQL examples for defining and modifying the DB2 environment

The following SQL examples were used to report, modify and implement the DB2 environment used in our scenarios. Change ???? to the appropriate prefix; IMS1, IMS2, or CICS before executing these sample SQL statements.

To insert a record in to a table:

```
INSERT INTO APPCOWN.????_TABLE (MSG_NAME,MSG_TEXT)
     VALUES ('????0001','????0001 MESSAGE TEXT') ;
```

To display the contents of a table:

```
SELECT * FROM APPCOWN.????_TABLE ;
```

To delete records for the table:

```
DELETE FROM APPCOWN.????_TABLE ;
```

DB2 attachment

To implement the CICS DB2 attachment facility, you must use the CICS-supplied group called DFHDB2.

Add Group(DFHDB2) to the CICS1 region GRPLIST.

Reference:

For information on how to run the CICS to DB2 IVP application refer to:

> *CICS TS Installation Guide,* Chapter titled “CICS Verification - Starting a DB2 organization or project application”

For further information on the DB2 IVP and CICS Attachment facility refer to:

> *DB2 Universal Database for OS/390 and z/OS Installation Guide,* Chapter 12. “Connecting the CICS attachment facility”

SPUFI and SQL

In order to examine the results of a backout or commit to the DB2 tables, you can run the SELECT SQL Statements though the SPUFI ISPF dialogs.

Select SPUFI (This may vary from site to site).

Example A-29: SPUFI Screen

```
SPUFI
 ===>

 Enter the input data set name:        (Can be sequential or partitioned)
  1  DATA SET NAME ... ===> 'RC62.GRAHAMD.APPC.DATA(SQLSEL)'
  2  VOLUME SERIAL ... ===>            (Enter if not cataloged)
  3  DATA SET PASSWORD ===>            (Enter if password protected)
```

```
 Enter the output data set name:       (Must be a sequential data set)
  4  DATA SET NAME ... ===> 'RC62.GRAHAMD.APPC.SQLOUT'

 Specify processing options:
  5  CHANGE DEFAULTS   ===> YES        (Y/N - Display SPUFI defaults panel?)
  6  EDIT INPUT ...... ===> YES        (Y/N - Enter SQL statements?)
  7  EXECUTE ......... ===> YES        (Y/N - Execute SQL statements?)
  8  AUTOCOMMIT ...... ===> YES        (Y/N - Commit after successful run?)
  9  BROWSE OUTPUT ... ===> YES        (Y/N - Browse output data set?)

 For remote SQL processing:
 10  CONNECT LOCATION  ===>


 PRESS:  ENTER to process    END to exit             HELP for more information






 PF 1=HELP      2=SPLIT     3=END       4=RETURN    5=RFIND     6=RCHANGE
 PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
```

Press Enter to Edit the SQL required to display the DB2 data.

Example A-30: SQL - Display the CICS table records.

```
EDIT       RC62.GRAHAMD.APPC.DATA(SQLSEL) - 01.03         Columns 00001 00072
Command ===>                                                 Scroll ===> PAGE
****** ***************************** Top of Data ******************************
000001 SELECT * FROM APPCOWN.CICS_TABLE
000002   WHERE MSG_TEXT LIKE '%' ORDER BY MSG_TEXT;
****** **************************** Bottom of Data ****************************
```

Press PF3 and then press Enter to run the SQL commands. Example A-31shows a sample of the table display.

Example A-31: SPUFI SQL report

```
---------+---------+---------+---------+---------+---------+---------+---------+
SELECT * FROM APPCOWN.CICS_TABLE
  WHERE MSG_TEXT LIKE '%' ORDER BY MSG_TEXT;
---------+---------+---------+---------+---------+---------+---------+---------+
MSG_TIMESTAMP              MSG_NAME  MSG_TEXT
---------+---------+---------+---------+---------+---------+---------+---------+
2004-11-03-11.13.06.711246  PCMIT     IMPAPPC: 03/11/04 11:13:07 SPG1: Called  s
2004-11-03-11.23.12.948011  PCMIT     IMPAPPC: 03/11/04 11:23:13 SPG1: Called  s
2004-11-03-11.23.26.883461  PCMIT     IMPAPPC: 03/11/04 11:23:27 SPG1: Called  s
2004-11-05-09.42.35.216741  PCMIT     IMPAPPC: 05/11/04 09:42:35 SPG1: Called  s
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE612I DATA FOR COLUMN HEADER MSG_TEXT COLUMN NUMBER 3 WAS TRUNCATED
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
```

```
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 2
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 22
```

B

APPC exploiter sample source code

This appendix provides an overview of the sample source used during the redbook exercise.

This appendix provides descriptions for the CICS and IMS source programs. The samples referred to here are provided on an as-is basis and can be downloaded from the redbook Web site. You can use these samples as the building blocks for developing an APPC application.

Refer to Appendix C, “Additional material” on page 161 for further information on obtaining source code.

CICS Programs

Several programs are addressed in this section. A short description is included for each program. Comments within the programs may also provide further instructions and descriptions that are specific to each program.

The CICS programs are written in Cobol and the samples include:

- CICS Inbound program - CICSPG1
- CICS Outbound program - GTCICS02

CICS Inbound program - CICSPG1

The Cobol program CICSPG1 is developed as an inbound program which is used as the *receiver* for a protected conversation with an IMS Outbound Program. CICSPG1 writes a record to the DB2 database each time it is invoked. Figure B-1 depicts the relationship between transactions and programs.

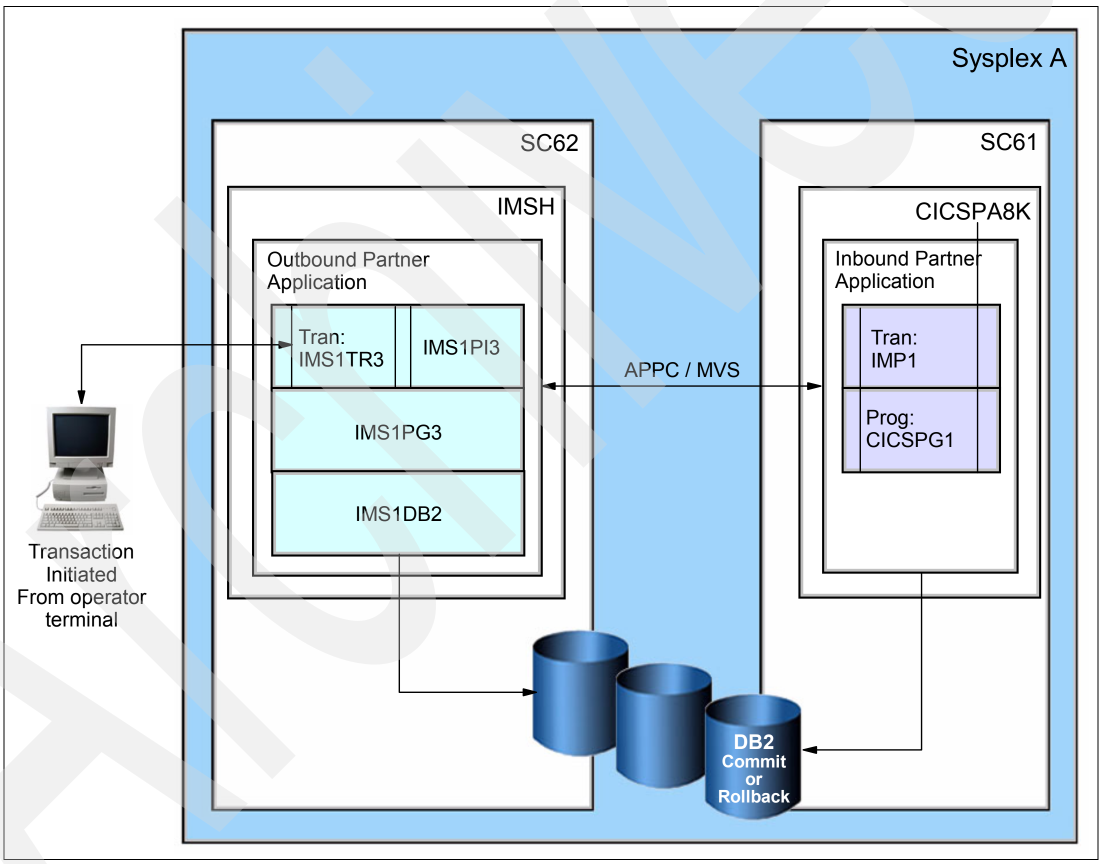


Figure B-1 CICS inbound scenario

CICS Outbound program - GTCICS02

The Cobol program GTCICS02 is developed as an outbound program which requests a protected conversation with an IMS Inbound Program. GTCICS02 writes a record to the DB2 database. Figure B-2 depicts the relationship between transactions and programs.

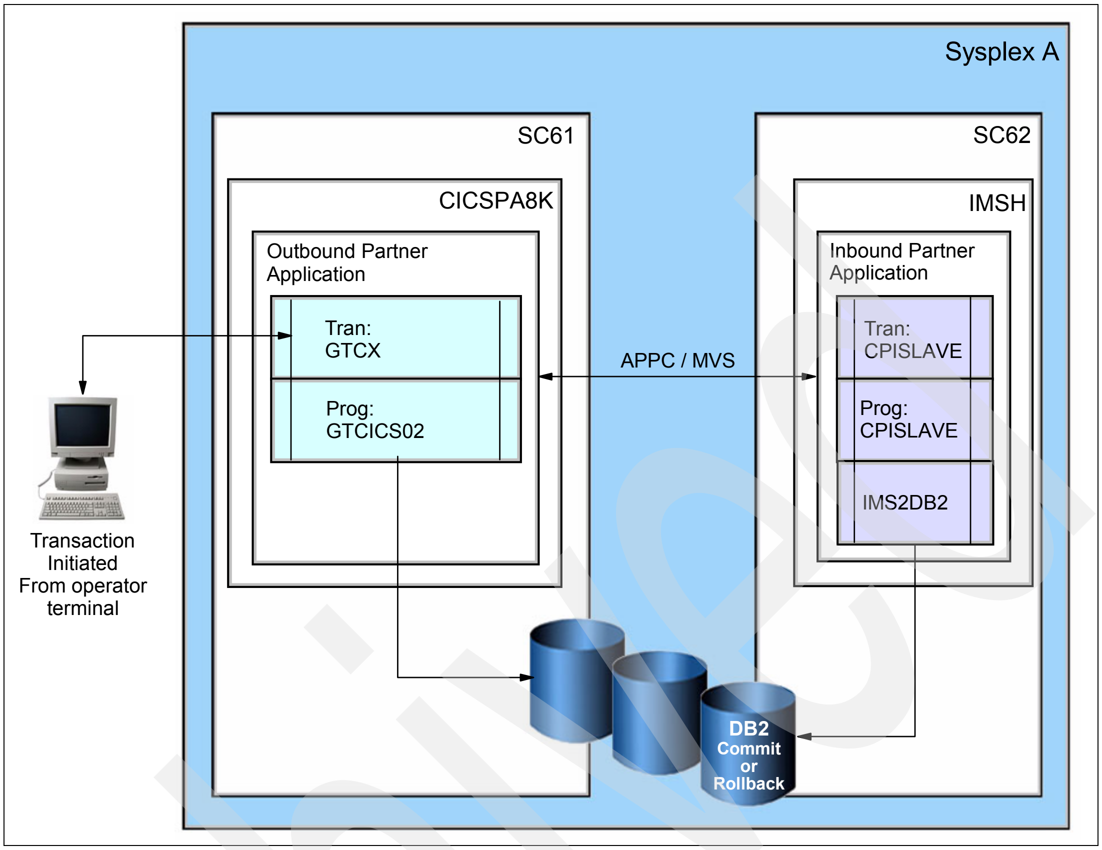


Figure B-2 CICS outbound scenario

IMS programs

The following programs are addressed in this section. A short description is included for each program. Comments within the programs may also provide further instructions and descriptions that are specific to each program.

The samples include both Inbound and Outbound pairs and Implicit and Explicit modes:

- IMS Inbound program - CPISLAVE
- IMS Outbound program - IMS1PS3
- IMS Outbound program - IMS1PI3
- IMS Outbound Implicit program - IMS1PI1
- IMS Outbound Implicit program - IMS1PS1
- IMS Inbound Implicit program - IMS2IMI
- IMS Outbound Implicit program - IMS1PI2
- IMS Outbound Explicit program - IMS1PS2
- IMS Inbound Explicit program - IMS2EXP

IMS Inbound program - CPISLAVE

The PL/I program CPISLAVE is developed as an Inbound program which is used as the *receiver* for a protected conversation with a CICS Outbound Program. CPISLAVE calls the

IMS2DB2 module to write a record to a DB2 database. Refer to Figure B-2 for an overview of where this program fits in the scenario.

IMS Outbound program - IMS1PS3

The PL/I program IMS1PS3 is developed as an Outbound program which is controlled via IMS1PI3 to request a protected conversation with a CICS Inbound Program. IMS1PS3 calls the IMS2DB2 module to write a record to a DB2 database. Refer to Figure B-1 for an overview of where this program fits in the scenario.

IMS Outbound program - IMS1PI3

The PL/I program IMS1PI3 is developed as an Outbound program which is controls IMS1PS3 when requesting a protected conversation with a CICS Inbound Program. Refer to Figure B-1 for an overview of where this program fits in the scenario.

IMS Outbound Implicit program - IMS1PI1

The PL/I program IMS1PI1 is developed as an Outbound program which is controls IMS1PS1 when requesting a protected conversation with a IMS Inbound Program in implicit mode. Refer to Figure B-3 for an overview of where this program fits in the scenario.

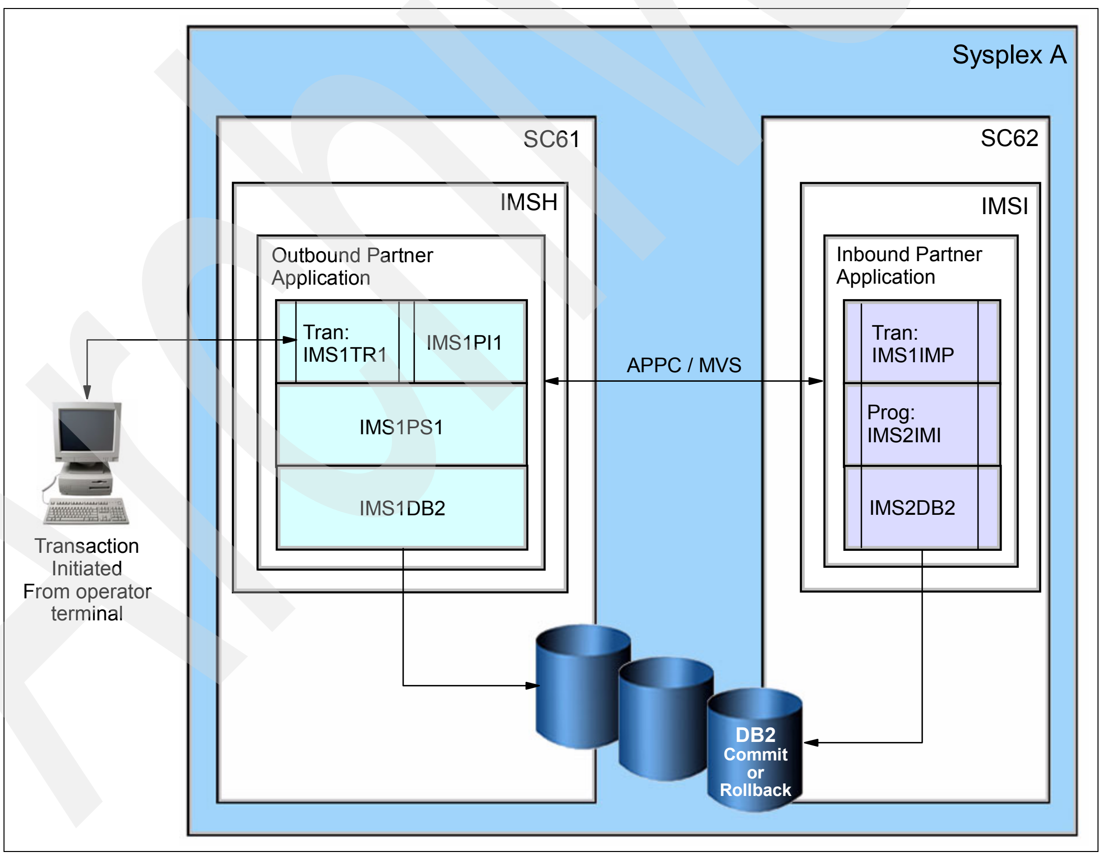


Figure B-3 Implicit IMS to IMS scenarios

IMS Outbound Implicit program - IMS1PS1

The PL/I program IMS1PS1 is developed as an Outbound program which is controlled by IMS1PI1. It issues the APPC calls to request a protected conversation with a IMS Inbound Program in implicit mode. IMS1PS1 calls the IMS2DB2 module to write a record to a DB2 database. Refer to Figure B-3 for an overview of where this program fits in the scenario.

IMS Inbound Implicit program - IMS2IMI

The PL/I program IMS2IMI is developed as an Inbound program which is initiated by IMS1PS1. The IMS Inbound Program is in implicit mode. Refer to Figure B-3 for an overview of where this program fits in the scenario.

IMS Outbound Implicit program - IMS1PI2

The PL/I program IMS1PI2 is developed as an Outbound program which controls IMS1PS2 when requesting a protected conversation with a IMS Inbound Program in implicit mode. Refer to Figure B-4 for an overview of where this program fits in the scenario.

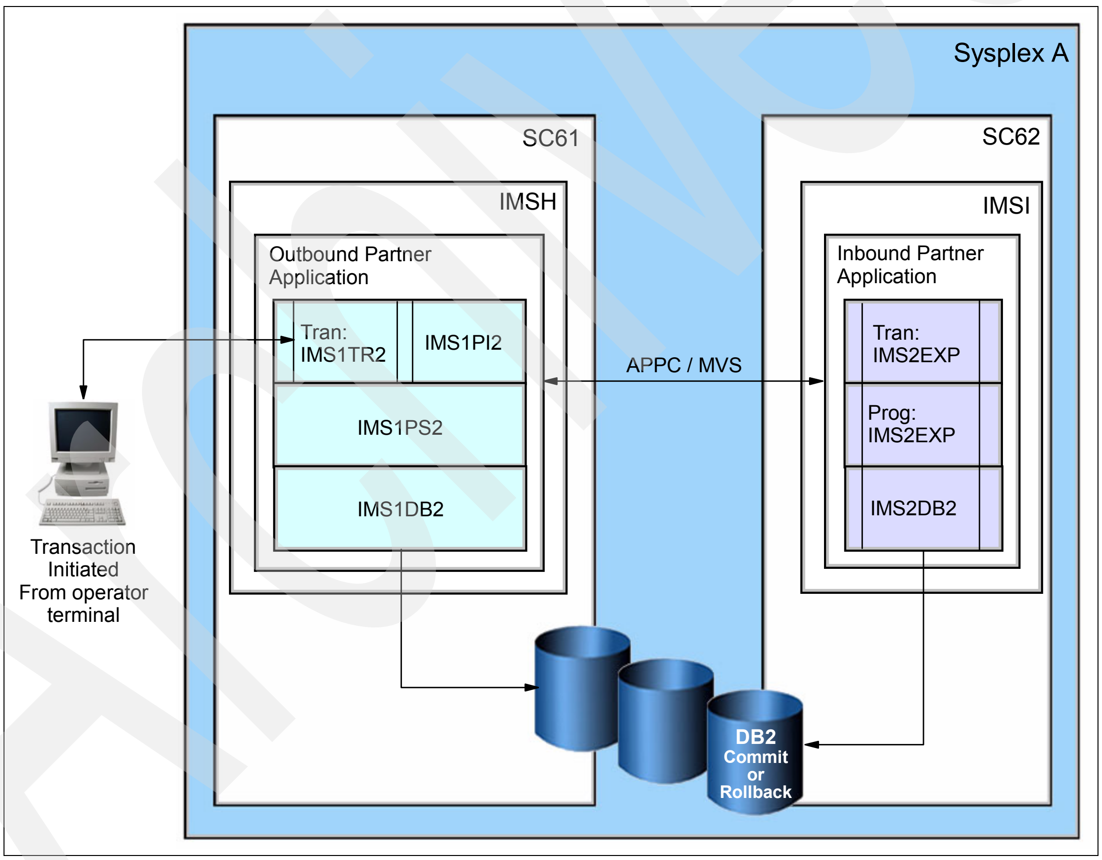


Figure B-4 Explicit IMS to IMS scenarios

IMS Outbound Explicit program - IMS1PS2

The PL/I program IMS1PS2 is developed as an Outbound program which is controlled by IMS1PI2. It issues the APPC calls to request a protected conversation with a IMS Inbound Program in implicit mode. IMS1PS2 calls the IMS2DB2 module to write a record to a DB2 database. Refer to Figure B-4 for an overview of where this program fits in the scenario.

IMS Inbound Explicit program - IMS2EXP

The PL/I program IMS2EXP is developed as an Inbound program which is initiated by IMS1PS2. The IMS Inbound Program is in explicit mode. IMS2EXP calls the IMS2DB2 module to write a record to a DB2 database. Refer to Figure B-4 for an overview of where this program fits in the scenario.

IMS DB2 program - IMS1DB2

The PL/I program IMS1DB2 writes a record to the DB2 database as directed by:

- IMS1PI1
- IMS1PS1
- IMS1PI2
- IMS1PS2
- IMS1PI3
- IMS1PS3

> **Note:** The CICS sample programs CICPG01 and GTCICS02 use the DB2 attachment facility to execute SQL and update DB2 tables directly.

DB2 updates are used in each of the scenarios. For an overview refer to figures:

- Figure B-1, "CICS inbound scenario" on page 156.
- Figure B-2, "CICS outbound scenario" on page 157.

IMS DB2 program - IMS2DB2

The PL/I program IMS2DB2 writes a record to the DB2 database as directed by:

- IMS2EXP
- IMS2IMI
- CPISLAVE

DB2 updates are used in each of the scenarios. For an overview refer to figures:

- Figure B-3, "Implicit IMS to IMS scenarios" on page 158.
- Figure B-4, "Explicit IMS to IMS scenarios" on page 159.

> **Note:** The CICS sample programs CICPG01 and GTCICS02 use the DB2 attachment facility to execute SQL and update DB2 tables directly.

C

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

```
ftp://www.redbooks.ibm.com/redbooks/SG246486
```

Alternatively, you can go to the IBM Redbooks Web site at:

```
ibm.com/redbooks
```

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246486.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
SG246486.zip	Zipped Code Samples, JCL, Readme file

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder. Open the Readme file: this file contains the description of the other files and the instructions about how to upload and use the files.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 164. Note that some of the documents referenced here may be available in softcopy only.

- *Getting Started with DB2 Stored Procedures: Give Them a Call through the Network,* SG24-4693
- *DB2 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7083

Other publications

These publications are also relevant as further information sources:

- *z/OS V1R6.0 MVS Setting Up a Sysplex,* SA22-7625
- *z/OS MVS Programming: Resource Recovery,* SA22-7616-04
- *z/OS V1R4.0 MVS System Management Facilities (SMF),* SA22-7630-06
- *z/OS V1R2.0 MVS Writing TPs for APPC/MVS,* SA22-7621-02
- *CICS TS Intercommunication Guide Version 2 Release 2*, SC34-6005-06
- *CICS Transaction Server for z/OS V2.3 CICS Customization Guide,* SC34-6227-03
- *CICS Resource Definition Guide*, SC34-6228-00
- *CICS TS Operations and Utilities Guide,* SC34-6014-00
- *CICS TS Installation Guide*, GC33-1681-43
- *CICS Applications Programming Guide, SC33-1687*
- *IMS Installation Volume 1: Installation Verification Version 8*, GC27-1297-02
- *IMS Application Programming: Transaction Manager Version 8*, SC27-1289-02
- *IMS Administration Guide: Transaction Manager Version 8*, SC27-1285-02
- *DB2 Universal Database™ for OS/390 and z/OS, Installation Guide, Version 7, GC26-9936-02*
- *OS/390 eNetwork Communications Server: SNA Resource Definition Reference,* SC31-8778-04

Online resources

These Web sites and URLs are also relevant as further information sources:

- Location for the source programs used to create the APPC workload scenarios

  `http://www.redbooks.ibm.com/redbooks/SG246486`

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM Redbooks

Implementing and Managing APPC Protected Conversations

(0.2"spine)
0.17"<->0.473"
90<->249 pages

Implementing and Managing APPC Protected Conversations

IBM®

Redbooks

APPC Protected Conversation environment setup

Operating in an APPC Protected Conversation environment

Sample scenarios

APPC Protected Conversation is a function provided by the operating system to exploiters running on z/OS. This function improves data integrity in distributed processing environments by enabling participation in the two-phase commit protocol.

This IBM Redbook provides system programmers with a solid understanding of the APPC Protected Conversation environment. It describes how to upgrade your environment to support protected conversations, how to configure protected conversation exploiters, how to operate in this environment, and how to manage resources. Sample scenarios illustrate how transactions are executed in a protected conversation environment, and how they fail. Design considerations for avoiding failures are also included, as well as a discussion of tools and utilities for monitoring and tuning your APPC environment. Detailed installation definitions are provided for protected conversation exploiters (IMS, CICS, and DB2).

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6486-00 ISBN 0738491942